Intention Scheduling for BDI Agent Systems

Shung-Bin Yan , Zu-Nien Lin, Hsun-Jen Hsu, and Feng-Jian Wang
Department of Computer Science and Information Engineering, National Chiao Tung University
Room 510, EC Building, 1001 Ta-Hsueh Road, Hsinchu City, Taiwan, ROC
Telephone: 886-3-571-2121 ext 54718 Fax: 886-3-5724176
Email: { sbyan,tnlin,hsuhj,fjwang}@csie.nctu.edu.tw

Abstract

BDI is one of the models in Agent Systems. Currently, most of BDI agents' researches are focused on the ability for agents to dynamically select plans to achieve a goal. There are fewer discussions on Intention Scheduling, the order in which the selected plans are executed. Agents must adapt to dynamical and unpredictable changes. Without a proper scheduling scheme, an agent may repeat unnecessary works, waste valuable resource or even fail the users' expect altogether. In this paper, we present an effective intention-scheduling scheme in BDI reasoning process.

Keywords: BDI agent, intention scheduling, time constraint, plan, efficiency.

1. Introduction

In general, a rational agent has the following properties: autonomy, social ability, reactivity and pro-activeness [1]. Among various agent architectures, BDI [2] model, where "Belief", "Desire", and "Intention", each represents a mental state in practical reasoning process, is a mature model and has been adopted by many academic and industrial applications.

During the development of systems with multiple agents of mobility and intelligence [3], we discovered that these agents adapt to dynamical and unpredictable changes. It is necessary to solve the problems of repeating unnecessary work, wasting valuable resource and even missing the users' expect altogether for the agents.

An intention-scheduling scheme might be applied to improve the performance of BDI agents by exploiting the positive interaction between input tasks and avoiding possible conflicts. Based on the exploitation, the agent saves the system's resources after doing a work but does not repeat similar works. With concern of temporal constraints, the agent does not miss the deadline. This paper presents an effective intention-scheduling scheme and some useful structures in BDI reasoning process, to make BDI agents more efficient and suitable for dynamic environment.

The rest of this paper is organized as follows. Section 2 surveys the current status of research regarding BDI agent theories and architectures, as well as related works on intention scheduling. Section 3 discusses our model on intention scheduling. Section 4 presents the implementation consideration. Section 5 shows a simulation of our system and discusses the contributions. Section 6 indicates a conclusion and our future works.

2. Background

2.1. BDI Agent Theory

The BDI (Belief-Desire-Intention) theory is based on the reasoning theory proposed by philosopher Michael Bratman [1]. Intention is the desire that an agent has committed to achieve [4]. As Cohen and Levesque [5] said, an agent has many desires which may not come true. A typical BDI agent has a set of Plans [6][7]. Intentions could also be treated as the plans for eventual execution. Rao and Georgeff [8] provided some logics for the BDI architecture. This model is effective, and applied in a number of works including air traffic control [9] and the handling of malfunctions on NASA's Space Shuttle [10]. Besides, Wooldridge and Jennings [11] worked on intelligent agent theory.

2.2. AgentSpeak(XL), TÆMS and DTC scheduler

AgentSpeak(XL) [15] provides PRS-like BDI agents which deal with the problem of intention scheduling with TÆMS[16] and DTC[17]. There is an overview of that approach to multi-agent systems in [18]. DTC (Design to Criteria) scheduler uses the information provided by TÆMS to generate a proper course of activities. The effects of duplicating the reasoning part, "from Desire to Intention", and long

Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)
0730-3157/05 $20.00 © 2005 **IEEE**

time for scheduling indicate that the duplication is not proper for these agents to be integrated with the current PRS-like agent architecture and blunt to the environmental changes.

Furthermore, TÆMS needs a lot of information long before the task is actually executed. While an agent acquires new kind of tasks from newly input plans, AgentSpeak(XL) doesn't answer how the TÆMS reflects these changes.

