5-2014

Decision Making Using Trust and Risk in Self-Adaptive Agent Organization

Kamilia Ahmadi
Utah State University

DECISION MAKING USING TRUST AND RISK IN SELF-ADAPTIVE AGENT ORGANIZATION

by

Kamilia Ahmadi

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Vicki H. Allan Major Professor	Dr. Nicholas Flann Committee Member
Dr. Dan Watson Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2013

ABSTRACT

Decision Making Using Trust and Risk in Self-Adaptive Agent Organization

by

Kamilia Ahmadi, Master of Science

Utah State University, 2013

Major Professor: Dr. Vicki H. Allan
Department: Computer Science

Self-organizing, multi-agent systems provide a suitable paradigm for agents to manage themselves. We demonstrate a robust, decentralized approach for structural adaptation in explicitly modelled problem-solving agent organizations. Based on self-organization principles, our method enables the agents to modify their structural relations to achieve a better completion rate of tasks in the environment. Reasoning on adaptation is based only on the agent's history of interactions. Agents use the history of tasks assigned to their neighbors and completion rate as a measure of evaluation. This evaluation suggests the most suitable agents for reorganization (Meta-Reasoning). In the first part of this research we propose Selective-Adaptation method. Our Selective-Adaptation has four different approaches of Meta-Reasoning, which are 1) Fixed Approach, 2) Need-Based Approach, 3) Performance-Based Approach, and 4) Satisfaction-Based Approach along with a Reorganization method, which needs less data but makes better decisions.

Interaction between agents is one of the key factors in Multi-Agent societies. Using interaction, agents communicate with each other and cooperatively execute complex tasks which are beyond the capability of a single agent. Cooperatively executing tasks may endanger the success of an agent by selecting poor choices for peers. Therefore, agents need to have a better evaluation mechanism in selecting peers. Trust is one of the measures

commonly used to evaluate the effectiveness of agents in cooperative societies. Since all of the interactions are subjected to uncertainty, the risk behavior of agents is considered as a contextual factor in decision making. In the second part of this research we propose the concept of adaptive risk and the use of recommendation-based trust in our adaptive society. We also introduce the agent's strategy and propose an algorithm which helps agents to make decision in an adaptive society using adaptive risk and recommendation-based trust.

(79 pages)

PUBLIC ABSTRACT

Decision Making Using Trust and Risk in Self-Adaptive Agent Organization

Kamilia Ahmadi

Interaction between agents is one of the key factors in multi-agent societies in order to cooperatively execute complex tasks which are beyond the capability of a single agent. In the self-adaptation society, agents try to keep best neighbors around themselves. Agents use history of past iterations to evaluate their neighbors and locally modify their structural links. Main characteristics of this model are its decentralization, dynamicity and no need of external control. This research at first deals with implementing self-adaptive agent organization. Then it focuses on evaluating cooperation peers in decision making. Trust as an evaluation mechanism lies at the core of all interactions in the agent organizations. Agents build trust about a target based on their direct experiences and third party recommendations. Since agents have different perception of trust, context of decision should be taken into account. Uncertainty is one of the main contextual factors that affect agents' decisions. The aim of the second work is helping agents have better evaluation in decision making utilizing recommendation-based trust and adaptive risk.

CONTENTS

Page

ABSTRACT iii

PUBLIC ABSTRACT v

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

2 EFFICIENT SELF-ADAPTING AGENT ORGANIZATION 5
- 2.1 Abstract 5
- 2.2 Introduction 5
- 2.3 Previous Work 6
- 2.4 Our Proposed Model 12
- 2.5 Experiments and Results 23
- 2.6 Conclusion and Future Work 28

3 DECISION MAKING USING RISK AND TRUST IN SELF-ADAPTING MULTI-AGENT SYSTEMS 30
- 3.1 Abstract 30
- 3.2 Introduction 30
- 3.3 Previous Work 33
- 3.4 Our Proposed Model 34
- 3.5 Experiments and Results 45
- 3.6 Conclusion and Future Work 57

4 CONCLUSIONS 60

REFERENCES 63

Appendix 68
- CHAPTER A Reprint Permission 69

LIST OF TABLES

Table Page

2.1 Loads and costs of a_x related to different relations 21

LIST OF FIGURES

Figure Page

2.1 Diagram of the task tree. 9

2.2 Organizational structure. 9

2.3 Process of assigning a task to an agent. 10

2.4 Diagram of possible actions. 12

2.5 Pseudocode of Fixed Approach. 15

2.6 Pseudocode of Need-Based Approach. 16

2.7 Pseudocode of Performance-Based Approach. 17

2.8 Pseudocode of Satisfaction-Based Approach. 18

2.9 Pseudocode of Reorganization part. 19

2.10 Pseudocode of task scheduling. 22

2.11 Effect of task scheduling. 24

2.12 Profit of different approaches over time. 25

2.13 Effect of Agent Shock on different approaches. 27

2.14 Effect of Task Shock on different approaches. 29

3.1 Structure of the task tree. 35

3.2 Example organizational structure. 37

3.3 Task passing mechanism. 37

3.4 Diagram of possible actions. 39

3.5 Profit comparison under three different workloads. 47

3.6 Queue length and standard deviation of trust-based and no-trust approaches in low workload. 48

3.7 Queue length and standard deviation of trust-based and no-trust approaches in medium workload. 49

3.8 Queue length and standard deviation of trust-based and no-trust approaches in high workload. 50

3.9 Comparison of task failure rate of trust-based and no-trust approaches in different workloads. 51

3.10 Contribution of risk seeking, risk neutral, and risk averse agents under different workloads. 53

3.11 Risk changing behavior of risk seeking agents. 54

3.12 Risk changing behavior of risk neutral agents. 55

3.13 Risk changing behavior of risk averse agents. 55

3.14 Effect of adaptive risk in reducing task failure, better task completion rate and higher profit. 57

CHAPTER 1

INTRODUCTION

Multi-agent systems consist of intelligent agents and their environment. Intelligent agents, which can be software agents, robots or humans, interact with their environment and accomplish actions in order to reach their goals. Agents are autonomous, self-interested and goal-driven, which makes them capable of solving problems. Multi-agent systems are designed to solve problems that are beyond the individual capabilities or knowledge of each single agent. Therefore, cooperation between them is necessary for achieving goals. Agents have two main capabilities. The first capability is autonomously deciding what actions to take that will lead them toward their own goal. The second capability is interacting with other agents in the environment. Since the multi-agent systems we consider are designed for cooperatively solving problems, the notion of interaction between agents is a central point for the design of multi-agent applications. Interaction allows them to find each other and exchange information. Interaction also includes social activities like cooperation, negotiation and argumentation in terms of reaching their goal [1–3].

As computing systems get larger, their complexity and the number of components increase. This makes it difficult for designers to anticipate all the needs of the whole system at design time. The need for self-management is essential to the success of such large, complex systems. Autonomous systems, capable of decentralized self-organization, have been proposed as a solution to managing complex computing systems that must deal with node failure [4–7]. Responding to their own history, individual agents exhibit the ability to adapt to changing circumstances and have the ability to evaluate reorganization options and decide when reorganization is necessary. Social interactions and success in jointly solving problems determine desirable structure. Expected properties of a self-organization system are as follows:

- Adaptation needs to be performed without external control: Agents autonomously establish and prioritize relationships within the organization.
- Adaptation is performed in a decentralized fashion: Multi-agent systems are distributed in nature. Thus, adaptation should also be distributed. All interactions between agents are local and all members control their own relationships. The system is robust against failure because there is not a central point of management or power in the system.
- Adaptation needs to be dynamic: Adaptation is a continuous activity of the system. Agents gradually make the changes, and self-management is expected to evolve with time.

One of the desirable properties of multi-agent systems is the autonomy of the entities. Autonomy makes the predictability of other agents' behavior impossible, both because of the difference in goals and the methods to achieve the goals. When the behaviors are at cross purposes, there is increased vulnerability to the system due to optimistic promises. The goals of one agent may negatively impact the achievement of the goals of another agent. Thus, some sort of societal control is needed in a society of autonomous agents, as is true in human societies. The term soft security refers to security which protects from harm in unobtrusive ways [8]. Soft security deals with preventing the repetition of undesirable behaviors. Trust as soft security is one of the most successful social controls in human societies.

From a local point of view, trust lies at the core of all interactions in the agent system. A task assigned to an agent may have components which need to be done by another agent. An agent will ask another agent to complete part of the task (delegation) based on both capabilities and availability. Cooperation between agents requires each agent to rely on other agents' capabilities and competency to achieve goals. However, cooperation may jeopardize an agent's success by relying too heavily on the statements and actions of its neighbors [9–12]. Determining which agents to select for delegation of tasks is a very complicated decision for agents. The agent should have a clear evaluation of the partners

they choose. Therefore, the concept of trust is one the fundamental necessities in the society and resolves some of the uncertainty in interactions. Agents build trust about their neighbors and use trust to make decisions about partners.

In the global point of view, trust works as a social control mechanism. Agents who repeatedly fail to complete promised tasks lose the chance of cooperating in tasks and are socially excluded from activities. This creates a form of supervision on the whole system in which all entities are involved [13].

In choosing a partner, trust is an important consideration, but the perception of trust differs from agent to agent. Agents do not necessarily agree on the definition of poor behavior as it is dependent on context. In addition, agents may be willing to risk a lesser result if their circumstance is tolerant to failure or their partner options are limited. Risk and uncertainty exist in all interactions of societies with non-deterministic behaviors. Most trust models do not pay much attention to the context of the organization in which interactions take place. The risk attitude of an agent as well as the rewards and costs associated with actions are key factors in decision making, and must be considered as part of the context of the decision. Situations like this necessitate having a mechanism that would allow individualized evaluation of risk in the environment. Risk-seeking agents can handle a higher amount of uncertainty, while risk-averse agents are more cautious. While trust is necessary for successful delegation and social control, it is not sufficient on its own due to the context based nature of many decisions agents make [14, 15].

The first part of this research deals with implementing self-adaptation in agent organization along with proposing four different methods namely: Fixed Approach, Need-Based Approach, Performance-Based Approach and Satisfaction-Based Approach. These methods are strategies of agents for adaptation. Since one of the purposes of self-adaptation is robustness against failure, we test our model with two types of shocks to see the effectiveness of our self-adaptive system under unexpected circumstances.

The second part of this research focuses on studying the behavior of agents in this adaptive society (built in the first part). We propose an algorithm for decision making which

allows agents to utilize the concepts of trust and adaptive risk. Often risk is categorized by three different categories: risk-seeking, risk-neutral and risk-averse [16–19]. The concept of Adaptive Risk we propose, which introduces a range for risk behaviors. In addition, adaptive risk gives agents the opportunity of updating their risk measure based on the effectiveness of prior decisions. Also we propose the concept of Agents' Strategy, which is a strategy to helps them with prioritizing tasks in their work queue. Furthermore, agents have the opportunity of updating their strategy. We use a Recommendation-Based Trust model, which has been frequently used for studying the concept of trust in multi-agent systems [13, 18, 20–22].

CHAPTER 2

EFFICIENT SELF-ADAPTING AGENT ORGANIZATION

[1]

2.1 Abstract

Self-organizing multi-agent systems provide a sui paradigm for agents to manage themselves. We demonstrate a robust, decentralized approach for structural adaptation in explicitly modelled problem solving agent organizations. Based on self-organization principles, our method enables the agents to modify their structural relations to achieve a better completion rate of tasks in the environment. Reasoning about adaptation is based only on the agent's history of interactions. Agents use the history of tasks assigned to their neighbors and completion rate as a measure of evaluation. This evaluation suggests the most suitable agents for reorganization (Meta-Reasoning) and then updates relations (Reorganization) with the target neighbors. Our Selective-Adaptation has four different approaches of Meta-Reasoning, which are 1) Fixed Approach, 2) Need-Based Approach, 3) Performance-Based Approach, and 4) Satisfaction-Based Approach for Meta-reasoning part and a Reorganization phase, which needs less data but makes better decisions

2.2 Introduction

A multi-agent system consists of interacting intelligent agents and their environment. Agents can be software agents, robots, or humans. Multi-agent systems solve problems that are difficult or impossible for an individual agent to solve alone. In multi-agent systems,

[1] Ahmadi K. and Allan V. H (2013).
Efficient Self-Adapting Agent Organizations.
In Proceedings of the 5th International Conference on Agents and Artificial Intelligence, pages 294-303
DOI: 10.5220/0004261902940303

interaction between agents is one of the important factors, which allows them to find each other and exchange information [2, 3].

Social interaction and success in jointly solving problems determines a desirable structure for the organization of agents. The task environment contains a stream of tasks requiring some services, and agents need to provide these services by providing required resources. The number of links and the specific connections are designed to minimize communication overhead and facilitate task completion.

Autonomous systems, capable of de-centralized self-organization, have been proposed as a solution for managing complex computing systems that must deal with node failure and dynamic problem characteristics. Responding to their own history of interactions, individual agents exhibit the ability to modify the organizational structure. Our adaptation method is based on the agents forging and dissolving relations with other agents. Agents use the history of tasks assigned to their neighbors and the degree of successful completion of these tasks as a measure of evaluation. The system evaluates existing links for possible increase or decrease in the overall performance. After finding the target neighbors for reorganization, the agent may decide to change the two-way relationship with them or replace the target agent with another agent for probable improvement [4, 6].

Various approaches promote self-organization, like reward-based mechanisms for selfish agents [7], stigmergy (indirect coordination through the environment) [5], reinforcement mechanisms, and cooperative actions of agents. Each of these approaches has advantages and disadvantages, but none of them directly deals with organization structure. Self-organized systems are decentralized, without any external control. Such autonomic systems are more robust as there will not be a single point of failure.

2.3 Previous Work

Hubner et al. introduced the MOISE framework [23]. MOISE is a top-down framework which utilizes a group of expert agents for reorganization. The expert group is in charge of monitoring the system and finding the deficiencies of current structure; this group also has the capability to propose changes and the authority to implement them. In this model, a

group of complex agents have management and reorganization capabilities and the rest of the agents are just simple working agents. This framework does not satisfy our requirements to have a decentralized adaptation which involves all agents in management and reorganization processes.

Bou et al. introduced ISLANDAR model [1]. In this model, a group of central authority agents regulate norms of the organization. These norms guarantee the accomplishment of the goals of the organization. This central management also constantly modifies the organizational structure in order to satisfy the predefined norms of the system. First of all, this model is centralized, which is not in keeping with our ideal model. Furthermore, as the system grows in size, defining the norms becomes more complex because the central authority agents need a lot of information to consider all possible situations of the system. Finally, not all of the agents are involved in the adaptation process.

Hoogendoorn used a model named AGR [5]. In AGR, the organization is considered as a graph in which agents are nodes and the relationships between them are the edges of the graph. A central authority tries to identify the bottleneck of the system. The adaptation process aims to improve performance of the system by duplicating the roles and associated links of agents to increase capacity in certain areas. The problem of this model is not only its centralized behavior, but also its lack of a mechanism for removing redundant links.

In the Barton work, the network structure is composed of agents with a given skill set and connections between agents [24]. Tasks requiring a set of skills are introduced into the system. Agents communicate with other agents through links in their surrounding network, which is the agent's local neighborhood. A set of agents form a coalition to complete each task. In this model, all completed tasks have equal utility, while uncompleted tasks have zero utility. Agents attempt to reorganize themselves to improve the utility of the system. Barton evaluates several approaches in this work. Egalitarian approach establishes connections between agents with relatively few connections. Inventory approach connects agents possessing a needed skill in the neighborhood. Structural approach seeks to connect agents with the largest number of connections. This model differs from our model in that a

tree of services (subtasks)is not considered for modelling tasks. We have different types of dependency links between subtasks, which are nodes on the tree. This model permits only one kind of relationship between agents, while we have different authority links between agents. Based on the links they have, each agent has different roles and different duties along with access to various amounts of information.

Miralles structures the problem as a set of resources that work together to share data [25]. A separate meta-level is in charge of adaptation. A peer can potentially contact any other agent, but typically, it interacts with a small number of them. Agents reorganize in response to changes in connection quality or information flow. Each connection is limited in terms of the number of units of data that can be sent in a time step. Having a meta- level in charge of management contradicts with the decentralized property that is a desirable property of the system.

Kota et al. represent the task environment as a dynamically incoming stream of tasks requiring multiple services [4, 6]. There is sequential dependency between tasks. Kota represents tasks as a tree of service instances (SIs) in which the parent SI must be completed before the child SI. Figure 2.1 demonstrates a tree of task dependencies. Nodes represent a service instance (SI). Arrows represent a dependency relationship. A tuple represents the services (skills) required and the amount of computation needed for each of the five SIs. Since finishing the task requires multiple services, agents pass the task between themselves in order to complete all of the services required. The Kota model assigns tasks to the agents randomly, and the assigned agent utilizes its neighbors including subordinates, peers, superiors and acquaintances to accomplish the task. The task is complete when its entire tree of SIs has been executed.

