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

A Framework to Evaluate Multi-Objective Optimization Algorithms in Multi-Agent Negotiations

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—**Multi-objective optimization algorithms are designed to find Pareto frontier set. This set plays a major role in multi-agent systems' negotiations. Different applications might be interested in different parts of Pareto frontier. In this paper we present a framework to show how a multi-objective optimization algorithm is evaluated against others. We used eleven algorithms implemented in MOMHLib++ library to test our framework on a two agent negotiation of binary issues and binary dependency. But our framework is easily expandable to higher number of objectives and all types of negotiations. Our analysis shows that a single scalarization value of Pareto frontier is not enough to compare multi-objective optimization algorithms, as it is done in most cases.**

Multi-objective optimization; Pareto frontier; multi-agent systems; negotiation

I. INTRODUCTION

Negotiation techniques are used in multi-agent systems for different tasks. In general, negotiation includes two or more agents which interact over a number of issues to come to an agreement, we call a deal. The goal of each agent is to maximize its own utility which is defined as a function of debatable issues. Deals must be mutually acceptable between agents. Upper bound of deals in utility space is Pareto frontier explained in detail later.

In this paper we have applied eleven different algorithms of multi-objective optimization to find Pareto frontier of problems with two agents and 100 binary issues. The algorithms are chosen from MOMHLib++ [1]. Each algorithm was allowed to run within a fixed amount of time. The constant execution time together with the fact that all the algorithms share most of their code, allow an accurate comparison of performances. The purpose of this paper is to show how to evaluate such algorithms. Selected algorithms are: MOGLS, PSA, PMA, IMMOGLS, SMOSA, MOSA, NSGA, NSGAII and NSGAIIC, SPEA and a simple multiple-objective multiple start local search, MOMSLS. For more information about each algorithm, and their references please refer to [1].

The rest of paper is organized as follow. First we present some preliminary concepts in section two. Then, in section three, we will define the problem. Evaluation framework and related topics are detailed in section four. Finally, results and conclusion will be presented in section five and six.

II. PRELIMINARY CONCEPTS

Woolridge defines negotiation as a technique for agents to reach agreements on matters of mutual interest [3]. Each 'matter' is called an issue and a set of issues is called a deal. There are two kinds of deals, acceptable and unsatisfactory. A deal is acceptable if all the agents in negotiation accept it. We define $\vec{x}$ a vector of issues to show a deal in issue space and $\vec{z}$ a vector of utilities to show a deal in utility space, while $\vec{x}_i$ and $\vec{z}_i$ indicates i[th] issue and i[th] agent's utility respectively. Comparison of deals is done in utility space with the concept of dominance. A deal $\vec{z}^1$, dominates $\vec{z}^2$, $\vec{z}^1 \succ \vec{z}^2$, if $\forall_i$: $\vec{z}_i^1 \geqslant \vec{z}_i^2$ and $\exists_j$: $\vec{z}_i^1 > \vec{z}_i^2$. A deal that is not dominated by any other deal is called nondominand or *Pareto optimal*. The set of all Pareto optimal deals for a problem is called *Pareto frontier*. Finding Pareto frontier is the aim of multi-objective optimization algorithms. There is no preference over the set of Pareto frontier members and all Pareto optimal deals are of the same value to us unless some extra criteria are introduced. One of the most important criteria in this field is Nash Bargaining Solution. Nash Bargaining Solutions are those deals that maximize the product of agents' utilities. To indicate the hardness of such problems, special cases of negotiation can be reduced to the *set covering problem* which is of NP-complete complexity [4].

III. PROBLEM STRUCTURE

The definition of a negotiation problem includes: the number of agents, a structure for the deal and utility functions for each agent. In this paper we have tested a two-agent

978-1-4244-3820-4/09/$25.00 ©2009 IEEE

problem with 100 binary issues. A deal in this problem is presented by a bit string with 100 bits. Nonlinear utility functions are used which include binary dependencies between issues. By dependency between issues we mean co-occurrence of them can convey extra meaning and affect utility of an agent. A problem with no dependency between issues is considered linear and is relatively easy to solve [5]. In our paper issues have a utility of their own and an additional contribution when happen with other issues. Such problems consist of a lot of local optimums [2]. To implement a utility function with binary dependencies, we used an influence matrix with elements of random values between -1 and +1. Each element in such a matrix shows the influence of enabling two bits of a deal simultaneously. For example, element in location *(i, j)* of matrix is the utility of issue *i* and issue *j* when both enabled. Values on main diagonal of the matrix indicate the influence of issues individually. Obviously, such a matrix is symmetric and only half of it completely presents the utility function. Thus, having a deal $\vec{x}$ as a vertical vector, and influence matrix H, one can compute the agent's utility by (1).