2.3. Dynamic Discovery of Goals

Padgham and Thangarajah worked on dynamical reasoning for the goals in BDI agent. Their results include representation and reasoning for goals [19], detecting resource conflicts [20], detecting similar goals [21], and interference between goals [22]. They created "Goal-Plan-Tree" structure to detect (positive or negative) interactions between goals, but did not take the importance of the goals and the time constrains into consideration.

The intentions can be treated in another way. In this paper, intensions are defined as committed goals. The above techniques will be applied to discuss the interactions between intentions, treated as factors in the whole scheduling process dynamically.

3. Intention Scheduling Concept

This section first discusses the major factors of intention scheduling, including utility, time constraints, interactions, degree of completeness and fairness. Then it introduces the *Intention Tree* for gathering these factors. Finally, our intention scheduling method is described.

3.1. Factors of Intention Scheduling

3.1.1. *I-utility*

Normally, the tasks of more importance need be given more time and higher priority to execute. JAM[7] uses "utility" to describe the importance of a goal and uses it as the only factor to select an intention. *I-utility* defined in the paper represents the importance of the goal for user. *I-utility* here is a real number or a simple function, and its value can be evaluated dynamically. *I-applicability* indicates the extents a plan can fulfill the goal. *I-applicability* can also be measured dynamically. *Priority* indicates the urgency of the intention. The *I-applicability* defined in this paper is computed from the *I-applicability of* associated plans, goal's *I-utility* and some factors to be discussed.

3.1.2. Time Constraint

To avoid missing the deadline, the priority of a task need be incremented when its deadline is approaching. Therefore, the agent must know the deadline of the goal, how long the intention costs on running, and how the priority is incremented.

Calculating the running time of an intention is a little bit complicated. There are two problems to be concerned at least. First, the associated plan may contain subgoals. The running time of a goal can not be calculated if the plan is not chosen. This problem is called *Undecided Plan Problem*. Besides the subgoals, each step might be run in a distinct time interval, according to different machine or network speeds. Here, an agent designer uses *Estimated Running Time, ERT,* to describe how many time units a plan may need. Together with the Intention Tree, *ERT* can help the agent to calculate the possible minimum and maximum running time of its intentions. And, a function, *Deadline Utility Function DUF,* can be used to compute the priority modification of the intention affected when its deadline is approaching. The function is defined in chapter 4 of [23].

3.1.3. Interaction

Executing an intention may modify the belief of the agent, and other intentions. The relationships between two intentions to help each other are called *positive* interactions. Those hindering each other (intention) are called *negative* interactions. An agent can schedule its intention to avoid conflicts or save system resources by applying the interaction information. The interaction relationships can also indicate the relationships of the tasks and their magnitude of mutual influence. However, the interactions might not be useful always. For example, it may not be strong enough to affect current priorities still

The interactions can be specified explicitly in Plan. But, some interactions not specified may appear. For example, during goal achievement, plans may have side effects affecting the belief. Some pre-conditions or context, which must be kept before or during execution, may also cause interactions between intentions. This paper does not discuss how to automatically discover interactions between intentions. It can be studied further.

3.1.4. Degree of completeness

An agent might lose its result, when one of its intentions is interrupted. For example, the cost to

interrupt an intention of 80% completeness with utility of 90 is much more than the cost to interrupt an intention of 20% completeness with utility of 100. Therefore, the *Degree of Completeness, DoC,* of the intention need to be considered when there are negative interactions between intentions.

The simplest way to indicate the DoC of an intention is to count how many lines of code in a plan has been executed for an intention. However, each line of code in a plan has a distinct contribution. A *progress label* (PL) is thus provided to show how much work has been done when reaching a position of code. The plan-decision problem exists, and an Intention Tree defined in the paper is used to compute the DoC.

3.1.5. Fairness

Besides efficiency, the fairness need be considered. The low-utility intentions may never get a chance to execute when high-utility ones keep executing. In order to avoid this starvation problem, the priority of an intention that has not executed for a long while need be raised. However, if the low-utility intention has some negative interactions with other intentions, raising the priority of this intention might cause other intention to fail and waste their work done. Therefore, fairness might be considered only for the intention which has no interaction with others.