In the Kota work, agents are known to each other with three levels of relationship: (a) acquaintance (knowing existence, but having no interaction), (b) peer (low frequency of interaction) and (c) superior-subordinate (preferred interaction). The superior-subordinate relation is an authority relation as it depicts the authority held by the superior agent over the subordinate agent. The peer relation is present between agents who are equal in authority

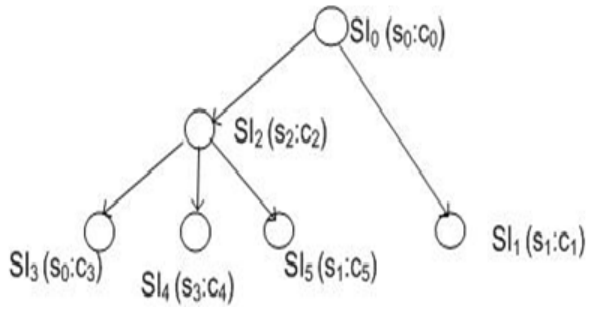

Figure 2.1: Diagram of the task tree.

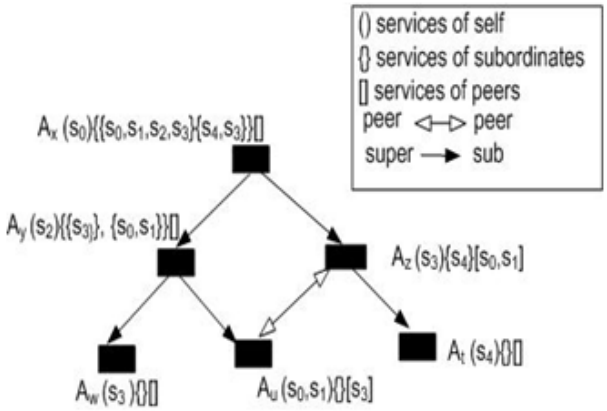

Figure 2.2: Organizational structure.

with respect to each other. The type of relationship between agents determines both the task allocation the amount of information agents know about each other. Structure of the organization regulates the interactions between agents. Figure 2.2 shows an example of the organizational structure of agents. Agents (nodes) keep track of their own service(s) (shown in parentheses), the services of their subordinates (shown in braces as a possibly nested set of services), and the services of peers (shown in square brackets).

In the Kota model, every agent has a fixed number of services it can provide and a known computational power. Thus, an agent is of the form $A_x =< s_x, c_x >$ where $s_x \subseteq S$ (S is the complete list of services) and c_x is the agent's capacity in terms of computational units in a time step. An agent prefers to allocate the subtasks to its subordinate agents as subordinate agents give priority to tasks assigned by the superior. An agent is always try

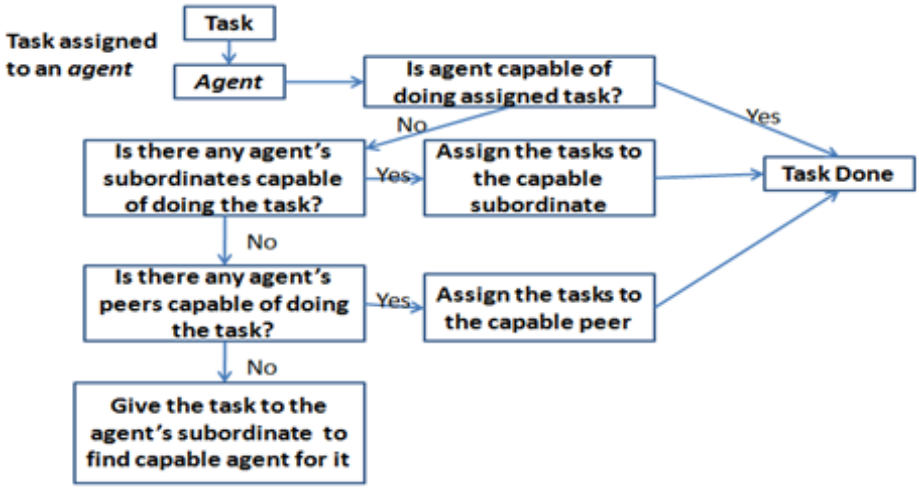

Figure 2.3: Process of assigning a task to an agent.

to execute an SI if it contains the service and available computational power. If it is not possible for the agent to execute that SI, it can delegate it to one of its neighbors.

Figure 2.3 shows the process of assigning a SI. Each agent can respond to only one request per time-step. Therefore, agents store requests in a waiting queue. Requests in a waiting queue are considered in a first-come, first-served basis. Each task has a deadline associated with it. If the agent spends time on a task that is not finished, it does not get reward and the agent wasted its resources. If it does not attempt the task, there is no penalty as there was no wasted effort. Also, each task has an estimated amount of expected time. The utility of the task decreases if it takes more than the estimated expected time. Equation 2.1 shows the relationship between utility and time. Here t stands for time.

$$earnedU_{task} = AssignedUtility_{task} - (t_{task}^{taken} - t_{task}^{expected}) \tag{2.1}$$

When an agent is consistently looking for another agent to perform a given service, it is motivated to reorganize to form a direct relationship with an agent providing that service. This process is called adaptation. This process seeks continuously to improve the profit of the system. Agents can adapt only locally and change only their own links. Though based on local adaptation by the agents, the method should lead to the benefit of the organization

as a whole. The Adaptation process consists of two main parts, named Meta-Reasoning and Reorganization. Meta-reasoning decides about which agents should be considered by $agent_x$ for reorganization. The number of agents considered for reorganization at time t, k_t, is computed as showed in Equation 2.2.

$$k_t = max \begin{cases} 1 \\ \frac{(L_x - l_x)}{R} \\ acqts_x \times \frac{changed_{x,t-1}}{k_{t-1}} \end{cases} \tag{2.2}$$

In Equation 2.2, L_x is the computational capacity of the $agent_x$, l_x is the current load on the agent and R is the reorganization load coefficient, denoting the amount of computational units consumed by an agent while changing a single relation. $acqts_x$ represents the number of acquaintances of $agent_x$, $changed_{x,t-1}$ denotes the number of changed relations of $agent_x$ in the previous iteration and k_t1 denotes the k value used in the previous iteration. Based on Equation 2.2, at least one of the $agent_x$'s neighbors is considered for reorganization in each iteration. The second term, which is $\frac{(L_x - l_x)}{R}$, indicates that reorganization can consume the remaining computational capacity of $agent_x$ in current iteration, regardless of the need for that much reorganization. The third term, $acqts_x \times \frac{changed_{x,t-1}}{k_{t-1}}$, estimates the number of relations which should be considered for reorganization based on the history of past iterations.

After finding the value of k_t, $agent_x$ randomly picks k_t agents from the list of its neighbors including its peers, subordinates and acquaintances for reorganization. In the Reorganization part of the Kota method, $agent_x$ evaluates its relations with considered agents in the Meta-Reasoning part. This evaluation considers changing those relations to another type of relation in order to increase profit. Figure 2.4 shows the possible actions between two agents, dependent on the current relationship. Then $agent_x$ evaluates the utility of each of the possible actions. After calculating the utility of each action, $agent_x$ selects the best reorganization action.

One of the deficiencies of the Kota method is the lack of a suitable task scheduling

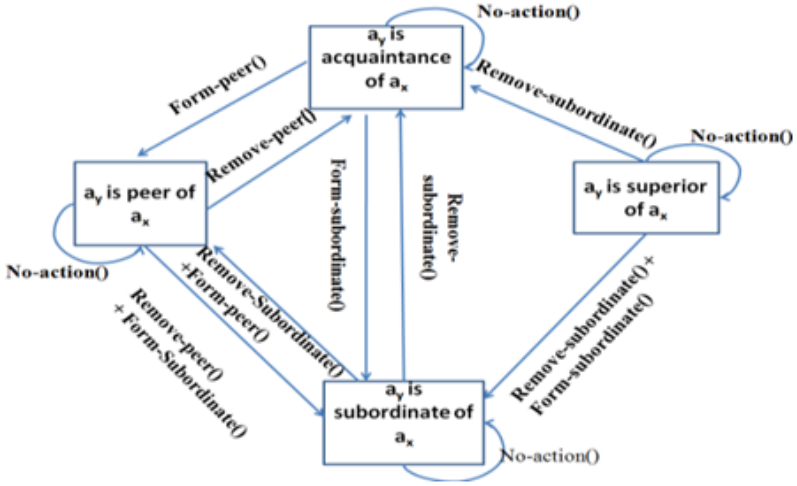

Figure 2.4: Diagram of possible actions.

algorithm; tasks are assigned to agents randomly. Randomly assigning tasks increases the load on the assigned agent when it cannot provide needed capabilities. Figure 2.3 shows the process of finding a capable agent in such a case. This approach adds communication cost to the system and keeps the assigned agent busy finding a capable agent. Therefore, an intelligent way of task scheduling is needed in order to improve the profit and reduce the cost. Other deficiencies include randomly choosing neighbors for reorganization in Meta-Reasoning part and complex method for evaluating possible actions in the Reorganization process.

2.4 Our Proposed Model

In this research, we adopt the structural constraints of the Kota research [4], but focus on the deficiencies of its model. In our method, Selective-Adaptation, each agent selects agents among its neighbors based on different approaches. The adaptation part of Selective-Adaptation method is composed of two parts which are Meta-reasoning and Reorganization. The relation of an agent and its neighbors is based on the two-way task passing. Most of the time agents utilize the capabilities from their neighborhood subordinates and peers; however, there are some cases in which agent's needs are not fulfilled using its peers and subordinates. Therefore, an agent passes the request back to its superior. Its superior is in

charge of finding a suitable agent for this request.

Figure 2.3 summarizes the task passing mechanism; it shows how an agent and its neighbors cooperate in executing different parts of tasks. We term service-providing agents (which are not currently connected as a peer or superior/subordinate) as outsideHelpers. Since the goal of adaptation is promoting relations with agents who are useful to it, we need to consider outsideHelpers. In this model, we use the term neighbors for peers and subordinates of $agent_x$. In our system, an agent's activities includes executing tasks, management (communications, evaluation of neighbors, task passing and updating neighbor information) and Reorganization. Load refers to the computational units used to perform an agent's duties.

2.4.1 Meta-Reasoning

In Meta-Reasoning, each agent determines the number of agents from its neighborhood which should be considered for reorganization. Determining this number is critical because evaluating too many agents wastes the resources of the current agent. In addition, the agent needs to find out which neighbors to consider for reorganization. Meta-Reasoning used in this research is a history-based process and utilizes different approaches namely Fixed Approach, Need-Based Approach, Performance-Based Approach and Satisfaction-Based Approach. We discuss these approaches in the following subsections.

Fixed Approach

In the Fixed Approach, agents have an opportunity to evaluate their relationship with their neighbors in all iterations. Our experiments show that the cost of reorganization is one of the most important factors that affect profit of the system. In order to increase the profit, each agent needs the most useful neighbors. The best neighbors are the ones that result in a higher utility for the system. In order to make reasonable decisions about reorganization, the reorganization load coefficient, R, has been defined in [4]. Evaluating many relationships might exhaust the resources of the agent. Thus, $agent_x$ has to restrict the set of its neighbors to consider for reorganization. Agents are recharged with computational power every d

time steps. They use the recharge interval to estimate how much of the resource can be consumed in any iteration. Since fuel costs are not negligible, the use of fuel should be wisely monitored. "InitialComp" variable keeps the amount of initial computational power of agent. Since agents firstly execute their assigned tasks in each iteration and then they go to the reorganization phase, the remaining computational power of each iteration after executing tasks can be used on reorganization. By dividing the amount of remaining computational power by R, the number of agents that can be considered for reorganization is determined and stored in k_t. Equations 2.3 and 2.4 show the process of determining k_t. In these Equations, i stands for current iteration and "RemainedComp" indicates the remaining computational power of iteration.

$$RemComp_i = InitialComp_i - load_i \tag{2.3}$$

$$k_t = \frac{RemainedComp_i}{R} \tag{2.4}$$

The number of agents to be considered for reorganization, k_t, should be divided between neighbors and outsideHelpers of $agent_x$. For this division, we use the fraction W_f to determine the proportion of agents in each category based on Equations 2.5 and Equations 2.6.

$$numOutsideHelpers = min \begin{cases} count(outsideHelpers) \\ W_f \times numAgents \end{cases} \tag{2.5}$$

$$numNeighbors = k_t - numOutsideHelpers \tag{2.6}$$

We found that the system reached highest profit when $W_f = 0.3$. Thus, 30 percent of agents we consider for reorganization are from outsideHelpers and the rest are $agent_x$'s neighbors. Figure 2.5 shows the pseudocode of the Fixed Approach algorithm.

Fixed Approach

1.k_t=$agent_x$. calcNumAgentsToConsider();

2.numOutsideHelper= $agent_x$.calcNumOutsideHelpers();

3.numNeighbors=k_t- numOutsideHelper;

4.outsideHelpers= $agent_x$.findOutsideHelpers(numOutsideHelpers);

5.Neighbors=$agent_x$.findNeighbors(numNeighbors);

6.AgentsToConsider= $agent_x$.combine(outsideHelpers, Neighbors);

Figure 2.5: Pseudocode of Fixed Approach.

The strategy of $agent_x$ in selecting the suitable outsideHelpers and neighbors is different. $Agent_x$ calculates earned utility of all of them and ranks them by this attribute. The more utility they have earned, the better rank they have. $Agent_x$ tries to replace some of its inefficient neighbors, but makes a stronger link with outsideHelpers which were helpful in the past.

Need-Based Approach

In the Need-Based Approach, at each iteration, $agent_x$ estimates the number of links it needs to reorganize using two parameters from past iterations: 1) the number of relations considered for reorganization, and 2) the number of relations changed in past iterations. By dividing the number of relations changed by the number of relations considered, $agent_x$ can estimate how many of its past predictions have been accurate. Therefore, k_t estimates the number of agents which need to be considered in the current iteration (t) based on Equation 2.7.

$$k_t = \frac{\sum_{i=t-h}^{t-1} ChangedRelations_i}{\sum_{i=t-h}^{t-1} ConsideredRelations_i} * numNeighbors \tag{2.7}$$

After deciding the number of agents to be considered, (k_t), $agent_x$ needs to divide k_t between its neighbors and outsideHelpers using Equations 2.5 and 2.6. By testing different values for W_f, the system reaches the highest profit where W_f=0.3. Neighbors and outsideHelpers

get their rank based on their earned utility for $agent_x$. Figure 2.6 shows the pseudocode of this approach.

Need-Based Approach

1. k_t=$agent_x$. calcNeedBasedNumAgents();
2. numOutsideHelper= $agent_x$.calcNumOutsideHelper(W_f);
3. numNeighbors=k_t - numOutsideHelper;
4. ousideHelpers= $agent_x$.findOutsideHelpers(numOutsideHelper);
5. neighbors=$agent_x$.findNeighbors(numNeighbors);
6. AgentsToConsider= $agent_x$.combine(ousideHelpers, neighbors);

Figure 2.6: Pseudocode of Need-Based Approach.

Performance-Based Approach

In the Performance-Based Approach, we utilize a measure of performance to decide which agents to consider for reorganization. Each task in the system has an assigned utility. When an agent is assigned a task, this agent can earn the whole utility of the task. Therefore the possible utility $agent_x$ can earn is the sum of the utilities of all tasks it has been assigned as shown in Equation 2.8.

$$possibleUtility = \sum_{task \in Queue} AssignedUtility(task) \tag{2.8}$$

Sometimes agents cannot earn that amount of possible utility due to the features like deadline violations and slowness in completing tasks. The amount of utility earned in comparison with the possible utility available can be a good measure of performance for agents.

$$performance = \frac{\sum_{i=t-h}^{t-1} earnedUtility_i}{\sum_{i=t-h}^{t-1} possibleUtility_i} \tag{2.9}$$

Equation 2.9 shows the measure of performance for each agent based on its history of earning utility. In Equation 2.9, i stands for any previous iteration, and t indicates current iteration.

The history variable, h, indicates the number of past iterations to consider. Variable h is in the range of $[1, t-1]$, where $h = t-1$ considers the history of all iterations so far, and $h = 1$ uses only the history of the last iteration.

In this approach, $agent_x$, finds the performance for all of its neighbors and outside-Helpers. Then $agent_x$ selects W_i ratio of its worst neighbors in terms of performance along with W_o ratio of its best outsideHelpers in terms of their performance. For setting the values of W_i and W_o, we tested the system with different values and compared the results. In this case, system reaches highest profit in W_i=0.25 and W_o=0.15, which means that 25 percent of least efficient neighbors of $agent_x$ along with 15 percent of best outsideHelpers have been selected. If $agent_x$ does not have any outside helpers, it just considers its neighbors for reorganization. Figure 2.7 shows the Performance-Based Approach.

Performance-Based Approach

```
1.neighbors=agentx.findNeighbors();
2.rankedNeighbors =agentx. calcPerformance(neighbors);
3.leastEffecientNeighbors=agentx.leastEffecient(rankedNeighbors,wi);
4. if count(outsideHelpers)>0
5.        outsideHelpers= agentx. findOutsideHelpers();
6.        rankedOutsideHelpers =agentx. calcPerformance(outsideHelpers);
7.        bestOutsideHelpers=agentx.mostEffecient(rankedOutsideHelpers,wo);
8.        agentsToConsider= agentx.combine(leastEffecientNeighbors, bestOutsideHelpers);
9.else
10.       AgentsToConsider= leastEffecientNeighbors;
```

Figure 2.7: Pseudocode of Performance-Based Approach.

