CIMSA 2009 - International Conference on Computational Intelligence for Measurement Systems and Applications
Hong Kong, China
May 11-13, 2009

On the Hardness of Negotiations in Multi-Agent Systems

Mehran Ziadloo
Computer Engineering Dep.
Iran University of Science and Technology
Tehran, Iran
mehran@method.ir

Siamak Sobhany Ghamsary
Method Ltd.
Tehran, Iran
siamak@method.ir

Nasser Mozayani
Computer Engineering Dep.
Iran University of Science and Technology
Tehran, Iran
mozayani@iust.ac.ir

Abstract – **In multi-agent negotiation the difficulty of the problem depends on how many issues are under negotiation and how complex agents' utility functions are. In this paper we propose a framework for evaluating different techniques for solving negotiation problems and used it to show how hard a negotiation problem can become. We used mediated single text negotiation protocol with genetic algorithms mediator and hill climber agents. Negotiations were conducted over deals with binary issues presented as binary strings. Utility functions with binary and higher levels of dependency between issues were used. Our results show that size of problem does not affect performance, until higher levels of dependency between issues are presented in utility functions. Genetic algorithm method was able to solve the problem with relatively good performance in all levels of dependency that we tested.**

Negotiation; multi-agent systems; binary issues; multi level dependency

I. INTRODUCTION

Negotiation is an important technique in multi-agent systems. This technique can be used for different types of problems, yet two of them are well studied: resource allocation and task assignment [1]. In this technique a group of agents negotiate over a set of issues until they come to an agreement. Each issue has at least one attribute for which an agreeable value should be found. A deal is an assignment of values to all issues' attributes. Agents try to find best values for these attribute considering their own utility (competition) and possibly community's overall utility (coordination). If no such deal is found, a predefined one will be applied. In this research we focus on coordination which is often used as a distributed problem solving (DPS) technique. In such negotiations, agents fully trust each other since they are all part of the same solution.

Negotiations usually include a protocol as their definition. A protocol specifies topology for communications, transmitted information and some rules to make sure that negotiation is going as planned (strategy). One of the earliest protocols for negotiation is Rubinstein's alternating offers model [2] which supports only two agents and is not suitable for our purpose. In this work we used mediated single text negotiation protocol [3]. This protocol supports arbitrary number of agents and can work with different types of strategies. Mediator in this protocol plays a unique role; it has responsibility of proposing new deals to agents and gathering their opinions about them. Then it should try to find better deals to propose according to previous responses. In original version of this protocol, agents would only respond with yes or no to deals. But in [4] it has been shown that having agents respond with richer information can improve performance of this protocol. This expansion requires a trust between agents which is not a problem in our case since we based our research on DPS.

Rest of this paper is organized as following: section two covers preliminary concepts. In section three we describe the problem structure. Section four presents our framework and section five the evaluation method. In section six and seven results and conclusion are presented.

II. PRELIMINARY CONCEPTS

A deal is a set of issues. Once a deal is accepted by all the agents in negotiation, we call it an agreement. A deal is defined in two spaces: issue space and utility space. A deal in issue space is a vector of issues' attributes. Since each agent assigns a utility to each deal, we can also show a deal by a vector of agents' utilities. We show a deal in issue space with $\vec{x}$ and with $\vec{z}$ in utility space. $\vec{x}_i$ and $\vec{z}_i$ indicates ith issue and ith agent's utility respectively. $|\vec{x}|$ will be the number of issues in the deals while $|\vec{z}|$ is the number of agents in negotiation. Considering two deals in utility space, $\vec{z}^1$ and $\vec{z}^2$, in a negotiation, they can have four relations in respect to each other.