3.2. Intention Tree

3.2.1. Undecided Plan Problem

Intention Tree*s* are mainly used to solve the plan-decision problem. Figure 1 shows an intention structure of JAM shopping agent during execution. The dash-line box in intention thread B represents the activities not taken yet. When the agent executes the "Moving to Site A" activity in intention thread A, it doesn't care whether intention thread B will have a "Moving to Site B" subgoal, because the "Buying shoes" goal has not chosen which plan to execute yet. The interaction does not appear until the agent decides to use "Buying Shoes' Plan 1." On the other hand, when this decision is being accomplished, the "Moving to Site A" part of actions might complete long time ago. The situation gets nastier when the "Moving to Site B" is deep down in intention-thread B.

Plan-decision problem also hinders other calculation such as the execution time and DoC of a plan. Our approach extends the concept of Intention Thread with Intention Tree to solve the plan-decision problem.

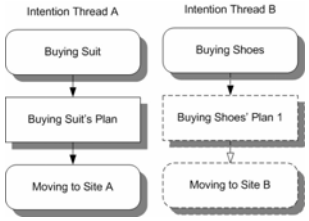

Figure 1 Shopping Agent's intention structure in the middle of execution

3.2.2. Structure of Intention Tree

The Intention Tree is applied to model the possible behaviors of an agent. Its nodes record not only static structure of the agent but also the runtime information. An Intention Tree mainly comprises two kinds of nodes, *goal-node*s and *plan-node*s. A goal-node contains the information including the goal's type, name, I-utility, deadline, and etc [23]. A plan-node contains the information including the plan's body, I-applicability, Estimated Running Time, and DcC. Inside a goal-node, there are a set of plan nodes able to achieve the goal, and each plan-node has its own subgoals. Figure 2 shows the Intention Tree of "Buying Shoes" Intention (intention thread B in previous example.)

Our initial construction for the information inside an Intention Tree is a recursive way from leaves till the root of the tree. The information is kept updated at running time, goals achievement, plan failure(s), or new plan addition. Thus, an agent can "foresee" what it might do without the actual execution.

Figure 3 indicates that the "Buying shoes" goal has not yet chosen which plan to execute, but the agent still knows that there is a "Moving to Site B" subgoal in intention-tree B. Consequently, the agent can discover the phenomenon: The Intention structure in Figure 3 does not contain Intention Thread but the Intention Tree.

The ability to foresee what an agent might do could also benefit the process of BDI's plan choice. For example, the agent could choose the plan that does not conflict with the current execution intention, or avoid the plan that cannot meet the goal's deadline definitely. Accordingly, the techniques to construct and update the Intention Tree described here can also be applied in other aspect of BDI agents.

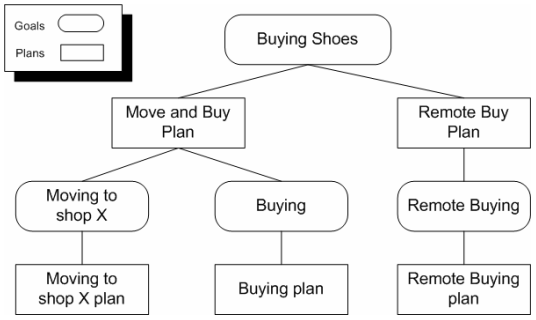

Figure 2 Intention Tree of the Buying Shoes Intention

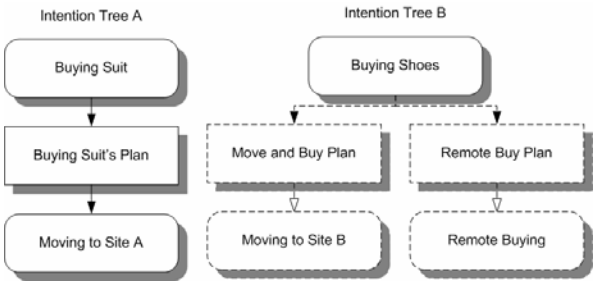

Figure 3 Shopping Agent's intention structure in the middle of execution

3.3. The Scheduling Process

This section presents our scheduling technique. Figure 4 shows the BDI agent's execution activities. The red rectangles are the activities related to newly intention-scheduling