Satisfaction-Based Approach

In the Satisfaction-Based Approach, agents decide on the adaptation based on satisfaction. As we mentioned earlier, the relation of an agent and its neighbors is based on subtask passing. Each agent is satisfied with a relation when the corresponding agent is

able to service most of the agent's requests; the stronger neighborhood in terms of providing requests, the more satisfied the agent is.

$$satisfaction = \frac{\sum_{i=t-h}^{t-1} numProvidedRequests}{\sum_{i=t-h}^{t-1} numRequsts} \tag{2.10}$$

When an agent accepts a subtask request, it means that it has the potential to accomplish that subtask. To compare agents in the neighborhood, we define a measure of satisfaction as shown in Equation 2.10. In this equation, i stands for iteration and t indicates current iteration. Variable h is in the range of $[1, t-1]$ as before. In Satisfaction-Based Approach, $agent_x$ calculates the satisfaction of all its neighbors and its outsideHelpers. Based on this measure, $agent_x$ selects W_i ratio of its least satisfactory neighbors to be considered for reorganization. If there are any outsideHelpers, $agent_x$ needs to select W_o ratio of its most satisfactory outsideHelpers for reorganization too. Values of W_i and W_o are set same as Performance-Based Approach. Figure 2.8 illustrates the pseudocode this approach.

Satisfaction-Based Approach

```
1.neighbors=agentx.findNeighbors();
2.rankedNeighbors =agentx. calcSatisfaction(neighbors);
3.leastSatosfactoryNeighbors=agentx.leastSatisfactory(rankedN,wi);
4. if count(outsideHelpers)>0
5.      outsideHelpers = agentx. findOutsideHelpers();
6.      rankedOutsideHelpers =agentx. calcSatisfactory(outsideHelpers);
7.      bestOutsideHelpers=agentx.mostSatisfactory(rankedOutsideHelpers,wo);
8.      agentsToConsider= agentx.combine(leastSatosfactoryNeighbors, bestOutsideHelpers);
9. else
10.       agentsToConsider= leastSatosfactoryNeighbors;
```

Figure 2.8: Pseudocode of Satisfaction-Based Approach.

For some of the approaches, we need to set the value of h. Our experiments show that considering total number of past iterations ($h = t-1$) is too much history and considering the history of last iteration ($h = 1$) may give a little insight about the past. We tested

```
Reorganization Part
1.for agent_y in agentsToConsider
2.      possible=agent_x.findPossibleActions(agent_y);
3.      desiredAction=agent_x.evaluate(possible);
4.      if desiredAction !=no-Action
5.          agent_x.takeAction(agent_y, desiredAction);
```

Figure 2.9: Pseudocode of Reorganization part.

different values for h and our experiments show that in $h = 30$ system reaches highest profit.

2.4.2 Reorganization

The second part of our Selective-Adaptation is Reorganization. Reorganization enables an agent to change its relations with some of its neighbors (kept in "AgentsToConsider" list) which are identified in the Meta-Reasoning step. In this phase, $agent_x$ evaluates all of the possible types of relation for each member of "AgentsToConsider" and changes the relations in order to achieve a higher utility. For each $agent_i$ in this list, $agent_x$ takes the best action among possible actions based on the current relationship and a measure computed from some evaluation functions. Figure 2.4 demonstrates possible actions for two agents based on their current relation. Figure 2.9 shows the pseudocode of the reorganization part. For each $agent_i$ in this list, $agent_x$ takes the best action among possible actions based on the current relationship and a measure computed from some evaluation functions. Figure 2.9 shows the pseudocode of the reorganization part. Reorganization changes the current relation type R_c to a new relation type R_n. Note that if the selected action is "NoAction" then $R_n = R_c$ (which means that relation will not change and this action does not have utility and cost). The most important part of the reorganization approach is evaluating the utility of possible actions and choosing the best one.

Kota method's evaluation function uses parameters like: 1) the number of subtasks assigned by $agent_x$ to $agent_y$, 2)the number of subtasks delegated by $agent_x$ to $agent_y$, 3)

the total number of time-steps that $agent_x$ existed, 4) the number of time-steps that $agent_x$ and $agent_y$ had a superior-subordinate relation or peer relationship, 5) the number of time-steps out of the total time that $agent_x$ had waiting tasks, 6) the total number of subtasks $agent_x$ assigned to other agents and 7) the communication cost due to the delegations from by $agent_x$ to $agent_y$. Kota method's evaluation function is overly complex. Selective-Adaptation needs less data but makes good decisions. The profit of an agent from the action act_d is:

$$profit(act_d) = Utility + Load \tag{2.11}$$

Based on Equation 2.11, the profit of each action consists of two terms, Utility and Load. This equation is used for both forming and removing a relation. In forming a desirable relation, the sign of Utility is positive. As any new relation adds some load to the agent, the sign of Load is negative. The signs are reversed in the case of removing a relation.

In forming a relation, to estimate the utility for $agent_x$ in the process of changing the relation with $agent_y$ (from R_c to R_n), two terms are used: 1) the average utility $agent_x$ earned from agents which were in the same relation type as R_n which is called U_{R_n} and 2) the average utility $agent_x$ earned from $agent_y$ which is called $U_{avgNewAgent}$.

$$Utility_{\,forminRelation} = \frac{U_{R_n} + U_{avgNewAgent}}{2} \tag{2.12}$$

Equation 2.12 shows this approach. For example, if $agent_x$ wants to make a peer relation with $agent_y$, it calculates the average earned utility of its peers so far U_{R_n} along with the average utility earned form $agent_y$ ($U_{avgNewAgent}$). This equation says that for having relation type (R_t) with agent (a_y), the experience of $agent_x$ from that type of relation (R_t) and the history of functionality of agent (a_y) affect the decision. In a case of removing a relation, for calculating Utility, we use Equation 2.12 too, but the value of $U_{avgNewAgent}$ will be set as zero.

$$LoadOnEachAgent = M + C + R \tag{2.13}$$

The second term of Equation 2.11 is Load and includes different types of loads as shown in

Equation 2.13. M is the management load of having a new relation, C is the communication cost and R is the reorganization load coefficient.

Table 2.1: Loads and costs of a_x related to different relations

	M	C	R
a_x is peer of a_y	M_{peer}	C_{peer}	R_{peer}
a_x is subordinate of a_y	-	$C_{subordinate}$	-
a_x is superior of a_y	$M_{superior}$	$C_{superior}$	$R_{superior}$

We used the general term M and C in Equation 2.13, however, the management load and communication cost depends on the type of relation between two agents. Each agent estimates the amount of M and C based on its history of this type of relation if known. Table 2.1 summarizes all the loads that $agent_x$ (a_x) experiences in changing a relation with agent a_y.

2.4.3 Task Scheduling

One of the deficiencies of the Kota method is its inefficient task scheduling algorithm as it assigns tasks to agents randomly. Random task assignment increases the load on the assigned agents when they are not capable. In this case, the assigned agent needs to go to the process of finding capable agent as depicted in Figure 2.3. Random assignment also adds communication cost to the system and keeps the responsible agent busy finding a capable agent among its neighbors. It wastes resources which the agent could use on executing tasks. Therefore, an intelligent way of task scheduling is crucial. In Selective-Adaptation, the system tries to find the most suitable agent for task assigning. Pseudocode of Task Scheduling is outlined in Figure 2.10.

```
Task Scheduling
1.CCList=findCompCapacityAgents(agents);
2.QLList=findQueueLengthAgents(agents);
3.SSList=findServicesSimilairity(Task);
4.suitableAgent=findSuitableAgent (CP,QL,SS);
```

Figure 2.10: Pseudocode of task scheduling.

To find the most suitable agent for each task, we consider: 1) computational capacity of agents, 2) queue length of agents (how busy the agents are) and 3) similarity of services (how many of the services needed for the task can be provided by the agent). The most desirable agent is the one which has the highest computational capacity and similarity of services. In addition, it needs to have a small queue length because if the system assigns a task to a busy agent, the task may wait a long time for execution. Since the goal of the system is to reach a higher profit, wasting time in the queue while there are other agents in the system which can execute the task is not reasonable. The length of the queue is determined by the summation of the cycles required of each task in the queue. Finding the most suitable agent is the duty of the "findSuitableAgent" function in line 4 of the pseudocode depicted in Figure 2.10. This function aims to find an agent which is the best based on the rank of Equation 2.14.

$$rank_i = \alpha_1 \times CC_i + \alpha_2 \times QL_i + \alpha_3 \times SS_i \tag{2.14}$$

In the Equation 2.14, CC stands for computational capacity of each agent, QL indicates queue length and SS stands for similarity of services. An agent which gets the highest rank will be selected as the most suitable agent. Our experiments reach the highest profit when these terms had an equal effect ($\alpha_1 = \alpha_2 = \alpha_3$).

2.4.4 System Evaluation

We evaluate the effectiveness of models based on the performance of their organization based on Profit, which is the summation of the profits of all of the individual agents. We examine the amount of profit per iteration using Equation 2.15 in order to determine if any improvement is achieved. Thus, for finding the profit we need to compute the amount of utility has been earned and total cost of that iteration. The earned utility by a given agent will be found using Equation 2.1. For finding total cost, Equation 2.16 is used. Equation 2.16 shows that costs in the system include reorganization cost and communication cost.

$$Profit = UtilityEarned - TotalCost \tag{2.15}$$

$$TotalCost = ReorganizationCost + CommunicationCost \tag{2.16}$$

Reorganization cost includes evaluating relationships and changing relationships. The process of assigning a task to an agent requires sending and receiving messages to/from that agent. Therefore, these processes also require inter-agent communication which adds to the total cost of the organization.

2.5 Experiments and Results

For our experiments, we compare the results of Selective-Adaptation with other methods. For each method, we averaged the results for 100 simulations of 1000 iterations each.

2.5.1 Effect of Task Scheduling

Figures 2.11 shows the effect of task scheduling on the various adaptation approaches. From top to bottom a) Fixed Approach, b) Need-Based Approach, c) Performance-Based Approach and d) Satisfaction-Based Approach. As it can be seen intelligent approach of task scheduling helps all of the approaches reaching higher profits.

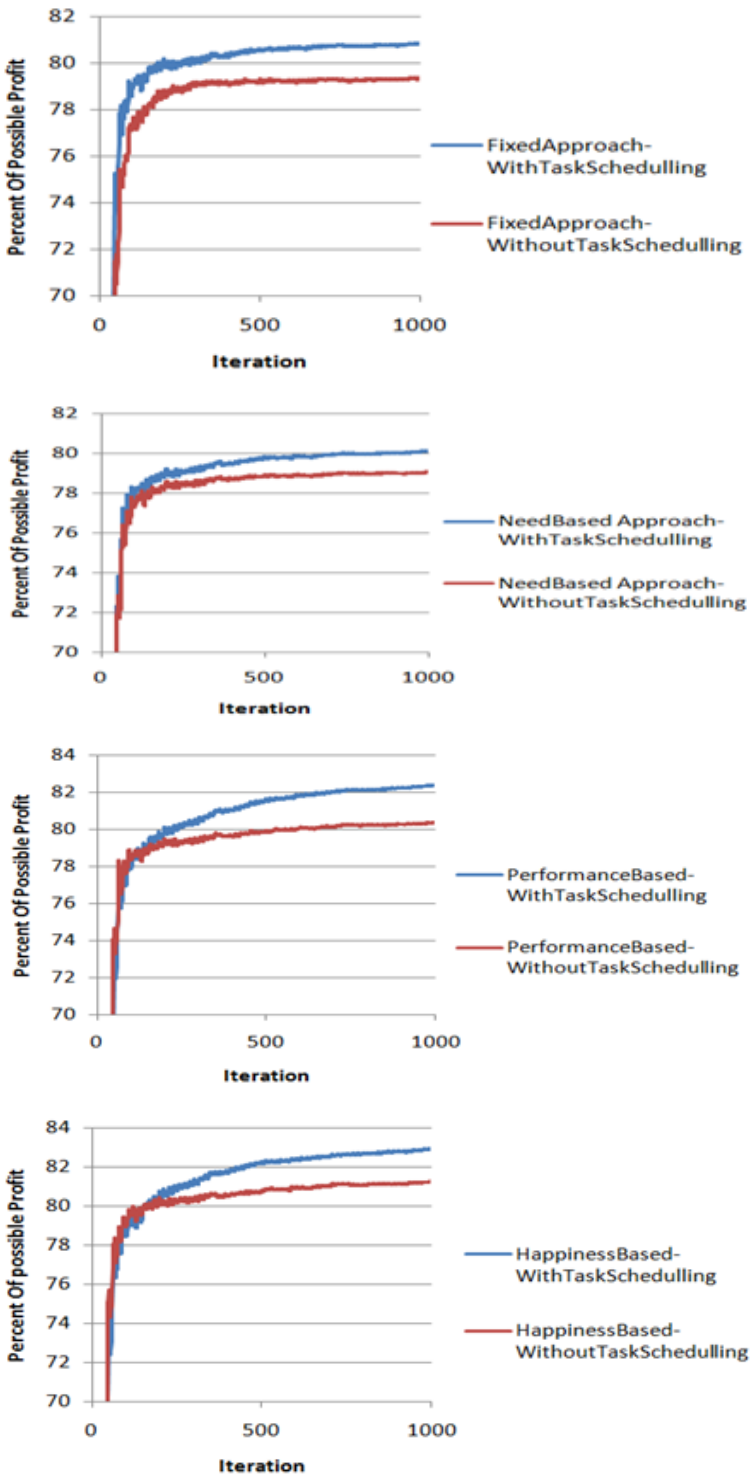

Figure 2.11: Effect of task scheduling.

2.5.2 Profit over Time

As can be seen from Figure 2.12, results show the different approaches of Selective-Adaptation in comparison with Kota method.

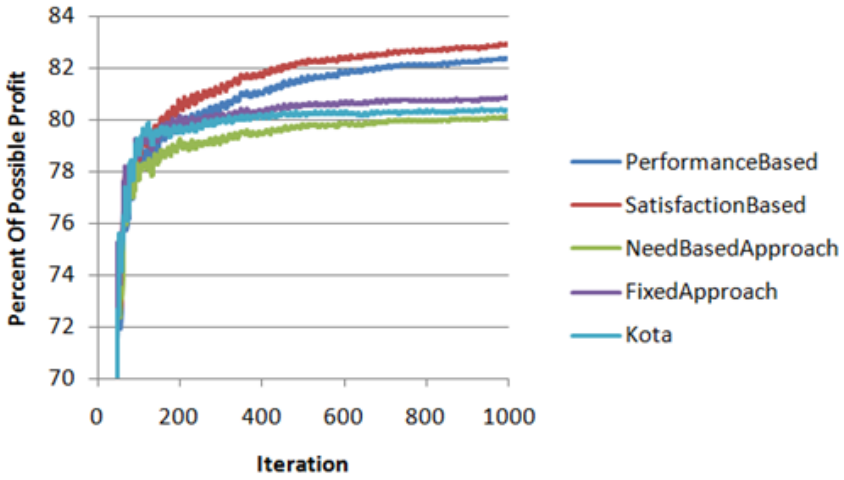

Figure 2.12: Profit of different approaches over time.

An identical reorganization part is used in all Selective-Adaptation approaches; therefore their different profit over time comes from their meta-reasoning approaches. It seems that just using history of past iterations in Need-Based Approach is not effective enough and this approach has the worst performance among all methods. Fixed Approach's overall behavior is similar to Kota but it has better profit due to utilizing intelligent way of selecting neighbors for reorganization. The Satisfaction-based Approach outperforms all other approaches by considering a proper measure of satisfaction for decision making.

2.5.3 Shocks to the System

The aim of the adaptation method is to determine and apply changes in the organization structure in order to improve the performance. Adaptation needs to respond to changes in the environment in a self-organized manner. In order to see the behavior of adaptation facing unpredicted events, we impose shocks upon the system.

Agent Shock Experiment

Agents are associated with particular sets of services. These sets can be overlapping; that is, two or more agents may provide the same service. Different tasks require different amounts of time, and the load requirement is not uniform; therefore agents use a queue to store tasks which are waiting for service. In our experiment with agent shock, every 200 iterations 1/5 of the agents are disabled. Because of this, disabled agents' peers and subordinates bear more load. They need to distribute disabled agents queue among other agents which are capable of doing them. After 15 iterations, new agents will be added to the system to replace the disabled agents. New agents (which need to be incorporated into the organization structure) are added as acquaintances to all other agents. Figure 2.13 shows the effect of Agent Shock on different methods. Also, note the improvement due to adaptation in comparison with the case without adaptation. From top to bottom a) Fixed Approach, b) Need-Based Approach, c) Performance-Based Approach and d) Satisfaction-Based Approach.

Task Shock Experiment

Tasks have some patterns in the dependency links between their subtasks. In this way, the dependencies between the subtasks may follow some frequent orderings (resulting from the dependencies internal to a pattern occurring in several tasks). For the shock test, we defined four different patterns between tasks. Every 200 iterations, we change the pattern to see the effect of the shock. Each agent creates its neighborhood based on the needed capabilities. In the case of changing the pattern, agents face the lack of capability among their neighbors. In such a case, as it can be seen in Figure 2.14, the profit of the system decreases. In Figure 2.14 from top to bottom a) Fixed Approach, b) Need-Based Approach, c) Performance-Based Approach and d) Satisfaction-Based Approach Agents' queues become longer and all of the agents are busy with passing tasks in order to find suitable agents for their assigned tasks.