$$u = (\vec{x} \times \vec{x}^{T}) . H . \quad (1)$$

Considering a deal with 100 issues, it constructs a problem with input space of 2^{100} cardinality. Such huge spaces cannot be searched exhaustively; search algorithms are of absolute need to find the optimums. In this paper we use random instances of this problem to evaluate selected algorithms. As mentioned before, there is usually more than one optimum for such problems and one can never be sure that have found them all. There is even no way to be sure that the optimums found, truly belong to Pareto frontier. This property makes it difficult to evaluate quality of best solutions found by each algorithm. Thus, the only possible evaluation approach is inevitably based on comparison between algorithms.

IV. Evaluation Framework

An accurate assessment of search results requires a concrete ground truth which is not available for our problem. As discussed in previous section, only comparison based evaluations are possible for this kind of problems. We have used three criteria to compare algorithms' results: average distance of deals to a reference point in utility space, distribution of found deals in utility space, and the total number of nondominant deals found by each algorithm.

A deal can be plotted as a point in utility space by mapping each agent's utilities to a coordinate axis. In order to compare two deals in a negotiation, a common way is to calculate a scalar value for each of them and compare them as their representers [1]. To do so, we need a reference point which in this problem we can assume origin as the reference point without losing generality. Deals are scalarized using (2).

$$Scalarize(\vec{z}) = \frac{\sum_{\vec{\Lambda} \in \Psi} \vec{\Lambda} . \vec{z}}{|\Psi|} . \quad (2)$$

Where $\vec{\Lambda}$ is a weight (scalarize) vector from Ψ, the collection of all possible assignments of weights to agent's utilities. And $\vec{z}$ is the deal to be scalarized. Ψ is defined as (3).

$$\Psi = \left\{ \vec{\Lambda} = [\lambda_1 \dots \lambda_2] \mid \lambda_i \in \left\{ \frac{b}{k}, b = 0 \dots k \right\}, \sum_{i=1}^{I} \lambda_i = 1 \right\} . \quad (3)$$

In (3), *k* is the sampling parameter. Increasing *k*, results in more accurate scalarization. In our experiment *k* was set to 100. We tend to consider the higher scalarized values as indications of better result sets. So now we can represent each deal with a scalarized value and easily compare these values. But a problem arises if we simply compare scalarized values. In Fig. 1 assume dot points are results obtained by algorithm A and cross points by the algorithm B, while all of the points belong to Pareto frontier. Since average magnitude of dot points are higher than cross points, if we calculate a scalarized value for both of them, algorithm A will show better performance. To resolve this problem we divided the utility space into nine sectors, each containing 10° of polar coordinate system as shown in Fig. 1. We would then compare the results of different algorithms only if they belong to the same sector. To be able to aggregate results of different problems with each other, we have normalized the scalarized deals of found results by all the algorithms for each sector separately. This way normalized scalarized deals above the zero are the results better than average and those below zero are worse than average. Equation (4) is normalization formula.

$$NS\_deal_{i,j,k,m} = \frac{(S\_deal_{i,j,k,m} - Avg_{m,k})}{STDEV_{m,k}} . \quad (4)$$

In (4) $S\_deal_{i,j,k,m}$ is the scalarized result of i^{th} run of j^{th} algorithm to solve the m^{th} problem that is placed in the k^{th} sector and $NS\_deal_{i,j,k,m}$ is its normalized value. $Avg_{m,k}$ presents the average of all results found by all the algorithms through all the runs for k^{th} problem in sector m^{th} and $STDEV_{m,k}$ is its standard deviation.

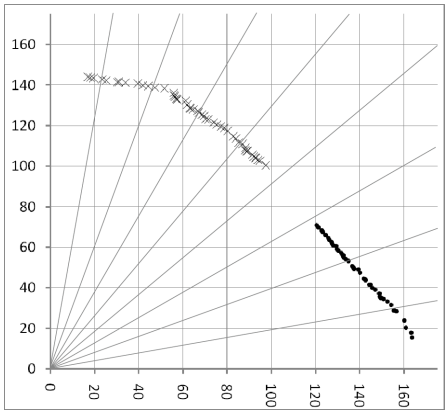

Figure 1. Problem of scalarizing Pareto frontier into one value and its solution.

As for the second criteria, distribution of results over clock is used as another indication of algorithm performance. So we also count the number of deals in each sector to form a distribution per angle. The total number of deals found by algorithms is also used for evaluation. This is important because two mutually nondominant deals are not comparable in general. Therefore, the total number of such deals found by an algorithm could be a good indication of its efficiency for some applications.