After loading the plans, an agent constructs its Intention Tree. The block "Update the Intention Tree" beside the block "Execution Cycle" are in the same execution thread. It indicates that the Intention Tree always keeps updated and can be used in a procedure other than "Intention Selection".

The "Select Intention" is the process that chooses the intention to execute next. It contains four major steps.

1. The agent calculates each Intention Tree's *base priority*, which is defined as *I-utility* multiplies *I-applicability* in the top-level plan. (The top-level plan is the current plan applied to achieve the top-level goal.) Here, only the top-level goal is considered because its importance does not change the decision for the addition or deletion of subgoal. Subgoals' I-utility and their plans' I-applicability are considered when applying deadline or interaction.
2. The agent checks if the intention is approaching its deadlines. An Intention Tree may have one or more deadline because its subgoal may have deadlines. The agent adjusts the priorities of the Intention Tree according to each deadline's DUF.
3. The agent takes the interaction between intentions into account. This step works with a help of the following graph. Each node in this graph is an Intention Tree of the agent. There are three kinds of links, *Base*, *Interactions* and *Final* links. The base link from node A to B with value N means that A has higher base priority magnitude N than B. Interactions links indicate that there exist interactions between the Intention Trees. Their values are either specified by the user directly or computed based on a pre-specified function. If there is no interaction, the agent adjusts the base links of the node according to the time it waits for execution. The agent consolidates these links between the nodes to produce the final links.
4. The agent selects the node which has highest priority based on input final links. For example, an Intention Tree with no input final link has higher priority than others. When this kind of trees does not exist, the agent recursively deletes the final link of the smallest value until there is a node with no input final links. Then, the node is selected.

Figure 5 shows an example. Although the Intention Tree C has the lowest base priority, its strong interaction to Intention Tree A makes a final link from C to A. Because each node has at least one input final link, the agent deletes the lowest-value final link, which is the link between C and B. After the computation, there is no final link pointing to C. Then, C becomes the next intention for execution.

4. Design and Implementation Consideration

4.1. JAM Script constructs for Intention Scheduling

We first explain the constructs added for intention scheduling.

We modified JAM's goal format to add additional functions to compute deadline and running time. The deadline value specified with an expression is defined this goal. Optional DEADLINE_UTIL_FUNC is the function, computing how much the timing factor affects the intention's priority. More detail describes in [23].