After some iterations utilizing adaptation, the system compensates for what it lost during shock time by making new neighborhood based on the new pattern. When the

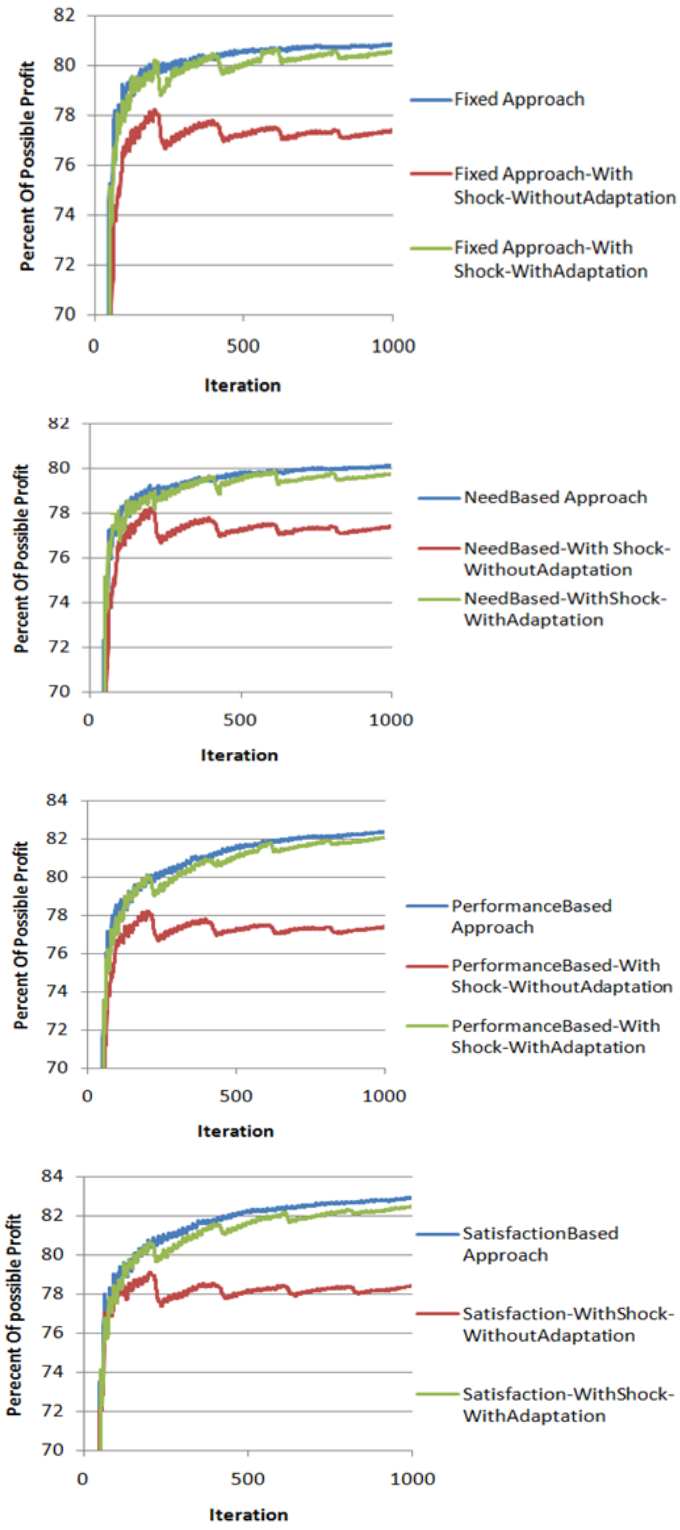

Figure 2.13: Effect of Agent Shock on different approaches.

pattern changed, agents no longer had capabilities in their neighborhood and their superior needed to find them acquaintances that could help. Since all agents are acquaintances to each other, finding the right one to help takes the superior a long time. We decided that the superior can keep a history of acquaintances that helped in the past and ask those for assistance first; if they do not help, the superior finds other acquaintances to help. This decreases the amount of communication in the system, enabling faster recovery from task shock.

2.6 Conclusion and Future Work

In this paper, we propose a new method of adaptation which is called Selective-Adaptation. We demonstrate a robust, decentralized approach for structural adaptation in problem solving agent organizations. Our adaptation method is based on the agents forging and dissolving relations with other agents. Agents use the history of past iterations as a measure of evaluation. This method consists of two main parts which are Meta-Reasoning and Reorganization. In the Meta-Reasoning, every iteration each agent selects some of its neighbors for reorganization based on different approaches like 1) Fixed Approach, 2) Need-Based Approach, 3) Performance-Based Approach, and 4) Satisfaction-Based Approach. After selecting neighbors, the agent tries to find all of the possible actions between itself and target agent based on the current relationship between them. Then the agent evaluates all of the possible actions and selects the best one in terms of its estimated utility. This method can successfully handle unexpected stress tests (shocks) to the system, along with showing higher profit in comparison with other existed method of self-organization (Kota). Possible future works include restricting agents' resources like the amount of memory agents can use for keeping information about others and network bandwidth.

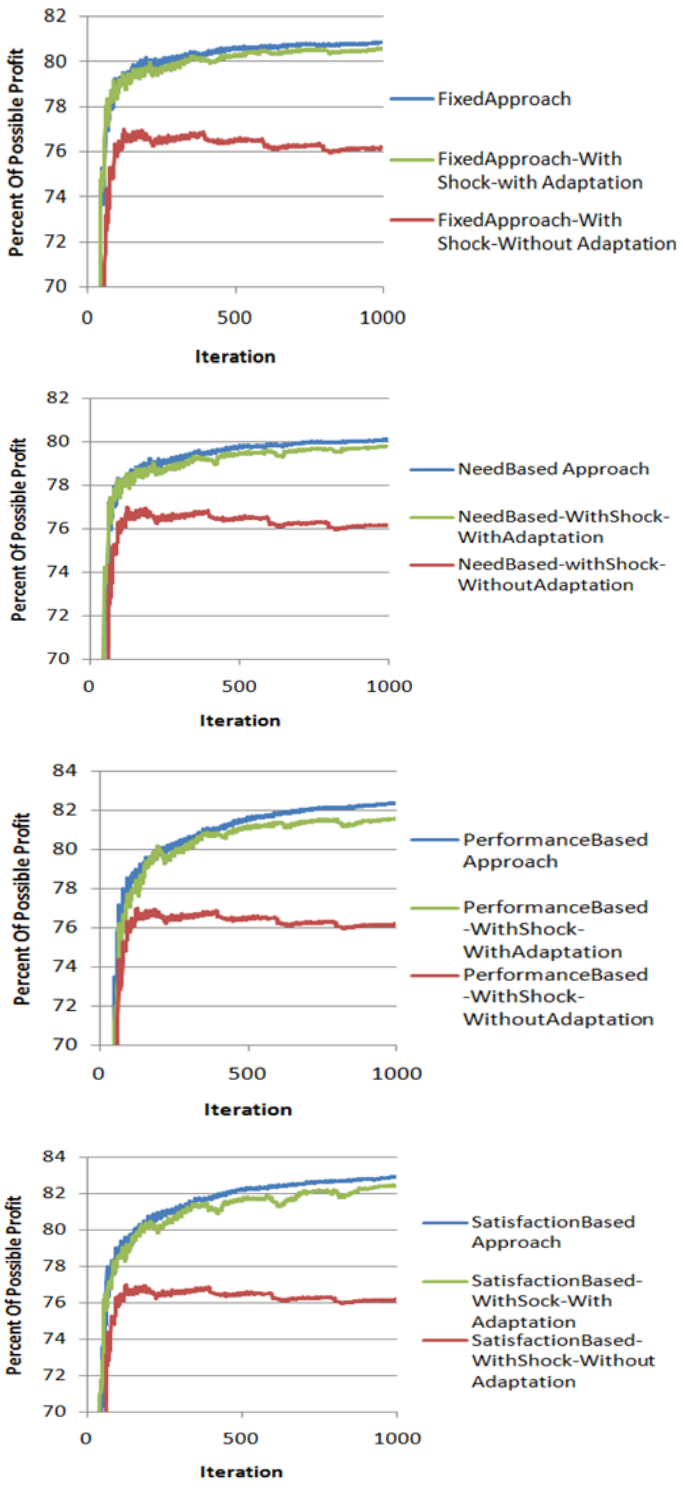

Figure 2.14: Effect of Task Shock on different approaches.

CHAPTER 3
DECISION MAKING USING RISK AND TRUST IN SELF-ADAPTING MULTI-AGENT SYSTEMS

3.1 Abstract

Interaction between agents is one of the key factors in multi-agent societies. Using interaction, agents communicate with each other and cooperatively execute complex tasks which are beyond the capability of a single agent. Cooperatively executing tasks may endanger the success of an agent if it attempts to cooperate with peers that are not proficient. Therefore, agents need to have an evaluation mechanism to select peers for cooperation. Trust is one of the measures commonly used to evaluate the effectiveness of agents in cooperative societies. Since all interactions are subject to uncertainty, the risk behavior of agents as a contextual factor needs to be taken into account in decision making. In this research, we propose the concept of adaptive risk along with agent's strategy and propose an algorithm which helps agents to make decisions in an adaptive society, utilizing adaptive risk and recommendation-based trust. Our Trust-based decision making increases the profit of the system along with higher task completion rate and lower task failure in comparison with the no-trust model in which agents do not utilize evaluation mechanisms for choosing their cooperation peers.

3.2 Introduction

multi-agent systems consist of intelligent agents and their environment. Intelligent agents, which can be software agents, robots or humans, interact with their environment and accomplish actions in order to reach their goals. Agents are autonomous, self-interested and goal-driven, which makes them capable of solving problems. Multi-agent systems are

designed to solve problems that are beyond the individual capabilities or knowledge of each single agent. Therefore, cooperation between them is necessary for achieving goals.

Agents have two main capabilities. The first capability is autonomously deciding what actions to take that will lead them toward their own goal. The second capability is interacting with other agents in the environment. Since the multi-agent systems we consider are designed for cooperatively solving problems, the notion of interaction between agents is a central point for the design of multi-agent applications. Interaction allows them to find each other and exchange information. Interaction also includes social activities like cooperation, negotiation and argumentation in terms of reaching their goal [2].

One of the desirable properties of multi-agent systems is the autonomy of the entities. Autonomy makes a sure knowledge of other agents' behavior impossible, both because of the difference in goals and the methods to achieve the goals. When the behaviors are at cross purposes, there is increased vulnerability to the system due to optimistic promises. The goals of one agent may negatively impact the achievement of the goals of another agent. Thus, some sort of control is needed in a society of autonomous agents, as is true in human societies. The term *soft security* refers to security which protects agents (and the system as a whole) from harm in unobtrusive ways [8]. Soft security deals with preventing the repetition of undesirable behaviors. Trust as soft security is one of the most successful social controls in human societies.

From a local point of view, trust lies at the core of all interactions in the agent system. A task assigned to an agent may have components which need to be done by another agent. An agent will ask another agent to complete part of the task (delegation) based on both capabilities and availability. Cooperation between agents requires each agent to rely on other agents' capabilities and competency to achieve goals. However, cooperation may jeopardize an agents' success by relying too heavily on the statements and actions of its neighbors [18]. Determining which agents to select for delegation of tasks is a very complicated decision for agents. The agent should have a clear evaluation of the partners it chooses. Therefore, the concept of trust is one of the fundamental necessities in the society and resolves some

of the uncertainty in interactions. Agents build trust about their neighbors and use trust to make decisions about partners.

In the global point of view, trust works as a social control mechanism. Agents who repeatedly fail to complete promised tasks lose the chance of cooperating in tasks and are socially excluded from activities. This creates a form of supervision on the whole system in which all entities are involved [13].

In choosing a partner, trust is an important consideration, but the perception of trust differs from agent to agent. Agents do not necessarily agree on the definition of poor behavior as it is dependent on context. In addition, agents may be willing to risk a lesser result if their circumstance is tolerant to failure or their partner options are limited. Risk and uncertainty exist in all interactions of societies with non-deterministic behaviors. Most trust models [12, 16, 20, 26, 27]. do not pay much attention to the context of the organization in which interactions take place. The risk attitude of an agent as well as the rewards and costs associated with actions are key factors in decision making, and must be considered as part of the context of the decision. Risk-seeking agents accept a higher amount of uncertainty, while risk-averse agents are more cautious. While trust is necessary for successful delegation and social control, it is not sufficient on its own due to the dependency of many of the decisions to the contextual factors.

This research focuses on studying agents' decision making in an adaptive society using trust and risk. By "adaptive society" we mean that the organization of agents enables them to modify their structural relations (adaptation) to achieve a better completion rate of tasks in the environment. Adaptation is decentralized and involves all agents in management and reorganization processes. Agents gradually make the changes and this process is a continuous activity of the system. We propose a decision making algorithm which allows agents to utilize the concepts of trust and adaptive risk. Often risk is categorized by three different labels: risk-seeking, risk-neutral and risk-averse [12, 16–18, 20, 27]. The proposed concept of adaptive risk introduces a range for risk behaviors. In addition, adaptive risk gives agents the opportunity to update their risk measure based on the effectiveness of prior

decisions. Also, we propose the concept of agents' strategy, which is a strategy to help them prioritize tasks in their work queue. Furthermore, agents have the opportunity to update their strategy. We use a recommendation-based trust model, which has been frequently used for studying the concept of trust in multi-agent systems [10, 11, 13, 19, 21, 22].

3.3 Previous Work

Falcone and Castelfranchi attempt to represent trust as a mental structure of beliefs by introducing a cognitive approach through trust evaluation [13]. The concept of trust is broken down to different beliefs and the complex interaction between these beliefs builds the body of trust. They define different dimensions of beliefs like competence, disposition, self-confidence and fulfillment. They combine these beliefs to build a trust evaluation mechanism for an entity. While their model is one of the richest models to have been proposed, it is very complex to build the beliefs considering all their defined dimensions. Evaluating the beliefs in order to build trust is another complicated procedure due to the necessity of cognitive information. These problems make this model impractical in most situations.

Grandison and Sloman introduce the SULTAN model [28]. This model was developed to support secure interactions in internet applications. Trust is specified as rules or policies of the system, such as accessing the resources of the system or deciding to accept a user's authorization key. The decision is made based on the rules/policies and risk threshold, which ultimately label some behaviors as distrusted and too risky. The problem of this model is that all the knowledge about risk, rules, and history of trust is centralized within the system. Therefore, it is not applicable to our decentralized adaptive model.

Griffiths employs a simple threshold-based decision-making approach that divides the agents into four categories: trust, distrust, untrust and undistrust [29]. The author claims that this categorization ensures a degree of trust for delegations. There are criticisms on this model. Firstly, finding the optimal threshold for dividing the agents in four groups is complex. Secondly, the threshold should not be the same in all circumstances and should vary depending on the context of different situations. Finally, in this model, agents make decisions using only threshold-based trust, rather than trust within the context of the

decision. These problems make this model inapplicable to our proposed model.

Brunett uses recommendation-based trust in ad-hoc societies [9]. Direct experiences are used to build trust. When an agent has no direct experience with a target, it uses the experiences of other nearby agents to build trust in the target. In highly dynamic societies, agents leave the community or newcomers join frequently. Therefore, building trust based on experiences is not always feasible. Brunett proposes a stereotyping method to build trust. Stereotyping tries to build trust based on predefined features for those agents without relevant experiences. In decision making, agents consider the context of the decision using concepts like risk, reward and cost of the action to satisfy the controls of the system. Although this model satisfies some of our desired properties, there are some mismatches with our proposed model. First of all, our model is a self-adaptive one, which adapts itself based on history and has some sort of dynamicity. The focus of Brunett's model is on ad-hoc organizations, which does not make sense for most history-based decisions. Furthermore, this model divides agents into three categories like risk-seeking, risk-averse and risk-neutral, while we desire agents to have adaptive risk behavior across of range of evaluations. Therefore, some approaches of Brunett's model are not employable by us.

3.4 Our Proposed Model

Normally, people use trust to decide with whom to interact. A trust-based system allows agents to select their interaction peers dynamically during execution. In this research, the trust model we use is Recommendation-Based Trust, which has been frequently used for studying trust in multi-agent system [12, 16, 19, 20, 22, 29]. As we mentioned, considering trust as the sole factor for decision making may not be reasonable in many situations. Here, we propose that the context of the decision should also be considered and that risk (as an important contextual factor) should be taken into account [9, 17, 18, 22]. In this research, risk is modeled as a continuous measure, which results in having agents with a range of risk behavior rather than just three distinct types (Risk Seeking, Risk Neutral and Risk Averse). We also introduce the concept of adaptive risk, which gives the agents the opportunity to change their risk behavior based on their experiences. Agents in our model adopt a strategy

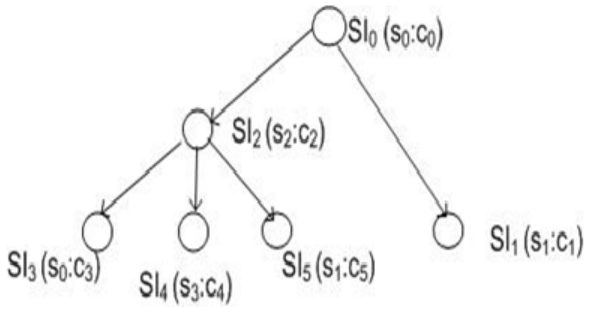

Figure 3.1: Structure of the task tree.

to affect the way they place tasks in their working queue. Finally, we propose an algorithm for decision making in a self-adaptive society with a tree structure task domain which utilizes trust and adaptive risk. Our self-adaptive society allows agents to evaluate their cooperation peers and change their organizational link with them in a decentralized way. With this mechanism of decision making, we aim to optimize total utility and number of tasks completed, and minimize time wasted in unprofitable tasks, idle time and overhead.