V. RESULTS

Before we start with results, it is necessary to specify the way we used MOMHLib++ library. MOMHLib++ defines a framework for algorithm implementation. Each implementation can define its own problem and solution representations together with three operations: Recombination, Mutation and Local Search. The library constructs solutions and applies operations during a unified search process. Algorithms in this library could be divided into four classes. To make a balance between these four classes, we have limited the execution time for each algorithm to 50 seconds so that we can fairly compare their performance with each other. The rest of parameters for each algorithm are set for best performance, based on experience. Here are the four classes:

Simulated Annealing based (SA): There are three algorithms in this class, MOSA, SMOSA and PSA which are all different kinds of SA. In the MOMHLib++, these algorithms only need the Local Search operation to be implemented. We used random bit flips for this operation. That is, whenever a local search is needed, one bit of the deal, representing an issue, is randomly selected and negated.

Genetic Algorithm based (GA): Four algorithms; SPEA, NSGA, NSGAII and NSGAIIC are members of this class. As the name shows, all of the algorithms in this class are variations of GA. Therefore mutation and recombination operator of MOMHLib++ need to be implemented for them. We used the same bit-flip technique explained above as the mutation operator and a uniform crossover as the recombination operator with a fixed probability of 0.5.

Hybrid Genetic Algorithms based (HGA): MOGLS, IMMOGLS and PMA are in this class. Again, all of the algorithms in this class are GAs but they use a recombination operator, combined with a local search. We have implemented recombination method with a uniform crossover, just like the previous class and local search as SA.

Random Start Local Search: The last class has only one member, MOMSLS. This algorithm initializes a random deal and finds its nearest local optimum. We used gray coding to define neighborhood between deals. That means two deals are neighbor if they differ only by one issue.

We used 10 randomly generated problems to avoid any possible bias induced by selecting a specific problem; each problem consists of one influence matrix for each agent filled with random numbers between -1 and +1. We ran each algorithm 10 times for each problem to avoid randomness outcome bias. Each run takes 50 seconds and after this period it was forced to exit and give out the results. Fig. 2 shows the averages, standard deviations and distributions of results for each algorithm per each sector. The analysis of charts in Fig. 4 could be so extensive and we will present just a small part of it. The worst performance belongs to MOMSLS (simplest algorithm) which indicates the hardness of problem. The results of this algorithm are distributed in sectors almost uniformly, which shows that even though a local search is performed after the random start, but since there are a lot of local optimum in problem, the results are not focused in a sector or two and each sector has its own local optimum(s). Three algorithms of MOSA, PSA and SPEA have done impressive among the rest. These algorithms are all above the average in all of the sectors. MOGLS, IMMOGLS, PMA and of course MOMSLS, on the other hand, are the worst ones as their results are all below the average. The rest of algorithms showed intermediate performance, above the average in some sectors and below in the rest. All solutions suffer from the size of standard deviation of their results. Such large standard deviation is an indication of instability. So if you are looking for a confident result, you must run each algorithm a number of times before you can trust the outcome (or use combination of algorithms together).

As for the last criteria, TABLE I shows the average number of deals each algorithm was able to find per each run.

VI. CONCLOSION

In this paper, we evaluated eleven different algorithms for the problem of negotiation between two agents and 100 binary issues with binary dependency. The algorithms were evaluated by applying them to 10 randomly generated problems. Nonlinearity nature of problems and huge input space makes them so hard that no ground truth can be found. So we proposed a framework for comparing different algorithms of multi-objective optimization. Our comparison showed that three algorithms of MOSA, PSA and SPEA are above the average in all sectors of utility space. While NSGA, NSGAII and NSGAIIC are considerable only when we are interested in deals around 45°, especially considering the number of deals these algorithms could find. The algorithms of HGA did not do well among the rest since all of their results are below the average. Yet the worst results belong to MOMSLS, as it was expected. A completely different result set would have gained if we were to use only a scalarized value for each algorithm.

Our evaluation covers only the type of problems with binary issue and binary dependency. Considering problems with higher order dependencies and working on other types of issue, e.g. real numbers are good paths for future works.