- $\vec{z}^1$ and $\vec{z}^2$ are equal (but not necessarily identical) if: $\forall_i : \vec{z}_i^1 = \vec{z}_i^2$. (they are identical if they are the same in issue space; $\forall_i : \vec{x}_i^1 = \vec{x}_i^2$)
- $\vec{z}^1$ dominates $\vec{z}^2$ if: $\forall_i : \vec{z}_i^1 \geq \vec{z}_i^2 , \exists_j : \vec{z}_j^1 > \vec{z}_j^2$
- $\vec{z}^2$ dominates $\vec{z}^1$ which is opposite of previous relation.
- $\vec{z}^1$ and $\vec{z}^2$ are non-dominant to each other, which is the case when none of the above situations applies.

If Z is a set of all possible deals, those deals that are not dominated by any other deal in Z are called nondominant or *Pareto optimal*. And the set of all Pareto optimal deals in Z is known as *Pareto frontier* of Z. Finding Pareto frontier of a set is the purpose of multi-objective optimization (MOO) algorithms. In this paper, we want to compare results of our own framework with results of a bunch of MOO algorithms from MOMHLib++ library [5]. To choose from Pareto optimal deals we should introduce other criteria. We will use Nash Bargaining Solution and Utilitarian Deal as evaluation criteria. In Nash Bargaining Solution, we are looking for a deal which maximizes product of agents' utilities while in Utilitarian Deal we want to maximize sum of agents' utilities. Such criteria are defined not on the Pareto frontier but on the complete set of Z, and it has been proven that deals which satisfy any of these two criteria belong to Pareto frontier [6].

III. Problem Definition

Definition of a negotiation problem includes the number of participating agents, the number and type of the issues and a utility function for each agent. In this work we will conduct our negotiations between two agents. Deals will be binary strings of 30 to 100 bits (to compare their hardness), presenting each bit as an issue. Such a deal structure creates up to a state space of 2^{100} (more than 10^{30} deals), which is too huge to be searched exhaustively. Agents' utility functions have a crucial part in a problem's difficulty. In simplest case, the contribution of each issue in a deal to agent's utility is independent of other issues. That is, when the issue is present (its bit is 1) some predefined, fixed value is added to the agent's utility. Such utility functions are called linear which result in relatively easy negotiation problems [4]. More complex problems arise when using nonlinear utility functions. In these functions, in addition to independent impact of issues, combinations of issues also affect agent's utility. It means that presence of two or more specific issues can have more value to utility of an agent than the addition of their values. Solving such problems is much harder since they have so many local optimums which make it hard to find the global ones [7]. In Fig. 1 examples of such utility functions are plotted.

Finding global optimums in negotiation problems with nonlinear utility functions is near impossible when state space is relatively large. Because to make sure a deal is globally optimal one should review the whole space. In this case, the best thing to do is to estimate global optimums (using different search algorithms).

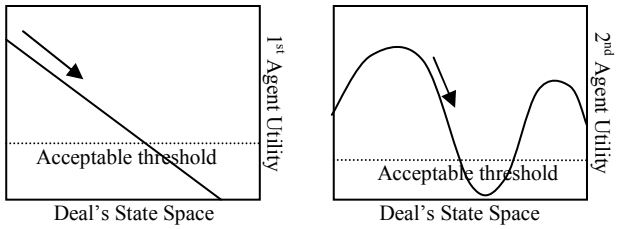

Figure 1. Examples of different utility function.
Left: Linear function. Right: Nonlinear function

In this paper we have used nonlinear utility function with second level and third level of dependencies. To implement such functions we used hyper matrices known as influence matrices. To implement such matrices we designed a special indexing into a linear array as you can see in (1):

$$H\underbrace{_{[1..I]}{}^{i}{}_{[1..i]}{}^{j}{}_{[1..j]}{}^{k}{}_{[1..k]}{}^{\ldots}{}_{\ldots}}_{D \leq I} . \quad (1)$$

In (1) H is the influence matrix with D dimensions (dependency of level D) and the range of each dimension is bounded by the index of previous dimension. The first dimension, like the number of dimensions, is bounded by number of issues (I). Having H as an influence matrix of an agent and $\vec{x}$ as a deal in issue space, Fig. 2 shows the pseudo code to calculate agent's utility.