3.4.1 Task Domain

In this model, a task is represented as a dynamically incoming stream of multiple services. These services are considered as nodes (subtasks) of the tree. Each service instance (SI) needs a known skill and computation time. There is sequential dependency between subtasks, which means that the parent SI must be executed before its child SI. When all of the nodes in the tree are executed, the task is complete [30]. Figure 3.1 shows the tree structure of a task. There is a deadline and specified amount of utility associated with a complete task. The utility of a task is the amount of reward the agent gets for investing resources on this subtask and finishing it before its deadline. Since multiple agents cooperate in executing a task, agents get reward for the parts they execute. Therefore total utility of the task is divided between nodes. Reward associated with each node depends on the amount of computation the node needs for execution and the number of nodes sequentially dependant to the execution of this node. Utilizing the mentioned mechanism of dividing reward, nodes with more children get slightly higher reward in comparison with the same

node with fewer children (as explained before, children of a node must execute after the execution of their parents). This policy increases the agents' incentive for giving higher priority to the subtasks with more dependants and minimizes waiting time of the dependants of a special node. Details of the prioritization mechanism are explained in section 3.4.5.

If a subtask is completed after the deadline, the agent does not get a reward and the resources spent on the subtask have been wasted. To motivate progress, the reward of the task decreases if the finishing time exceeds what is expected for the task, known as *expected time*. Equation 3.1 shows the relationship between utility and time. Here t stands for time.

$$earnedU_{task} = AssignedUtility_{task} - (t_{task}^{taken} - t_{task}^{expected}) \tag{3.1}$$

Note that expected time differs from deadline. Expected time is the average estimated time for finishing the task. If the agent cannot finish the task by the expected time, the only punishment is decreasing its expected reward. This mechanism aims to reward a sooner finishing time, which is better for the system. Deadline is the last chance for the agent to accomplish the task and get the reward.

3.4.2 Self-Adaptive Agent Organization

Agent organization regulates the interaction between agents by defining various types of relationship links. These links specify the type of relationship between agents. There are four levels of relationship between agents listed in order of importance: (a) superior (obligation to satisfy), (b) subordinate (preferred delegation), (c) peer (low frequency of interaction) and (d) acquaintance (knowing existence but having no interaction). Acquaintances are strangers to each other, peers have equal authority and the superior-subordinate relationship depicts the greater authority of the superior over the subordinate. The type of relationship between agents specifies the amount of information they have about each other and shows the preferences of agents in the task-passing mechanism. Figure 3.2 shows an example of the organizational structure of agents.

Agents have a set of skills and known computational capacity needed for executing

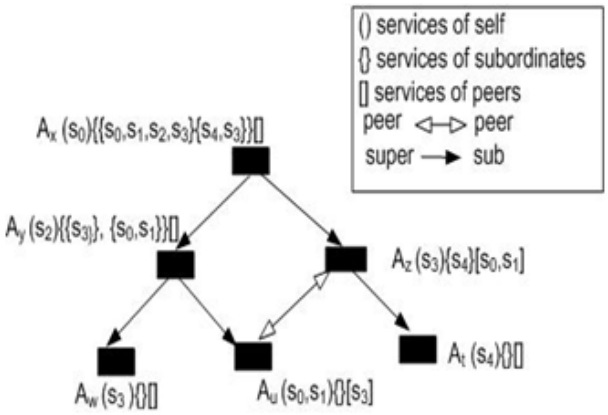

Figure 3.2: Example organizational structure.

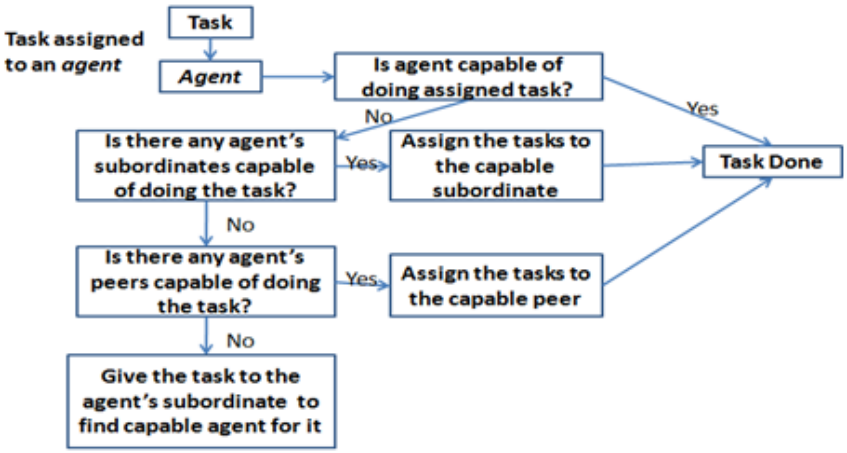

Figure 3.3: Task passing mechanism.

tasks and management computations. Since finishing a task requires cooperation between multiple agents, a task-passing mechanism allocates tasks to different neighbors. Based on its relationship with a neighbor, $agent_x$ prefers to allocate tasks first to its subordinates, then to peers and finally to its acquaintances. Figure 3.3 demonstrates this task passing mechanism.

When an agent is consistently looking for another agent with a specific skill, it is motivated to reorganize to form a direct relationship with an agent providing that service. This process is called adaptation. This process seeks to improve the profit of the system.

Agents can adapt locally, though based on local adaptation by the agents, the method leads to the benefit of the organization as a whole. The adaptation process consists of two main parts, named meta-reasoning and reorganization. Meta-reasoning determines which agents should be selected among an agent's neighbors, and reorganization evaluates and changes the organizational link with selected agents. This adaptation process aims to promote the relation type of helpful agents (based on history) and demote the relationship of agents who have shown less collaborative contribution in the neighborhood. Meta-Reasoning evaluates neighbors using the Satisfaction Measure. Each agent is satisfied with a relation when the corresponding agent is able to service most of the agent's requests; the stronger the neighborhood in terms of servicing requests, the more satisfied the agent is, as shown in Equation 3.2.

$$satisfaction = \frac{\sum_{i=t-h}^{t-1} numProvidedRequests}{\sum_{i=t-h}^{t-1} numRequsts} \tag{3.2}$$

When an agent accepts a subtask request, it means that it has the potential to accomplish that subtask. In this equation, i stands for iteration and t indicates current iteration. Variable h is in the range of $[1, t-1]$ and shows the amount of history that should be considered. In this approach, $agent_x$ ranks its neighbors based on the satisfaction measure. Then it selects some of its least satisfactory subordinates and peers along with some strong acquaintances for the reorganization process.

Reorganization enables an agent to change its relations with some of its neighbors, which are identified in the Meta-Reasoning step. For each $agent_y$ in this list, $agent_x$ takes the best action among possible actions based on the current relationship and a measure computed from evaluation functions. Figure 3.4 demonstrates possible actions for two agents based on their current relation.

However, creating a new relation brings an amount of load to the system; it gives the system the opportunity to earn utility via this new relation. In the case of removing a relation, the system gets rid of the load of that relation while not earning utility any more. Evaluation functions estimate the amount of utility and load associated with changing

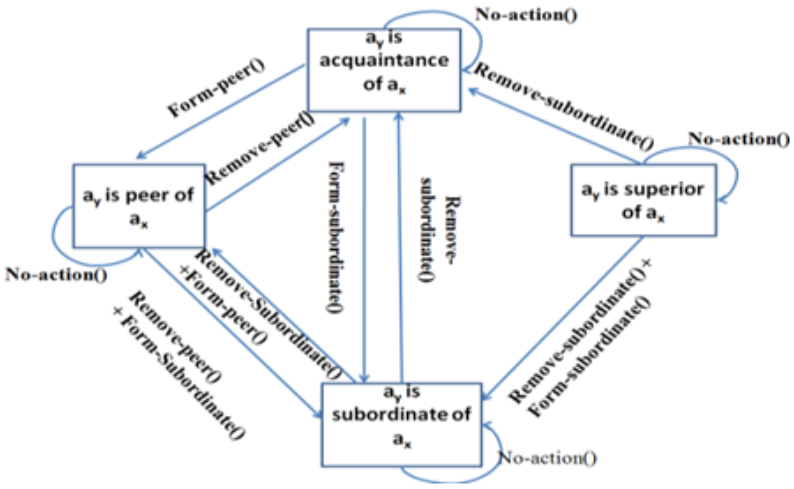

Figure 3.4: Diagram of possible actions.

a current relation R_c (between $agent_x$ and $agent_y$) to new relation R_n. Loads consist of management load, communication load and load of changing the relation, and are estimated based on history. Utility is estimated based on past experiences with the type of relation R_n and the effectiveness (in term of earned utility) of $agent_y$. For example, if $agent_x$ wants to make a peer relation with $agent_y$, it calculates the average earned utility of its peers so far, along with the average utility earned from $agent_y$. This means that the experience of $agent_x$ with that type of relation (R_n) and the history of functionality of $agent_y$ affect the decision [30].

3.4.3 Recommendation-Based Trust

Consider the case in which $agent_x$ needs to build trust in $agent_z$. To evaluate the amount of trust to extend, $agent_x$ needs to use experiences. A requestor agent seeks bids from possible provider agents, and contracts with a selected agent to complete a task. This experience between the requestor and the selected agent allows the requestor to evaluate the selected agent based on how well it accomplishes its promise. Experiences that are related can be direct or from third parties. Although direct experience has a stronger effect on the process of building trust, $agent_x$ may want to take into account the experiences provided by third parties. Recommendations of other agents help $agent_x$ to have better

judgment in building trust about $agent_z$. Suppose that in $agent_x$'s direct experience with $agent_z$, $agent_z$ was late or could not accomplish the given promise. Therefore, the bad experience of $agent_x$ affects its trust in $agent_z$. But when $agent_x$ considers third-party recommendations of $agent_z$, the effect of bad experience is adjusted. In addition, having other members' opinions helps $agent_x$ judge $agent_z$ from different points of view. The number of recommendations $agent_x$ needs to build trust in $agent_z$ depends on the risk behavior of $agent_x$. Risk-averse agents need more recommendations and more positive recommendations than risk-seeking ones. Recommenders are picked from among categories below:

- agents that most recently interacted with $agent_z$
- superiors of $agent_z$
- agents that have the most interaction with $agent_z$

Then, $agent_x$ combines the experiences for determine the amount of trust for $agent_z$.

3.4.4 Adaptive Risk

In our model, risk is a number in the range of $[0, 2]$, where 0 represents a fully risk-averse agent, 1 represents a risk-neutral agent, and 2 represents a risk-seeking agent. This allows a continuum of risk behavior rather than categorizing agents into only three options. This risk measure affects all the decisions each agent makes. Each agent gets a Risk Random Initial Value ($RRIV$) as its risk measure in the initialization phase. $RRIV$ is in the range of [0, 2]. We use a random initial value for risk to have agents with various risk behaviors in the model. Each agent has the opportunity to update its risk behavior based on its experiences (history) and current resources. The measure of earned utility and measure of loss are factors which $agent_x$ considers in updating its risk measure.

- Measure of Earned: This measure is calculated from Equation 3.3. and shows a percentage of earned utility.

$$Measure of Earned = (\frac{SumEarnedUtility}{SumPossibleUtility}) \tag{3.3}$$

The possible utility is the sum of all utility of the tasks accepted by $agent_x$. SE is considered a percentage of earned utility. Generally, agents with high earnings have a good social standing and a higher acceptability of risk.

- Measure of Loss: This measure is calculated from Equation 3.4. The possible utility is the sum of all the tasks offered to $agent_x$. Both Lost and Possible utility are based on accepted tasks by $agent_x$. Measure of Loss is considered a percentage of lost utility. Normally, agents who lose a lot try to be more cautious.

$$Measure of Loss = (\frac{SumLostUtility}{SumPossibleUtility}) \tag{3.4}$$

3.4.5 Reprioritization Algorithm using Agent's Strategy

Each agent can only accept one request per time-step. Therefore, agents store requests in a waiting queue. To estimate the expected finish time of the requested task, $agent_x$ needs to prioritize its queue and find the estimated time based on the anticipated task's place in the queue. Factors which affect prioritization include:

- Deadline (D): Passing the deadline results in getting zero reward and wasting expended resources.

- Utility (U): Expected utility of completing the task (incentive for doing the task).

- Strength of Relationship (SR): Strength of relationship with requestor. An agent could give higher priority to requests from superiors or from additional relationships that an agent wants to encourage (such as a peer or subordinate that is deemed valuable).

For special $task_i$, agents use (D (deadline), U (utility), SR (strength of relationship)) to specify priority for the task based on Equation 3.5.

$$MeasureOfPriority = \alpha_1 \times D + \alpha_2 \times U + \alpha_3 \times SR \tag{3.5}$$

Each agent has a different weighting scheme (α_1, α_2, α_3). We call this triplet the *strategy* of agent, which specifies the order of importance of each factor. Some agents may pay more attention to gaining utility than other factors. Some may worry about deadline (keeping their promises). Other agents focus mainly on satisfying their superiors to keep their relationship strong. We try various combinations of behaviors to see what is more profitable. Furthermore, agent strategy can evolve. The update policy is based on copying the strategy of one of the successful agents in their neighborhood. Various factors influence the measure of success which is utilized.

3.4.6 Decision-Making Algorithm

In task passing, $agent_x$ can pass the task to different levels of neighbors including subordinates, peers, superiors and acquaintances. In each level of neighbors, there may be agents who claim they can help (volunteer to do the task). Volunteers provide their suitability with three estimations, which include their best case, average case and worst case time estimation of accomplishing the task. To find the optimal agent among volunteers, $agent_x$ needs to consider factors like trust of volunteers, their estimation of finishing time, and its own risk measure. The main application of trust is identifying who to interact with. The risk measure and the volunteers' estimated times provide the context of the decision which $agent_x$ is going to make. Decision making involves these steps:

1. $Agent_x$ sends request for estimation of task completion to its subordinates

    - Subordinates who are capable of doing the task reply to $agent_x$, with their best, average and worst case estimation time of doing the task along with the probabilities of finishing the task in the estimated time. (The approach of making estimations is explained later of the current section).

- $Agent_x$ removes from consideration subordinates whose responses are dominated by other agents.
- $Agent_x$ updates trust value for the remaining subordinates based on the approach explained in 3.4.3.
- $Agent_x$ uses trust to evaluate their responses (to what extent $agent_x$ can accept their promises).
- $Agent_x$ maps the estimated times to utility. This mapping is based on Equation 3.1. Equation 3.1 simply says that an agent can get the whole defined utility of the task if it finishes it before a certain time. After that, the amount of utility the agent can earn decreases by passing time. Therefore, the different estimated times can be mapped to different utilities.
- $Agent_x$ finds the optimal agent based on *Expected Utility Theory* (details are explained later in the current section), which uses the agents' answers (converted to utility) and $agent_x$'s risk measure to pick the best option.

2. If there is no capable subordinate, $agent_x$ goes through the same steps as above for finding the optimal peer.
3. If there is no capable peer, $agent_x$ passes the task to one of its superiors. Then the superior is in charge of finding capable agent.
4. If $agent_x$'s superiors fail to find suitable agent, $agent_x$ tries to assign the task to the first volunteer acquaintance.
5. If $agent_x$ cannot find a capable agent after all of the following steps, it adds the task on the back of its queue.

Estimation Approach of Agents

For the estimations, each agent utilizes the reprioritization algorithm to find the anticipated place of the task in its queue. The anticipated place of the task in queue requires the agent to evaluate the amount of time the preceding tasks in the queue need. Furthermore,

there are always barriers to reach that time. Agents may finish tasks slower than what they anticipated due to unforeseen circumstances, such as another task with higher priority arriving and moving the current task down the agent's queue. Utilizing the occurrence history of these events and anticipated place of the requested task, each agent (queried agent) gives three estimations along with the probability of those estimations occurring. In all of the equations, the number of tasks in the queue is considered as n.