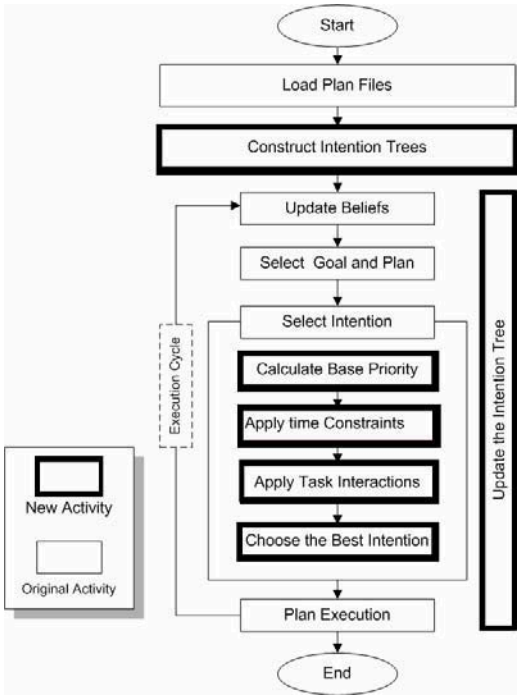

Figure 4 Agent Execution Activities

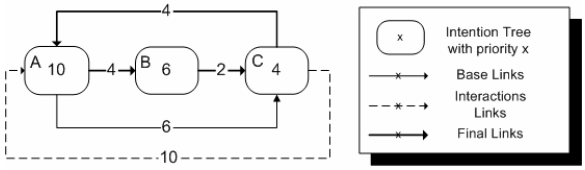

Figure 5 An example of applying interactions between Intention Three

We add these new fields in to JAM's plan: I-APPLICABILITY, ERT, FAIRFACTOR, and INTERACTION.

PL (progress label) facilitates the calculation of DoC and the running time of the plan, and ELT (estimated loop time), helps an agent to calculate the running time and DoC accordingly.

4.2. Construction of the Intention Tree

Figure 6 shows the Class Diagram of Intention Tree. The agent's Interpreter accesses Intention Tree from its root, which is the GoalNode representing the top-level goal. GoalNode has more than one PlanNode. The PlanNode is a tree, which consists of one or more plan constructs. Various actions, including subgoaling actions, are done associated with the plan constructs.

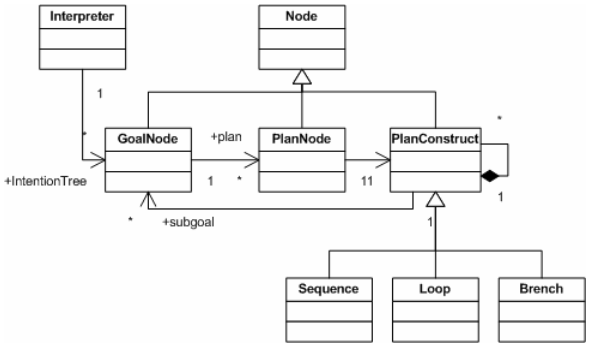

Figure 6 The Class Diagram of Intention Tree

When a top-level goal is issued, the corresponding Intention Tree is built. The interpreter generates a new GoalNode, and puts it into its Intention Structure. For a plan that can fulfill the goal, a PlanNode is generated and appended under the GoalNode. The PlanNode consists of a PlanConstructs tree, which are built according to the plan's body. Then, the PlanCostruct with subgoaling actions generates GoalNodes which are appended as its children. The construction continues recursively until there is no subgoaling anymore.

4.3. Information Propagation in Intention Tree

The information needed for computation in Intention Tree includes interaction, deadlines, and runtime information. The runtime state of the node in the Intention Tree shows the current status of the agent.

The initialization process is done by updating each leaf node of the Intention Tree, to gather the information for intention scheduling. The gathered information will be stored in the tree's root for future access. During the agent's execution, the information in the Intention Tree is mostly updated whenever actions are performed, goals are achieved, plans failed, or new plans are added.

There are at least three kinds of computation schemes used to update the Intention Tree. They are Sequence (for the sequence plan construct.), Branch (for the branch plan construct and the goal node) and Loop (for the loop plan construct and the recursive block).

4.4. The Execution Cycle and Intention Selection

In every cycle, the interpreter first updates the beliefs after observing the world. Second, the plans of the pending goals are checked against the current beliefs. Next, the interpreter selects an intention to execute from the Intention Structure. After the

execution, the Intention Tree is updated accordingly: If the action is to load other plans, every Intention Tree with the goals which can be achieved by these new plans is updated. The interpreter starts the whole cycle recursively until all the top-level goals completes.

The agent produces a node for every Intention Tree not in the pending state, and computes the base priority of each node. For each deadline a node has, agent adjusts the base priority according to its DUF. After computing the interactions between the nodes, the agent can apply these interactions and produce the interaction links. The fairness factor is a user specified value used in fairness calculation of priority. Finally, the agent chooses the (relative) best intention, which is not pointed by final links. If there is no such node, the agent iteratively cancels the final links with the value closest to 0.