IV. Solution

Our solution is a combination of mediated single text negotiation, hill climber agents and a genetic algorithm mediator. Our mediator gathers true utility values of agents to search for the best possible deal which is only applicable when there is enough trust between agents. In our framework, the mediator generates a deal by GA and proposes it to agents. Agents will respond by their true utility values and whether they accept the deal or not. True utility values are used as fitness for GA. We start by describing agent design.

Latest agreement which is initialized by a predefined deal is used by agents to decide on new deals. An agent accepts a deal only when its utility is higher than some predefined threshold and the previous agreement. This is called hill climber agent design. Last part of our framework is mediator's deign which is implemented by a simple genetic algorithm. Since the deal is a binary string, we used the deal itself as the chromosome. A uniform crossover is used with equal parent selection probability (50-50). As for the mutation a simple bit flipping is applied over all issues with small probability of 0.001.

```
Utility = 0
for i1=0 to I
  for i2=0 to i1
    ...
      for iD=0 to iD-1
        if x⃗[i0] = 1 and ... and x⃗[iD] = 1 then
          Utility += H[i1][i2]...[iD]
```

Figure 2. Pseudo code to calcualte agents' utilities.
H: influence matrix, I: number of issues, Deal: proposed deal, Utility: utility

Fitness function definition is relative to the evaluation criteria, Nash or Utilitarian. For Nash, fitness is the product of agents' utility values and for Utilitarian it is sum of the values. The rest of design decisions for GA are the same as in simple genetic algorithms (SGA) [8].

V. EVALUATION METHOD

To evaluate an agreement we need to find its distance to the best known deal. Different distance concepts should be defined for different criteria. We define an error term for a deal as the maximum distance of that deal in utility space to the locus of all deals with best global utility. Fig. 3 shows how to measuring a deal's error.

In Fig. 3-(a) circle mark presents a deal with best global utility according to Nash criteria while cross mark is a sample deal. The y=a/x curves indicates locus of deals with the same Nash value. Error of deals are measured on the y=x line because it is the largest distance between two curves. The same goes for Fig. 3-(b) showing locus curves for Utilitarian criteria which are y=a - x lines.

To convert the error into percentage, we based the origin as the default deal where no agent gains anything and divided the error value by the distance of best global utility's locus on the line of y=x to origin. One hundred minus the error percentage gives us the performance percent of each deal.

VI. RESULTS

We have tested our framework in two phases. First we applied it on a small group of problems to see its effectiveness on binary dependency. Then, in phase two, a complete set of problems with different dependencies and different number of issues were considered. In both phases we used 10 algorithms implemented in MOMHLib++ library to find the Pareto frontier which is an estimation of true answer. We ran each of the algorithms 10 times and gathered the results into one set. Then best deals of Nash and Utilitarian criteria were found by an exhaustive search in the resulting set. Problems in our experiment are generated as discussed in section three. Influence matrices of our agents are filled with random real numbers between -1 and 1.

In first phase, we instantiated 10 randomly generated problems with 100 binary issues and binary dependencies. GA's population size was set to 500 and number of generations to 1000. To minimize the effect of finding results by chance, we solved each problem ten times and calculated an average and a standard deviation instead. Fig. 4 and Fig. 5 show the progress of GA in solving problems as achieved performance percentage over the number of fitness evaluations. There is 5000 fitness evaluation between each two following columns.

As for the second phase we considered a larger and more diverse set of problems. We tested problems of three dependency levels: linear (D=1), binary dependency (D=2) and third level of dependency (D=3). In each level, number of issues was changed from 30 to 100 with intervals of 10. This time, 40 randomly generated problems were used for each configuration to reduce the effect of randomness.