- Best case denotes the case where neither the preceding tasks in queue, nor the requested task are delayed. Equation 3.6 shows the best case time and probability estimations.

$$\begin{aligned} T(BestCase) &= \frac{\sum_{i=1}^{n} compNeeded_i}{AvgAgentCompPerIteration} \\ P(BestCase) &= (1 - p(lateness))^n \end{aligned} \tag{3.6}$$

- Average case estimation allows half of the tasks to be delayed, while the other half is not delayed.

$$\begin{aligned} T(AverageCase) &= \frac{\sum_{i=1}^{n} compNeeded_i}{AvgAgentCompPerIteration} + \frac{n}{2} \times T(lateness) \\ P(AverageCase) &= C_{\frac{n}{2}}^{n} \times (p(lateness)^{\frac{n}{2}}(1 - p(lateness))^{\frac{n}{2}}) \end{aligned} \tag{3.7}$$

- Worst case represents the case where all tasks, including the requested task, are delayed by these barriers.

$$\begin{aligned} T(WorstCase) &= \frac{\sum_{i=1}^{n} compNeeded_i}{AvgAgentCompPerIteration} + n \times T(lateness) \\ P(WorstCase) &= 1 - P(BestCase) - P(AverageCase) \end{aligned} \tag{3.8}$$

Expected Utility Theory

Agents use expected utility theory to help make decisions under uncertainty [9, 16–18, 20, 26]. This theory is a way to balance risk versus reward using a mathematical formula. In a case of having multiple choices, this theory allows agents to calculate the expected utility of each case and select the one with the highest expected utility. Risk measure affects the

way agents interpret the amount of utility [9,17,20]. Risk-seeking agents have an optimistic attitude toward the reward, while risk-averse agents are pessimistic and undervalue the actual reward of tasks. However, risk neutral agents are objective and see the actual reward of an action. In our model, we have a range of $[0, 2]$ for these three categories of agents where $[0, 1]$ represents risk-averse agents, 1 stands for risk-neutral agents and $[1, 2]$ depicts risk-seeking agents. In our model, an agent's interpretation of utility (IU) is calculated with Equation 3.9.

$$IU = Reward^{RiskMeasure} \tag{3.9}$$

When $agent_x$ wants to chose the optimal volunteer, it utilizes its risk measure and interprets the utility of each volunteer's estimations (best, average and worst case answers). Then based on Equation 3.10, $agent_x$ evaluates different choices and selects the agent with the highest expected utility according to the expected utility theory. In Equation 3.10, IU stands for Interpretation of utility, P stands for probability and $E(C)$ shows the expected utility of each choice.

$$E(c) = IU_{Best} \times P_{Best} + IU_{Average} \times P_{Average} + IU_{Worst} \times P_{Average} \tag{3.10}$$

3.5 Experiments and Results

3.5.1 Experiment Settings

To investigate the efficiency of our model in different situations, we introduced three scenarios, each imposing different levels of workload to the system. We compute the total capacity of the system as the maximum amount of work the system can accomplish when using the full capacity of all agents. Using this computed capacity, we define the low workload as 80% of the system capacity; and average and high workload as 100% and 120% of the system capacity respectively. For each scenario, we repeat 100 simulations of system, each performing for 1000 iterations. The results are averaged over the number of runs.

3.5.2 Profit Comparison

We evaluate the effectiveness of proposed model based on the performance of the organization measured as the system profit. System profit for each iteration is computed as the summation of the profits of all individual agents. We use Equation 3.11 to calculate the profit of agents per iteration using the amount of earned utility and the total cost of that iteration. Each agent can calculate its earned utility based on completed tasks using Equation 3.1.

$$Profit = UtilityEarned - TotalCost \tag{3.11}$$

$$TotalCost = ReorganizationCost + CommunicationCost \tag{3.12}$$

As Equation 3.12 shows, the total cost in the system includes reorganization cost and communication cost. Reorganization cost is used for evaluating and management of relationships. The process of assigning a task to the agents and trust building requires sending and receiving messages to/from the other agents. Therefore, these processes also require inter-agent communication, which adds some cost to the total cost of the organization.

Our first experiment compares the profit of the system using two approaches. The trust-based approach uses the decision making algorithm explained in 3.4.6. The no trust approach utilizes the task delegation approach depicted in Figure 3.3. Both of the methods respect the hierarchy of the agent organization for delegating tasks; they delegate the tasks in the following order: subordinates, peers, superiors, and finally acquaintances. Figure 3.5 shows the percentage of possible profit of the system under different workloads.

As it can be seen, in all the three scenarios, trust-based approach increases the system profit. In the low workload, the profit of the model in both approaches is higher than the two other scenarios. The reason is that agents are not as busy as the other workloads; so the failure rate of tasks is low. In this workload, the system has the least failed task percentage and highest task completion rate compared to other workloads. In the high workload scenario, the profit ratio (computed as percent of possible profit) of the system in both approaches is the worst in comparison with other approaches because the amount of workload is beyond the capability of agents. Figures 3.6, 3.7, and 3.8 show the queue

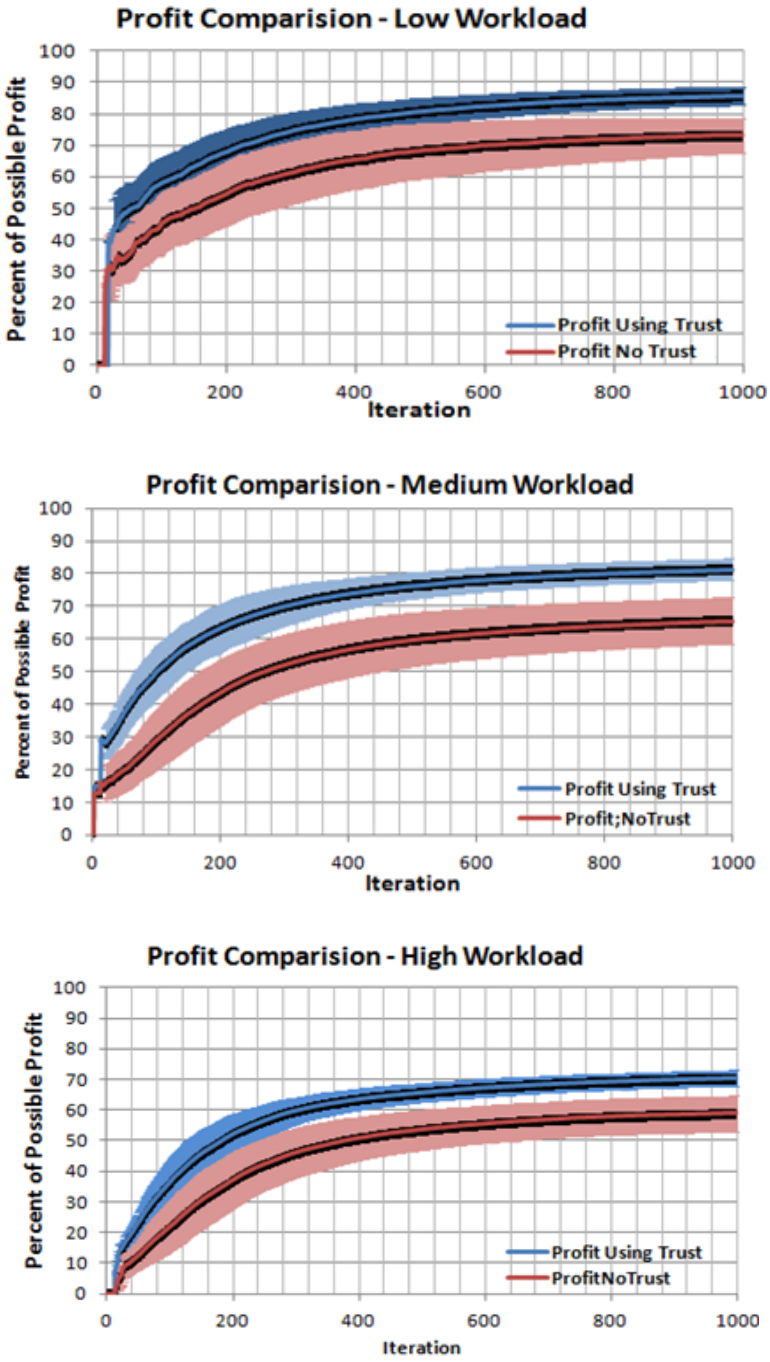

Figure 3.5: Profit comparison under three different workloads.

length of the system under the three workloads along with their standard deviation. Less standard deviation indicates even distribution of tasks among agents. In trust-based de-

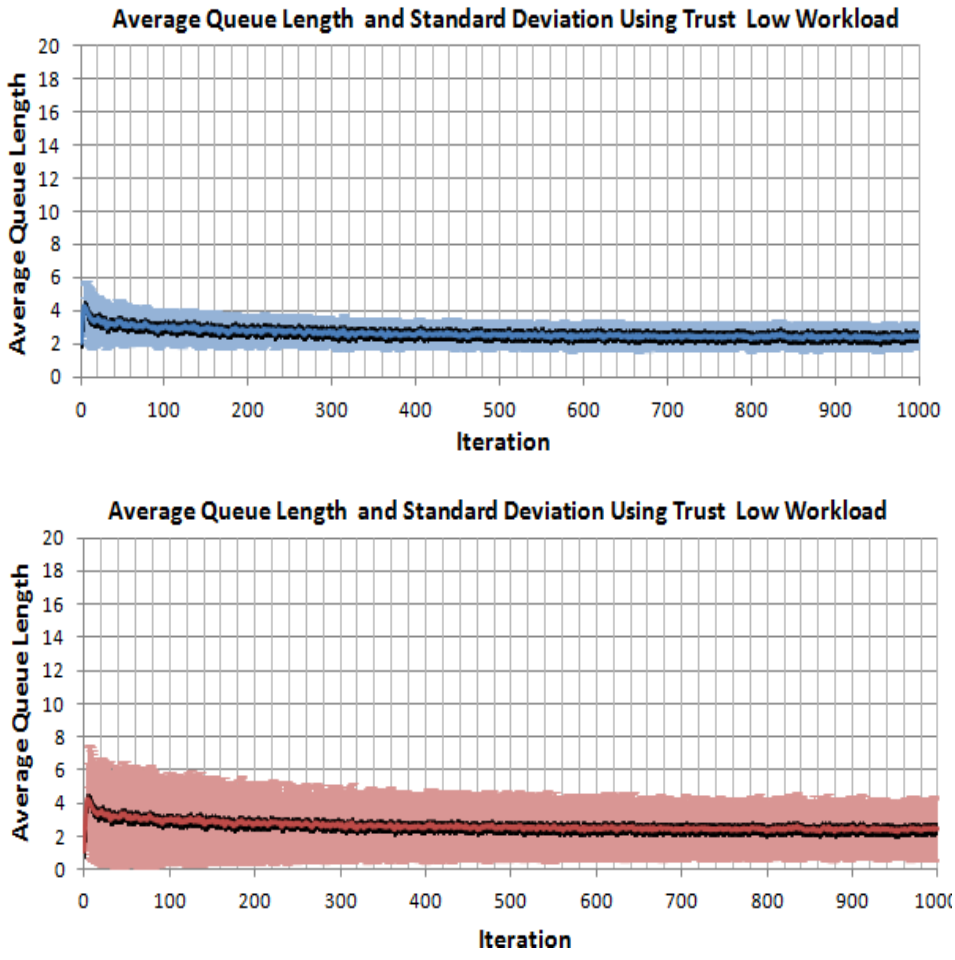

Figure 3.6: Queue length and standard deviation of trust-based and no-trust approaches in low workload.

cision making, volunteers for doing a task need to make estimation about finishing time of the proposed task. Since estimations are computed based on the queue length (details explained in 3.4.6), estimated time of agents with longer queue is higher than those with shorter queue. Therefore, busy agents (agents with longer queue) have less chance of winning the competition, and this fact results in even distribution of tasks among agents. It is clear to say that higher workload results in a longer queue length.

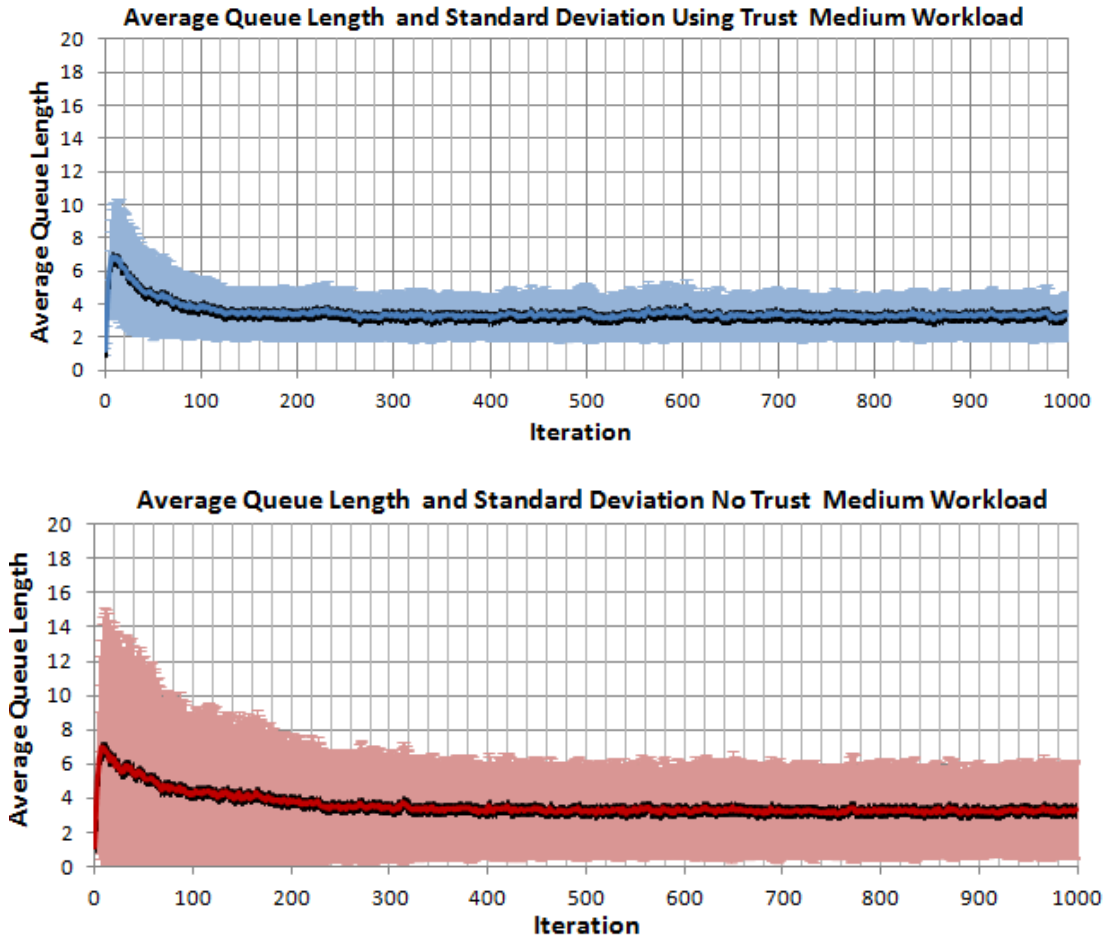

Figure 3.7: Queue length and standard deviation of trust-based and no-trust approaches in medium workload.

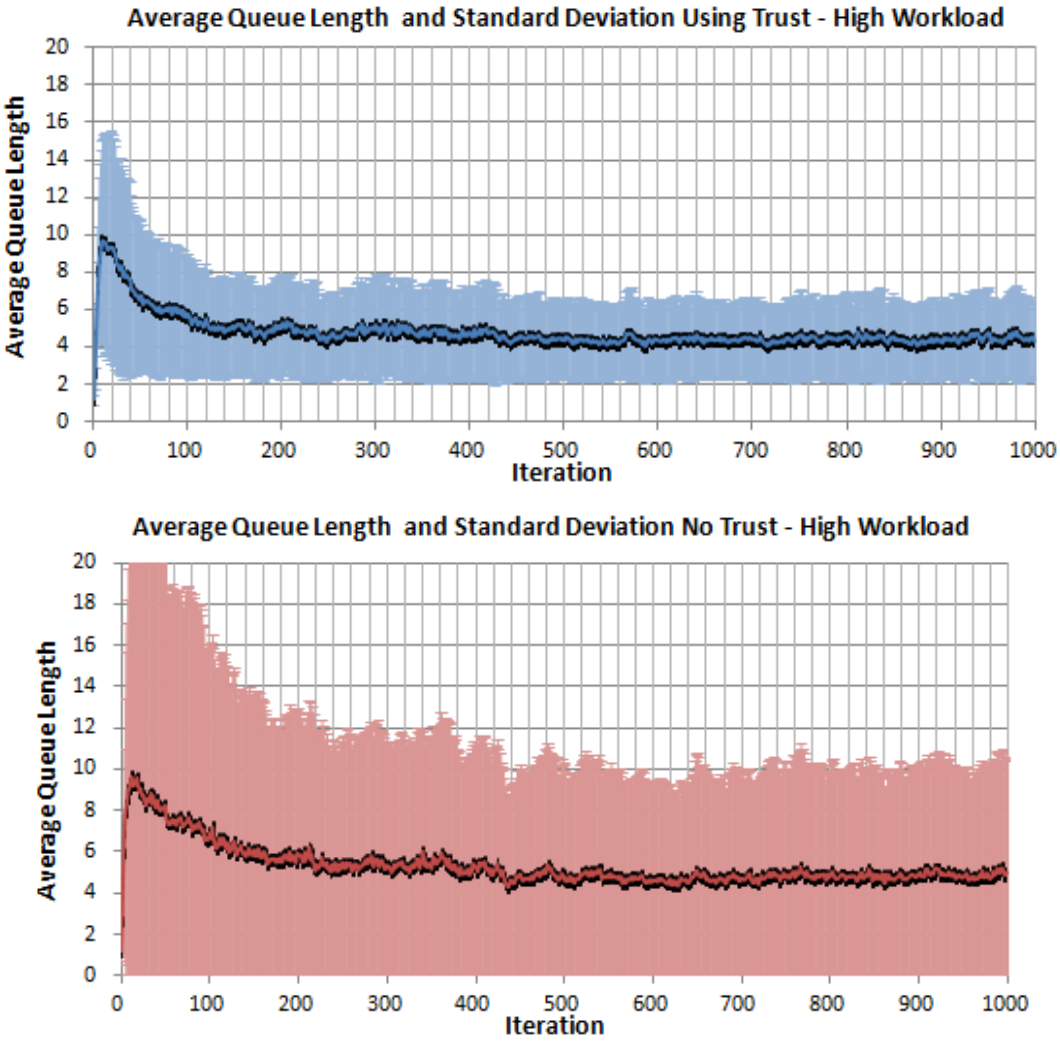

Figure 3.8: Queue length and standard deviation of trust-based and no-trust approaches in high workload.