5. Simulation and Discussion

5.1. Our Simulation Approach

Because the efficiency of scheduling heavily depends on the tasks the agent will receive, to show the effectiveness, we use a random task generator to issue goals and plans for the agents. The generated goals have random utility, deadlines, and applicable plans, etc. The generated plans have random applicability, actions, ERT, level of subgoalings, and interactions, etc. The value ranges and descriptions of the random parameters are shown in Table 1:

Two more controlling factors are applied to control the random process. The *deadline density* factor is applied to compute the number of deadlines the goals have at a time. The higher the factor, the more deadlines are counted. The *Interaction Density* factor is applied to compute the number of plans having interaction each other. The higher the factor, the more interactions will be.

Parameter	Description	Value range
SubGoalDegree	The subgoalings of a plan	0~3
PlanDegree	The plans of a goal	1~4
TreeLevel	The height of an Intention Tree	2,4,6,8
Step	The steps in a plan	1~12
DeadlineTightness	The time units before the goal timeout	10~100
I-applicability	The applicability of a plan	0.7~1.0
I-utility	The utility of a goal	10~100

Table 1 Parameters in the Random Task Generator

To focus on the scheduling algorithm, there are two kinds of interactions considered only. The first is "Forbid", which causes some other plans to fail. The second is "Promote", which causes some other plans to succeed without execution. Each agent is given 50 top-level goals, and uses different scheduling schemes:

Normal: concern the utility of the top-level goals only

Time-limited: concern the utility of the top-level goals and the deadline

Interaction: concern the utility of the top-level goals and the interactions between the goals

Interaction + Time-limited: concern the utility of the top-level goals, the deadlines and the interactions between the goals

After the execution, the agent will report three values, which are:

SU: the sum of utility of the goals that succeed / total utility.

FU: the sum of utility of the goals that fail / total utility.

TU: the sum of utility of the goals that are timeout / the sum of utility of the goals with deadline.

The results are showed in following sections.

5.2. Simulation Results

In Figure 7, the Normal TU increases rapidly when the deadline density increases. The Time-limited FU is slightly higher than the Normal FU. On the other hand, the performance in the Time-limited model is too low and does not increase. That is, Time-limited scheme will cause more failures because it doesn't consider about interaction

Figure 8 shows the result of the agents using Normal and Interaction scheduling schemes. The deadline density is set to 0, that is, no goals will be timeout. The interaction is changed from 0, 0.1 to 0.3. The figure shows that when there are more interactions between plans, the Normal scheme will fall drastically.

Figure 9 shows the result of all four kinds of schemes when the deadline density varies from 0.1 to 1 and the interaction density is set to 0.1. The SU of Deadline and Interaction+Time-limited schemes fall not quickly because they both consider deadline. The Interaction scheme has higher SU than Time-limited scheme when the Deadline Density is low.

However, when deadline starts to contribute more failures than interaction, the SU of Time-limited scheme is better than the SU of Interaction scheme. SU of "Interaction+ Time-limited" scheme remains the

highest among four until the deadline density is close to 0.9. Nevertheless, in the real application, deadline density seldom reaches so high. Thus, the Interaction+ Time-limited scheme is generally good in most situations.

Figure 10 shows the result of all four kinds of schemes when the interaction density varies form 0.1 to 0.5 and the deadline density is set to 0.3. The SU of Interaction+ Time-limited scheme remains the highest among the four.

6. Conclusion & Future Work

In the systems with multiple agents of mobility and intelligence, a good intention-scheduling scheme might greatly improve BDI agents' performance. We aimed at developing an effective intention-scheduling scheme, discussing the various factors affecting the intention scheduling, and proposing an Intention Tree model to describe the structure and behavior of the agents. Then we use this model to gather the information needed in the scheduling process. An intention-scheduling algorithm, which considers the utility, time constraints and task interactions at the same time, is also constructed. Finally, we show the effectiveness of our approach in a simple simulation.