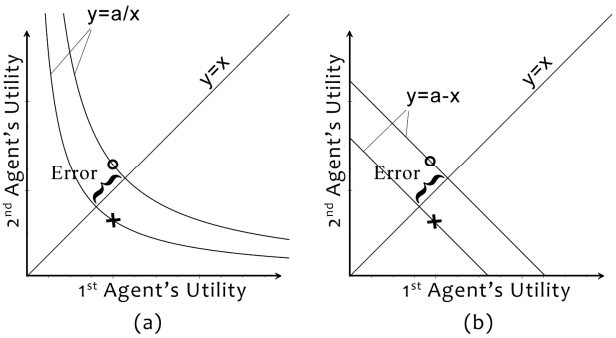

Figure 3. Error measure of deals in (a) Nash and (b) Utilitarian

This time we set the population size to 1000 since we wanted to maintain the diversity of GA's population and we observed from the first phase's results that long before generation 1000 GA's work is done, so the number of generations was set to 500 in second phase.

Fig. 6 shows results of second phase's averages and Fig. 7 is its standard deviation for Nash criteria. These values are plotted over the number of issues under negotiation. As you can see in these figures, the impact of dependency level to performance is much higher than number of issues. In Dep. 1 (linear utility function), as the number of issues increases, it seems the problems become easier for GA to solve. This behavior can be explained regarding the diversity of GA population. As the number of issues go up; while the number of optimums remains the same, cardinality of state space increase and lets the GA to keep its diversity and as the result, the problem is solved easier. But number of issues plays its part as the level of dependency increases. The unpredictable growth of curve in third level shows that the difficulty of problems is more related to dependency level (and hence complexity if utility functions) than the number of issues. The average and standard deviation of problems with 80 issues of Dep. 3 show that there could be problem in this configuration harder than a problem with 90 issues of the same dependency. This means since parameters of problems are set randomly, evaluating some solution's performance requires a rich set of problems before one can claim how good his/her solution performs.

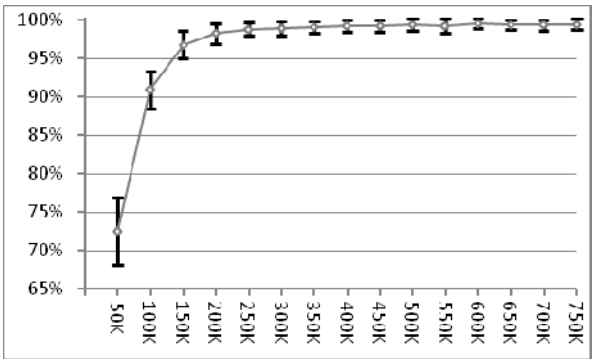

Figure 4. GA's progress over the number of evaluations for Utilitarian criteria

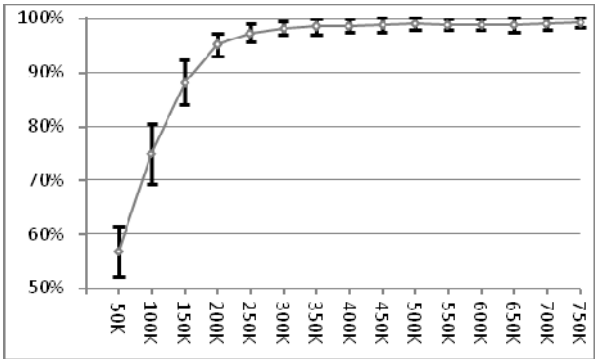

Figure 5. GA's progress over the number of evaluations for Nash criteria

The hardness of problems in second or third level increases with the number of issues. This means that if we define a ratio as the number of local optimums per state space cardinality, this ratio gets higher by the growth of issue number. This ratio has a direct relation to the hardness of problem because as the ratio goes up, there are fewer individuals to play the role of the global optimum's neighbors. And this makes problems harder for GA to solve since it's harder for GA to maintain its population's diversity. Such analysis can be proven once some diversity maintenance is applied to GA which is beyond this paper. As we monitored our GA's generations, population's individuals lose their diversity so soon and most of the job is done by mutation after a while because crossover cannot create unseen individuals anymore.