3.5.3 Failed Task Comparison

Task failure minimization is one of the aims of using trust in evaluating cooperative peers and selecting the best one for a particular task. A smaller task failure percentage effectively increases the amount of earned utility of agents and better performance of the system. In this experiment, we measure the accumulated task failure of the system under the three workload scenarios. Figure 3.9 shows the behavior of the system in this experiment.

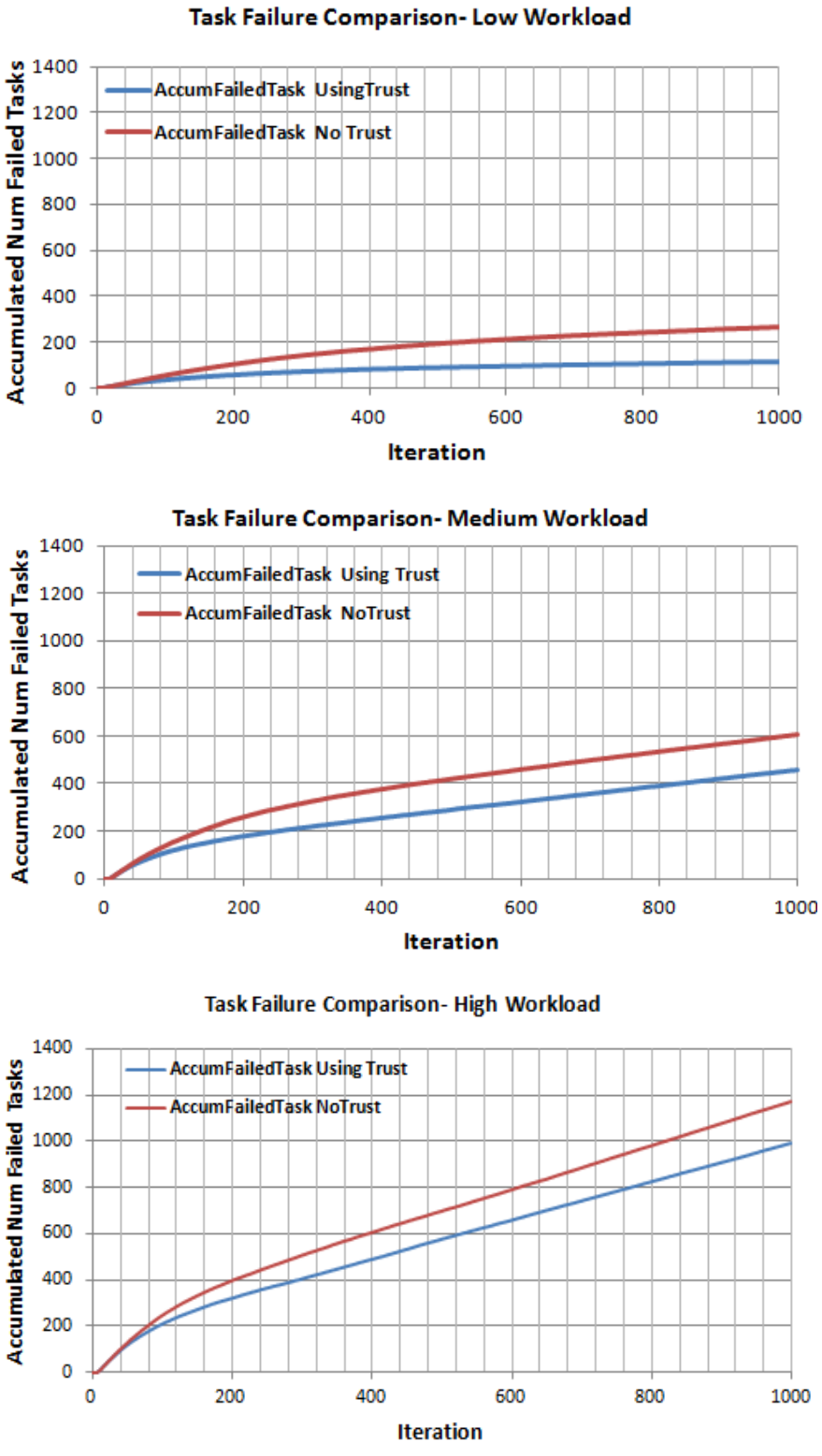

Figure 3.9: Comparison of task failure rate of trust-based and no-trust approaches in different workloads.

As it can be seen, the rate of task failure increases by raising the workload. For the high workload, we have a highest rate of task failure in both trust-based and no-trust approaches. As mentioned before, the reason is that the amount of workload is beyond the capability of agents. In other situations, there is a gap between trust-based and no-trust approaches. In all situations, trust-based approach helps in reducing the amount of task failure.

3.5.4 Contribution of various risk categories of agents

In this experiment, we study the contribution of agents in the whole profit of the system. Increasing profit requires earning more utility, and utility directly depends on task completion rate. The aim of this experiment is to see which risk category of agents is more successful in winning the competition of task delegation. Since our agents have a risk measure in the range of $[0, 2]$, we divide agents based on their risk measure of each iteration in three intervals $[0, 0.66]$, $[0.67, 1.32]$, and $[1.33, 2]$. The agents in the first interval is called risk averse category of agents, the agents in the second interval are risk neutral agents, and finally the rest are risk seeking agents. Figure 3.10 shows the contribution of the agents under three workload of the system. Since in this model agents' risk measure is adaptive, agents' contribution is categorized based on their momentary risk measure. For example in iteration 400, the red dot shows the contribution of all of the agents which their current (at the moment) risk measure is categorized in the neutral range. We have the same setting for risk seeking and risk neutral agents.

As it can be seen from the results of these experiments (Figure 3.10), risk neutral agents have a higher contribution in the whole profit of the system. It proves the idea that being too risk seeking or too risk averse is not as successful as being moderate. Moderate agents are more successful in getting tasks and have higher contribution in the profit of the system. Depending on the situation, each type of agent can have slightly more contribution. For example in low workload and high workload, risk averse agents have a slightly higher contribution, but in medium workload it seems both category of agents have almost equal contribution.

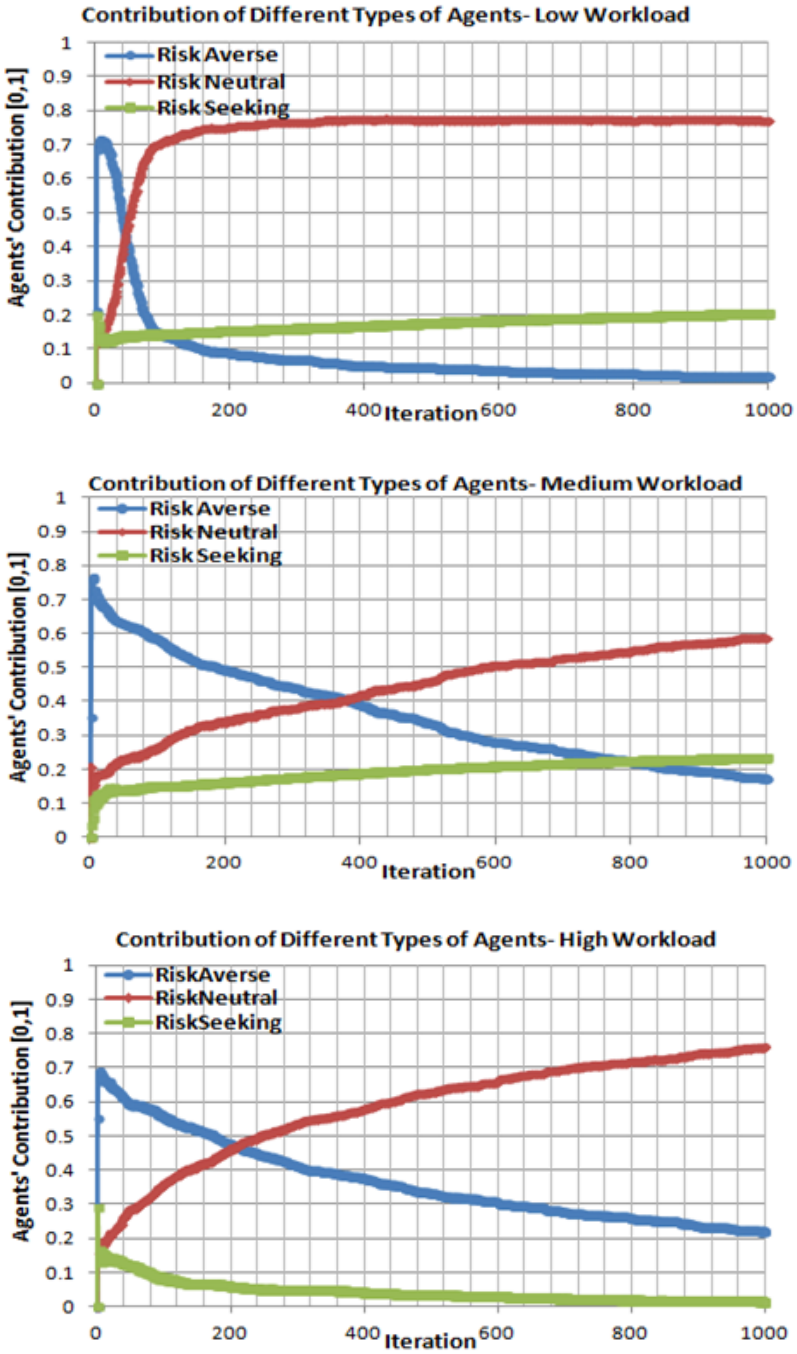

Figure 3.10: Contribution of risk seeking, risk neutral, and risk averse agents under different workloads.

3.5.5 Risk Changing Behavior

Risk measure affects agents' decision making based on expected utility theory explained

in 3.4.6. Adaptive risk is a proposed concept which lets the agents to update their risk behavior based on their past experiences. Normally agents who have been successful in the past have wider tolerance of risk and in contrast agent who lost a lot trying to be more cautious. This adaptive behavior lets the agents to evaluate their decisions history and try to make better decisions.

In this experiment we divide agents to three categories based on their initial risk measure. First interval is [0, 0.66] which indicates risk averse agents. Second interval is [0.66, 1.32] and agents in this interval are risk neutrals. The rest of the agents (in [1.32, 2] interval) are risk seeking agents. For each category of agents, we aim to see how agents change their risk measure over time for one simulation run utilizing change bars. Basically, change bar in $iteration = t$ represents the amount of change in risk measure of the corresponding category of agents in interval $[t-1, t]$. For example change bars in Figure 3.11, show the risk changing behavior of the agents which has been labeled as risk seeking initially over time. We have the same story for Figure 3.12 and Figure 3.13.

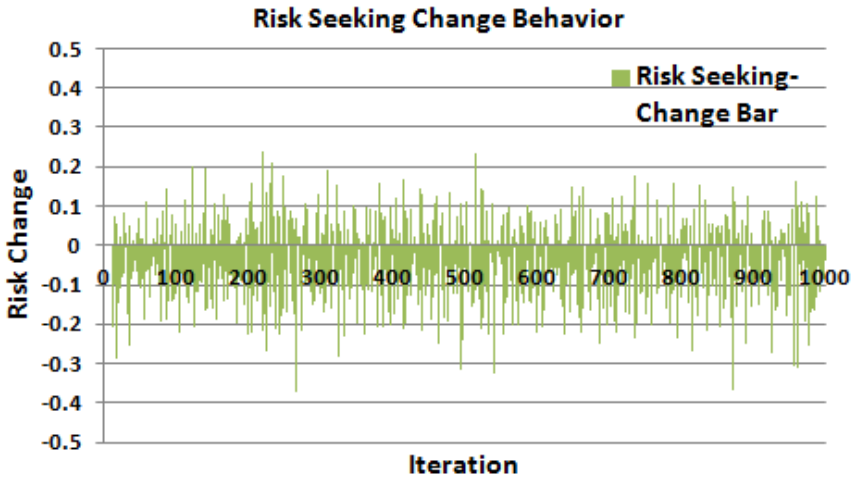

Figure 3.11: Risk changing behavior of risk seeking agents.

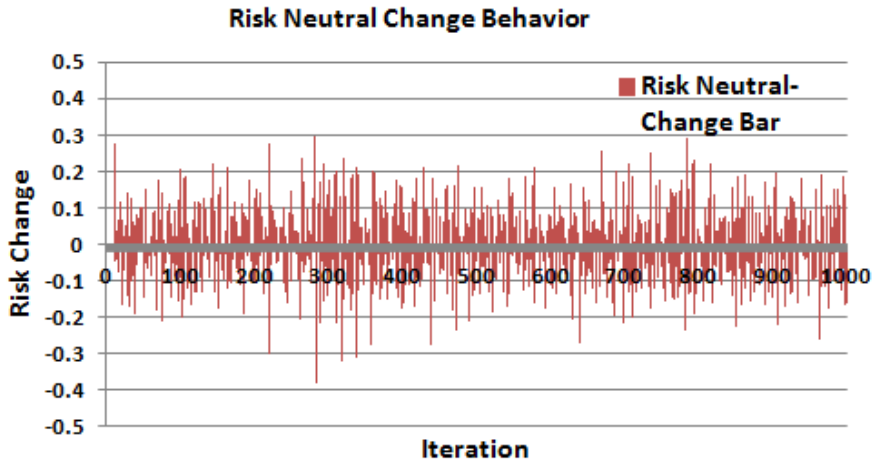

Figure 3.12: Risk changing behavior of risk neutral agents.

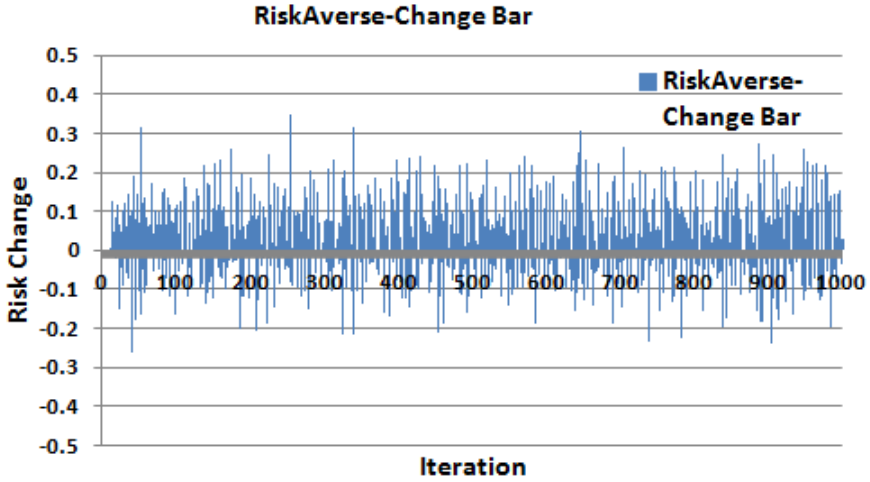

Figure 3.13: Risk changing behavior of risk averse agents.

Results show that agents which have been initially labeled as risk seeking, have higher accumulation of change bars toward decreasing risk measure, while for the agents initially labeled as risk averse is reverse. We can conclude that agents who started to be risk seeking intend to decrease their risk measure more often, while agent who started to be risk averse are more willing to increase their risk measure. For the agents initially labeled as risk neutral, we have almost equal accumulation of change bars in both sides. This

shows that although neutral agents experience change in their risk measure, they mostly stay in the neutral range. In the previous experiment 3.5.4, we saw that moderate agents (agents with the risk measure in the range of [0.66, 1.32]) had more contribution in the profit of the system. On the other hand previous experiment showed moderate agents are more successful than others in winning the competition of getting the tasks. Therefore, it is obvious to see that agents tended to change their risk behavior toward being more moderate.

3.5.6 Effect of Adaptive Risk

Risk directly affects decision making as explained in 3.4.6. Adaptive risk lets the agents change their risk behavior. Successful agents have wider risk tolerance, and agents who fail frequently need to be more cautious. Since functionality of agents is not constant over time, agents change their risk to help them make proper decisions. Agents are updating their risk measure based on the approach explained in 3.4.4. In order to see the effectiveness of adaptive risk, we tested our model under two different situations. The first situation gives the agents the opportunity of updating their risk behavior in each iteration. In the second scenario, agents have the fixed level of risk in whole experiment. In each of the experiments, we test the framework under three different workloads (high, medium, and low). Figure 3.14 summarizes the results of the experiments. Results demonstrate the usefulness of adaptive risk in reducing task failure, better task completion rate and higher profit.