There are several problems not thoroughly discussed in this paper. Examples include automatic interaction discovery, usages of Intention Tree structure in intending process, intention reconsideration, and other scheduling considerations in multi-agent environment. In the future, we plan to delve into these problems and integrate them to produce a more efficient BDI agent system.

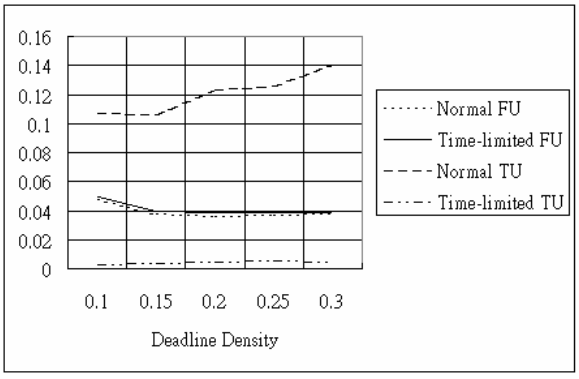

Figure 7 Normal versus Time-limited–FU, DU

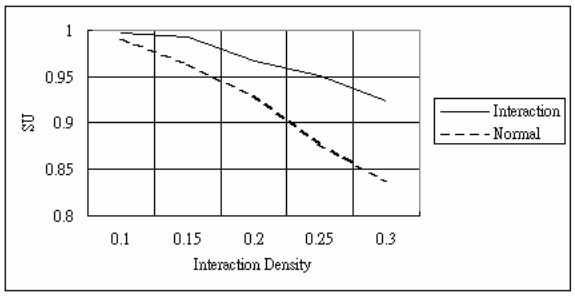

Figure 8 Normal versus Interaction – SU

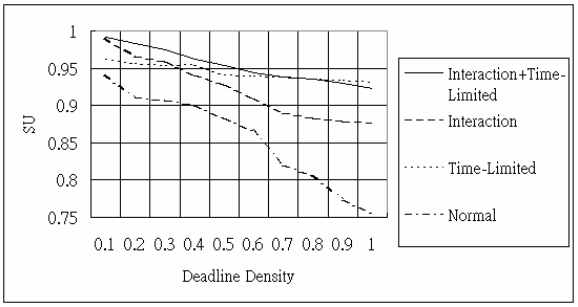

Figure 9 SU of all schemes while Deadline Density changes

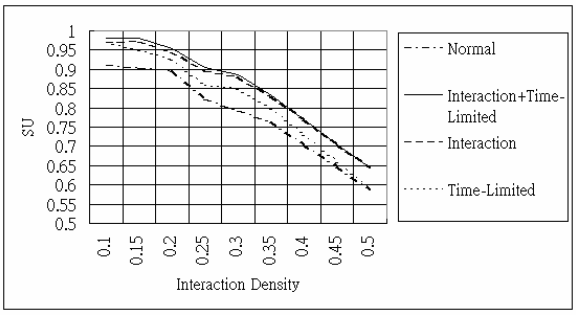

Figure 10 SU of all schemes while Interaction Density changes

7. References

[1] Paolo Busetta and Kotagiri Ramamohanarao, "An Architecture for mobile BDI Agents." Proceedings of the 1998 ACM symposium on Applied Computing, 1998, pp. 445-452.

[2] Bratman, M.E(1987) : "Intentions, Plans, And Practical Reason." Harvard University Press, Cambridge, MA, US, 1987. ISBN (Paperback): 1575861925

[3] Chia-Lin Hsu, Hwai-Jung Hsu, Da-Ly Yang and Feng-Jian Wang. "Constructing a Multiple Mobile-BDI Agent System." The 14th Workshop on OOTA,2003.

[4]Yoav Shoham. “Agent-Oriented Programming.” Artificial Intelligence, March1993. ISSN 0004-3702, pp.51-92

[5] Philip R. Cohen and Hector J. Levesque. “Intention is Choice with Commitment.” Artificial Intelligence, March 1990, pp. 213-261.