VII. CONCLUSION

In this paper we performed a set of tests to see if a simple genetic algorithm is suitable to search over the deals in a multi-agent negotiation to find the Nash Bargaining Solution and Utilitarian Deal. Results showed that for linear utility functions and binary dependency utility functions the framework works well enough, but as the level of dependency goes to three it becomes much harder to solve problems. SGA seems to lose population diversity and tend to trap in local optimums easily. Sophisticated diversity control techniques could help SGA to keep its efficiency even for higher dependency levels which we leave for future work. It is also a good practice to test other search methods like cellular automata or ant colony optimization. To take the research to another step, we recommend using real valued issues. This kind of negotiations is much harder to solve and requires other techniques like CMA-ES [9].

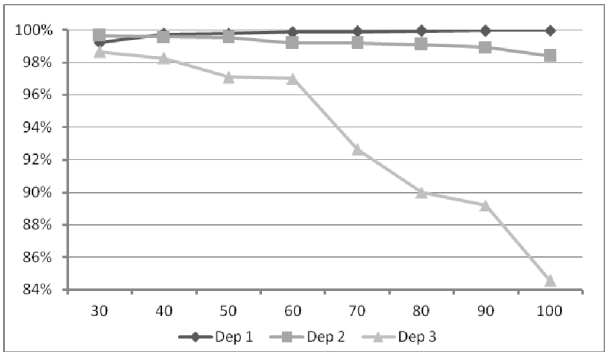

Figure 6. Average of performance against number of issues for different levels of dependency

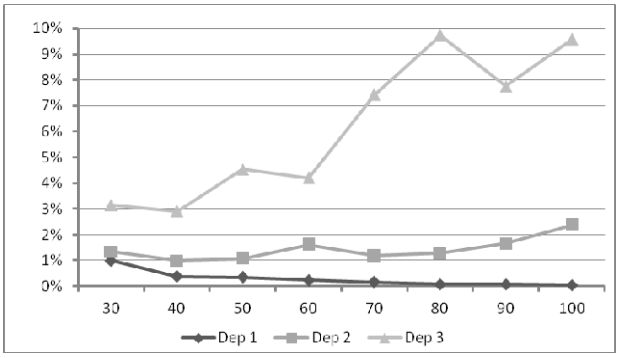

Figure 7. Standard deviation of performance against number of issues for different levels of dependency

REFERENCES

[1] Woolridge, M. and M.J. Wooldridge, Introduction to Multiagent Systems. 2001: John Wiley & Sons, Inc. New York, NY, USA.

[2] Rosenschein, J.S. and G. Zlotkin, Rules of encounter. 1994: MIT Press Cambridge, Mass.

[3] Raiffa, H., The Art and Science of Negotiation. 1985: Belknap Press.

[4] Klein, M., et al., "Negotiating Complex Contracts". Group Decision and Negotiation, 2003. 12(2): p. 111-125.

[5] Jaszkiewicz, A., "A Comparative Study of Multiple-Objective Metaheuristics on the Bi-Objective Set Covering Problem and the Pareto Memetic Algorithm". Annals of Operations Research, 2004. 131(1): pp. 135-158.

[6] Nash Jr, J.F., "The Bargaining Problem". Econometrica, 1950. 18(2): p. 155-162.

[7] Bar-Yam, Y., Dynamics of Complex Systems. 2003: Westview Press.

[8] Goldberg, D.E., "Simple genetic algorithms and the minimal, deceptive problem". Genetic Algorithms and Simulated Annealing, 1987. 74.

[9] Hansen, N. and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies", Evolutionary Computation, 9(2), pp. 159-195, 2001