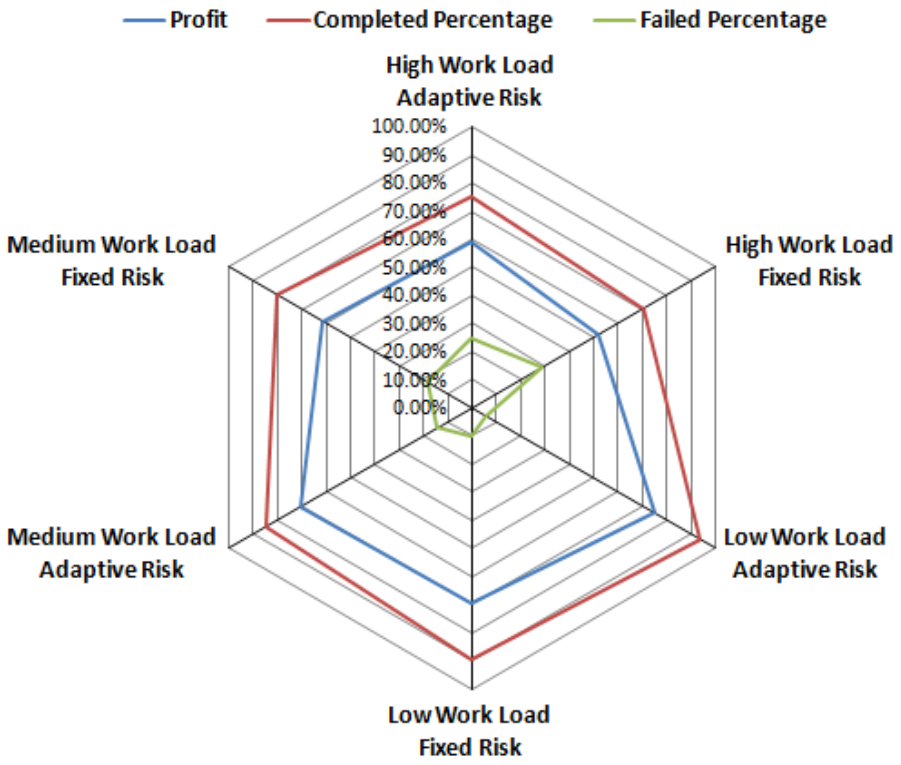

Figure 3.14: Effect of adaptive risk in reducing task failure, better task completion rate and higher profit.

3.6 Conclusion and Future Work

In this research, we proposed a decision making approach which gives the agents the opportunity of evaluating their cooperation peers utilizing recommendation-based trust and adaptive risk in a self adaptive society. The self-adaptive policy allowed the agents to evaluate the effectiveness of their neighbors based on the history and promote or demote the relationship with them in order to keep the most useful cooperative neighbors around themselves. Each agent builds trust about another agent using the recommendation of agents who had experience with the target. The experiences are direct and from third parties for having a variation of opinions in order to build more accurate trust estimation. Since perception of trust is not similar for all agents, context of the decision must be taken into account. Uncertainty as one of the main contextual factors plays an important role in decision making. In this model we proposed adaptive risk concept. This concept firstly

gives the agents a range of risk behavior rather than dividing them in three categories as risk seeking, risk neutral and risk averse. Secondly it gives the agents the opportunity of updating their risk measures based on their resources and experiences. For satisfying heterogeneity in the society, agents have different attitude toward reprioritizing their work queue called agent's strategy. This strategy mainly defines the agent's preferences over factors like earning utility, obligating to the tasks' deadline and respecting the strength of the relation with the requestor of agents. Besides reprioritization strategy of agents are adaptive. Each agent has the opportunity of copying the strategy of successful agents in its neighborhood. We tested our proposed method under three different workloads of the system: low, average, and high workloads. Average refers to the workload which is equal to the capacity of the system (total capability of agents). High workload is beyond the agents' capability, and low workload is less than system's capacity. In all the experiment, trust-based decision making approach shows higher profit, less task failure, higher completion rate of tasks, and even distribution of tasks among agents.

Possible future work includes using various reliability mechanisms to check the accuracy of provided information for recommendation-based trust. There are many sources of false data like correlated evidence and biased/extreme agents. Correlated evidence happens when multiple agents have a single experience and use that in making opinion about trustee. The receiver may consider them as different experiences, but actually they are the all the same. A possible solution is the clustering of agents which are somehow related to each other. Therefore two agents from one cluster probably share a common opinion about an experience. Biased agents are another main source of false data. Normally when there is a high chance of false data, there would be a big variance between provided opinions. There are some suggested approaches for handling these situations like building reliability factor of recommenders and gathering more and recent opinions. More opinions help on reducing the variance of data. Recent data seems to be more accurate than old data especially in dynamic societies. Agents should not rely on the recommendations which are basically reflect old experiences. Another possible future work invloves giving incentive for recommenders to

have mutual benefit between trustor and recommender. This mechanism increases the incentive of recommenders to cooperate in order to get reward and can reduce the possibility of biased or extreme opinions because of involving recommenders in delegation mechanism.

CHAPTER 4

CONCLUSIONS

The first part of this research deals with implementing self-adaptation in agent organization. We demonstrate a robust, decentralized approach for structural adaptation in problem solving agent organizations. Our adaptation method is based on the agents forging and dissolving relations with other agents. Agents use the history of past iterations as a measure of evaluation. We proposed a new method of adaptation which is called Selective-Adaptation. This method consists of two main parts which are meta-reasoning and reorganization. In the meta-reasoning, agents at each iteration select some of their neighbors for reorganization based on different approaches namely 1) Fixed Approach, 2) Need-Based Approach, 3) Performance-Based Approach, or 4) Satisfaction-Based Approach. After selecting neighbors, each agent evaluates all of the possible actions and tries to promote or demote the type of relationship between itself and the target agent. Since one of the purposes of self-adaptation is to improve robustness against failure, we test our model with two types of shocks to see the effectiveness of our self-adaptive system under unexpected circumstances. This method can successfully handle unexpected shocks to the system.

Possible future work include restricting agents' resources like memory and network bandwidth. Agents should have a limited amount of memory which is used for keeping information about others. In addition, we use network bandwidth for our communication purposes but we did not limit it. Another possible future work is changing this abstract model's task domain to real world problems like path finding or environment exploration. Dealing with real world problems needs agents with more realistic skills. Testing the adaptability and problem solving capability of this model under mentioned circumstances is considered as an ideal future work of this model.

In the second part of the research, we proposed a decision making approach which

gives the agents the opportunity of evaluating their cooperation peers utilizing recommendation based trust and adaptive risk in a self adaptive society. Self adaptive society let the agents to evaluate the effectiveness of their neighbors based on the history and promote or demote relationship with them in order to keep the most useful cooperative neighbors around themselves. Each agent builds trust about another agent using the recommendation of agents who had experience with the target. The experiences are direct and from third parties for having a variation of opinions in order to build more accurate trust estimation. Since perception of trust is not similar for all agents, context of the decision must be taken into account. Uncertainty as one of the main contextual factors plays an important role in decision making. In this model, we proposed adaptive risk concept. This concept firstly gives the agents a range of risk behavior rather than dividing them in three categories as risk seeking, risk neutral and risk averse. Secondly it gives the agents the opportunity of updating their risk measures based on their resources and experiences. For satisfying heterogeneity in the society, agents have different attitude toward reprioritizing their work queue called agent's strategy. This strategy mainly defines the agent's preferences over factors like earning utility, obligating to the tasks' deadline and respecting the strength of the relation with the requestor of agents. Besides strategy of agents are adaptive. Each agent has the opportunity of copying the strategy of successful agents in its neighborhood.

We tested our proposed method under three different workloads of the system: low, average, and high workloads. Average refers to the workload which is equal to the capacity of the system (total capability of agents). High workload is beyond the agents' capability, and low workload is less than system's capacity. In all the experiment, trust based decision making approach shows higher profit, less task failure, higher completion rate of tasks, and even distribution of tasks among agents.

Another possible future work is using various reliability mechanisms to check the accuracy of provided information for recommendation-based trust. There are many sources of false data like correlated evidence and biased/extreme agents. Correlated evidence happens when multiple agents have a single experience and use that in making opinion about trustee.

The receiver may consider them as different experiences, but actually they are the all the same. A possible solution is the clustering of agents which are somehow related to each other. Therefore two agents from one cluster probably share a common opinion about an experience. Biased agents are another main source of false data. Normally when there is a high chance of false data, there would be a big variance between provided opinions. There are some suggested approaches for handling these situations like building reliability factor of recommenders and gathering more and recent opinions. More opinions help on reducing the variance of data. Recent data seems to be more accurate than old data especially in dynamic societies. Agents should not confide on the recommendations which are basically reflect old experiences. Another possible future work is giving incentive for recommenders to have mutual benefit between trustor and recommender. This mechanism increases the incentive of recommender to cooperate in order to get reward. Considering mutual benefit can reduce the possibility of biased or extreme opinions because it involves recommenders in delegation mechanism.

REFERENCES

[1] E. Bou *et al.*, "Towards self-configuration in autonomic electronic institutions," in *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, ser. Lecture Notes in Computer Science, P. Noriega *et al.*, Eds. Berlin Heidelberg: Springer, 2007, vol. 4386, pp. 229–244. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74459-7_15

[2] M. Woolridge and M. J. Wooldridge, *Introduction to multi-agent systems.* New York, NY, USA: John Wiley & Sons, Inc., 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=559667

[3] J. M. Alberola *et al.*, "Multidimensional adaptation in MAS organizations," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 622–633, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1109/tsmcb.2012.2213592

[4] R. Kota *et al.*, "Decentralized approaches for self-adaptation in agent organizations," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 1, May 2012. [Online]. Available: http://dx.doi.org/10.1145/2168260.2168261

[5] M. Hoogendoorn, "Adaptation of organizational models for multi-agent systems based on max flow networks," in *Proc. of the Twentieth International Joint Conference on Artificial Intelligence, AAAI Press*, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.2125

[6] R. Kota *et al.*, "Self-organising agent organisations," in *Proc. 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multi-agent

Systems, 2009, pp. 797–804. [Online]. Available: http://portal.acm.org/citation.cfm?id=1558122

[7] W. Zheng-guang and L. Xiao-hui, "A graph based simulation of reorganization in multi-agent systems," in *International Conference on Intelligent Agent Technology, 2006. IEEE/WIC/ACM.* IEEE, Dec. 2006, pp. 129–132. [Online]. Available: http://dx.doi.org/10.1109/iat.2006.9

[8] L. Rasmusson and S. Jansson, "Simulated social control for secure Internet commerce," in *Proc. 1996 workshop on New security paradigms*, ser. NSPW '96. New York, NY, USA: ACM, 1996, pp. 18–25. [Online]. Available: http://dx.doi.org/10.1145/304851.304857

[9] C. Burnett, "Trust assessment and decision-making in dynamic multi-agent systems," Ph.D. dissertation. [Online]. Available: http://homepages.abdn.ac.uk/t.j.norman/pages/pdfs/BurnettThesis2011.pdf

[10] A. Koster *et al.*, "Opening the black box of trust: reasoning about trust models in a BDI agent," *Journal of Logic and Computation*, vol. 23, no. 1, pp. 25–58, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1093/logcom/exs003

[11] A. Koster and J. S. Mir, "Personalizing communication about trust," in *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 517–524. [Online]. Available: http://portal.acm.org/citation.cfm?id=2343650

[12] H. Yu *et al.*, "A survey of multi-agent trust management systems," *Access, IEEE*, vol. 1, pp. 35–50, 2013. [Online]. Available: http://dx.doi.org/10.1109/access.2013.2259892

[13] R. Falcone and C. Castelfranchi, "Trust and deception in virtual societies." Norwell, MA, USA: Kluwer Academic Publishers, 2001, ch. Social trust: a cognitive approach, pp. 55–90. [Online]. Available: http://portal.acm.org/citation.cfm?id=379157

[14] F. Cerutti *et al.*, "Context-dependent trust decisions with subjective logic," Sep. 2013. [Online]. Available: http://arxiv.org/abs/1309.4994

[15] S. D. Ramchurn *et al.*, "Trust in multi-agent systems," in *THE KNOWLEDGE ENGINEERING REVIEW*, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.2218

[16] C. Burnett *et al.*, "Supporting trust assessment and decision-making in coalitions," *IEEE Intelligent Systems*, vol. 12, no. 99, p. 1. [Online]. Available: http://dx.doi.org/10.1109/mis.2013.53

[17] M. Traub *et al.*, "Who goes there?: selecting a robot to reach a goal using social regret," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '11. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 91–98. [Online]. Available: http://portal.acm.org/citation.cfm?id=2030484

[18] L. D. Molm *et al.*, "Risk and trust in social exchange: an experimental test of a classical proposition." [Online]. Available: http://www.uvm.edu/~pdodds/files/papers/others/everything/molm2000a.pdf

[19] M. P. Singh, "Trust as dependence: a logical approach," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '11. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 863–870. [Online]. Available: http://portal.acm.org/citation.cfm?id=2031741

[20] C. Burnett and N. Oren, "Sub-delegation and trust," in *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 1359–1360. [Online]. Available: http://portal.acm.org/citation.cfm?id=2344005

[21] A. Ghaffarizadeh and V. H. Allan, "History based coalition formation in hedonic context using trust," *Int. J. of Artificial Intelligence & Applications*, vol. 4, no. 4, pp. 1–8, Aug. 2013. [Online]. Available: http://dx.doi.org/10.5121/ijaia.2013.4401

[22] P. Coleman *et al.*, "Towards human decision-making in multi-agent systems." [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.120.7522

[23] J. F. Hubner *et al.*, "Using the MOISE+ for a cooperative framework of MAS reorganisation," vol. 3171, pp. 506–515, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.1883

[24] L. Barton and V. H. Allan, "Adapting to changing resource requirements for coalition formation in self-Organized social networks," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, ser. WI-IAT '08, vol. 2. Washington, DC, USA: IEEE, Dec. 2008, pp. 282–285. [Online]. Available: http://dx.doi.org/10.1109/wiiat.2008.121

[25] J. C. Miralles *et al.*, "Multi-agent system adaptation in a peer-to-peer scenario," in *Proc. 2009 ACM Symposium on Applied Computing*, ser. SAC '09. New York, NY, USA: ACM, 2009, pp. 735–739. [Online]. Available: http://dx.doi.org/10.1145/1529282.1529437

[26] C. Burnett *et al.*, "Trust decision-making in multi-agent systems," in *Proc. Twenty-Second international joint conference on Artificial Intelligence - Volume One*, ser. IJCAI'11, Barcelona, Catalonia, Spain. AAAI Press, 2011, pp. 115–120. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/ijcai11-031

[27] M. Venanzi *et al.*, "Trust-based fusion of untrustworthy information in crowdsourcing applications," in *Proc. 2013 international conference on Autonomous agents and multi-agent systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 829–836. [Online]. Available: http://portal.acm.org/citation.cfm?id=2485052

[28] T. Grandison and M. Sloman, "A survey of trust in internet applications," *Communications Surveys & Tutorials, IEEE*, vol. 3, no. 4, pp. 2–16, 2000. [Online]. Available: http://dx.doi.org/10.1109/comst.2000.5340804

[29] N. Griffiths, "A fuzzy approach to reasoning with trust, distrust and insufficient trust," in *Cooperative Information Agents X*, ser. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2006, vol. 4149, pp. 360–374. [Online]. Available: http://dx.doi.org/10.1007/11839354_26

[30] K. Ahmadi and V. H. Allan, "Efficient self adapting agent organizations," in *Proc. 5th International Conference on Agents and Artificial Intelligence.* Barcelona, Spain. SciTePress - Science and and Technology Publications, 2013, pp. 294–303. [Online]. Available: http://dx.doi.org/10.5220/0004261902940303

APPENDIX

CHAPTER A

Reprint Permission

Gmail by Google

Kamilia Ahmadi <kamelia.ahmadi@gmail.com>

Request for Copyright of My Paper in My Own Thesis

Joaquim Filipe <jfilipe@insticc.org> Tue, Apr 15, 2014 at 3:56 PM
To: Kamilia Ahmadi <kamelia.ahmadi@gmail.com>

Dear Kamilia,

If you clearly indicate the reference of your pape rand a link to the conference website, you are authorized to put the paper as part of your thesis.

Kind regards

Joaquim Filipe

From: Kamilia Ahmadi [mailto:kamelia.ahmadi@gmail.com]
Sent: Tuesday, April 15, 2014 17:52
To: secretariat@scitepress.org
Subject: Request for Copyright of My Paper in My Own Thesis

Dear Sir/Madam,

I am the first author of paper named "Efficient Self Adapting Agent Organizations" which was presented in ICCART 2013. I want to put this paper as part of my thesis; university asked me for copyright of the publisher.

In the FAQ of the ICCART website stated that copyright for putting the work on the own thesis is normally given by the request of the authors. I was wondering if I could get the copyright very soon since I am on tight schedule for graduation.

If you need further information please let me know,

Thanks,

Kamilia

Information:

My Name (First Author): Kamilia Ahmadi

Paper Name: Efficient Self Adapting Agent Organizations

Event: International Conference on Agents and Artificial Intelligence (ICAART 2013)

Refrence in Harvard Style:

Ahmadi K. and H. Allan V. (2013). **Efficient Self Adapting Agent Organizations**. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, pages 294-303. DOI: 10.5220/0004261902940303