[6] Michael P. Georgeff and Amy L. Lansky. “Reactive Reasoning and Planning.” In Proceedings of the Sixth National Conference of the American Association for Arti_cial Intelligence (AAAI'87), volume 2, Seattle, WA, Morgan Kaufmann, July 1987, pp. 677-682

[7] Marcus J. Huber. “JAM: A BDI-theoretic Mobile Agent Architecture.” In Proceedings of the third annual conference on Autonomous Agents, Seattle, Washington, United States, 1999. ACM Press. ISBN 1-58113-066-x, pp. 236-243.

[8] Anand S. Rao and Michael P. Georgeff. “Modeling Rational Agents within a BDI Architecture.” Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), San Mateo, CA, USA, 1991. Morgan Kaufmann. ISBN 1-55860-165-1, pp. 473-484.

[9] Anand S. Rao and Michael P. Georgeff. “BDI-Agents: From Theory to Practice.” In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, USA, June 1995.

[10] Michael P. Georgeff and François F. Ingrand. “Decision-Making in an Embedded Reasoning System.” In Eleventh International Joint Conference on Artificial Intelligence (IJCAI' 89), volume 2, Morgan Kaufmann, August 1989, pp. 972-978.

[11] Michael J. Wooldridge and Nicholas R. Jennings. “Intelligent Agents: Theory and Practice.” Knowledge Engineering Review, 10(2), June 1995, pp.115-152.

[12] Jaeho Lee, Marcus J. Huber, Edmund H. Durfee, and Patrick G. Kenny. “UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications.” In Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS’94), Houston, Texas, March 1994, pp. 842-849.

[13] K. L. Myers and D. E. Wilkins, "The Act Formalism." Version 2.2, SRI International Artificial Intelligence Center, Menlo Park, CA, September 1997.

[14] URL: http://www.marcush.net/IRS/irs_downloads.html

[15] Rafael H. Bordini, Ana L.C. Bazzan, Rafael de O. Jannone, Daniel M. Basso, Rosa M. Vicari.” AgentSpeak(XL): Efficient Intention Selection in BDI Agents via Decision-Theoretic Task Scheduling.” (AAMAS’02), Bologna, Italy, pp.1294-1302.

[16] K. S. Decker and V. R. Lesser. “Quantitative modeling of complex environments.” International Journal of Intelligent Systems in Accounting, Finance and Management, 2(4), 1993, pp. 215–234.

[17] T. Wagner, A. Garvey, and V. Lesser. “Criteria-directed heuristic task scheduling.” International Journal of Approximate Processing, Special Issue on Scheduling,19(1–2), 1998, pp. 91–118.

[18] V. R. Lesser. “Reflections on the nature of multi-agent coordination and its implications for agent architecture.” Autonomous Agents and Multi-Agent Systems, 1(1), 1998, pp. 89–111.

[19] John Thangarajah, Lin Padgham and James Harland. “Representation and Reasoning for Goals in BDI Agents.” In Twenty-Fifth Australasian Computer Science Conference (ASCS2002), Melbourne, Australia.

[20] John Thangarajah, Michael Winikoff , Lin Padgham and Klaus Fischer. “Avoiding Resource Conflicts in Intelligent Agents.” Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002, Lyon, France), pp. 18-22.

[21] John Thangarajah, Michael Winikoff and Lin Padgham. “Detecting & Exploiting Positive Goal Interaction in Intelligent Agents.” In Proceedings of the 2nd international joint conference on Autonomous agents and multi-agent systems (AAMAS ’03), Melbourne, Australia, pp. 401-408

[22] John Thangarajah, Lin Padhgam, Michael Winikoff. “Detecting and Avoiding Interference Between Goals in Intelligent Agents.” In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003): Acapulco, Maxcio, pp.721-726.

[23] Zu-Nien Lin. “Intention Scheduling for BDI Agent Systems.” August 2004.