TABLE I: Algorithms' average number of deals (Avg. #) found per each run

Algorithm	Avg. #	Algorithm	Avg. #
MOSA	216.37	NSGAIIC	114.39
SMOSA	136.58	MOGLS	147.42
PSA	239.09	IMMOGLS	72.43
SPEA	178.57	PMA	149.04
NSGA	119.22	MOMSLS	9.89
NSGAII	117.79		

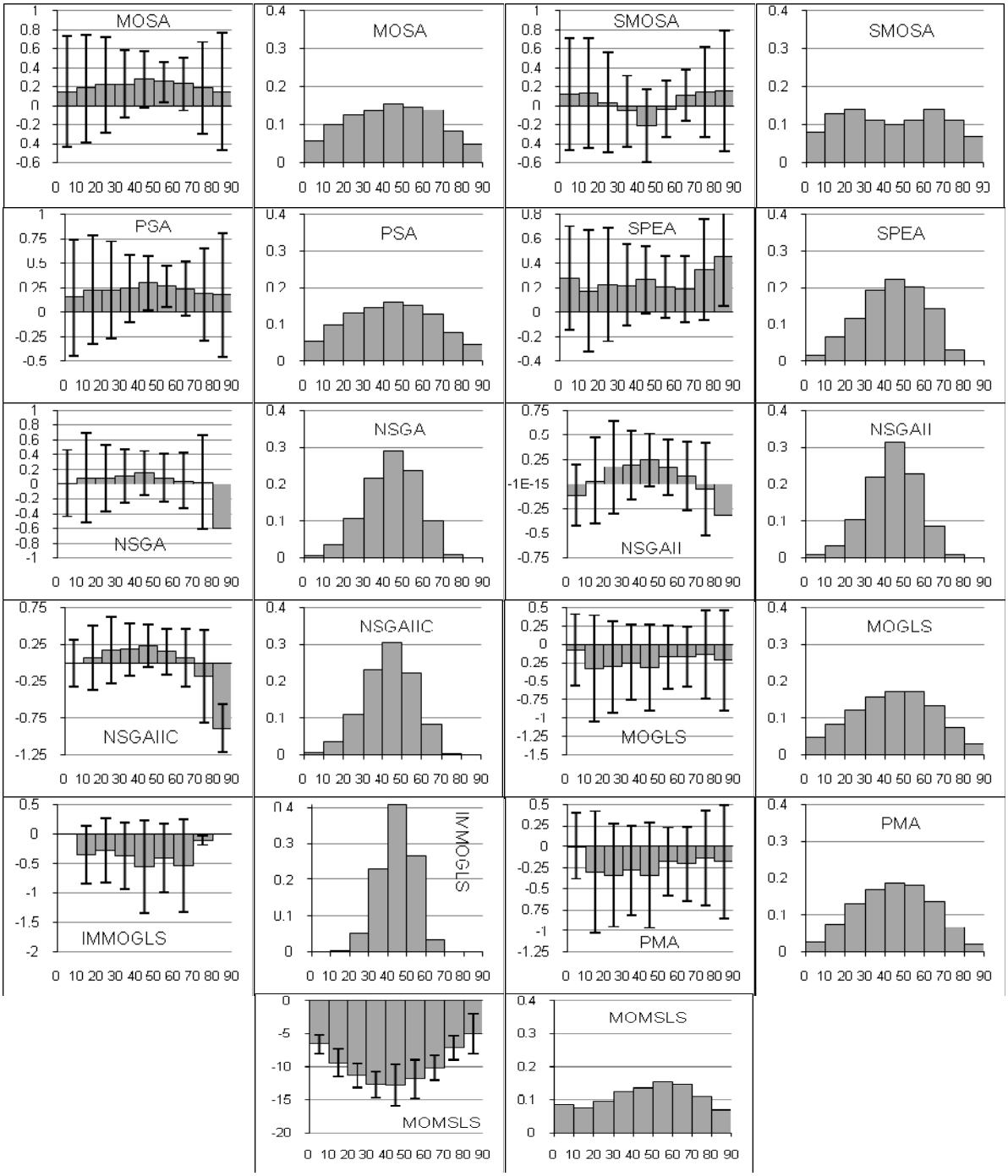

Figure 2. Averages, standard deviations and distributions of deals for each algorithm per sector. Charts are presented in pair, in each pair the left chart shows the averages and standard deviations and the right chart contains the distribution of results over the sectors. Horizontal axis in all of the charts shows ranges of degree from zero to 90° (sectors). Vertical axis in right column of each pair is the ratio of deals' occurrence in each sector. In the left column of each pair, the vertical axis shows the average of normalized scalarized deals. Each chart is labeled with the name of corresponding algorithm.

REFERENCES

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

[2] Y. Bar-Yam, Dynamics of Complex Systems, Westview Press, 2003, p. 172, pp. 211-214.

[3] M. Wooldridge, and M.J. Wooldridge, Introduction to Multiagent Systems, John Wiley & Sons, Inc. New York, 2001.

[4] Garey, M.R. and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.F. Freeman and Co., San Francisco, 1979.

[5] M. Klein, P. Faratin, H. Sayama, Y. Bar-Yam, "Negotiating complex contracts". In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2. 2002, pp. 753-757.