Evolving Optimal Agendas and Strategies for Negotiation in Dynamic Environments

Ahmed Kattan
Loughborough University
Department of Computer Science
Loughborough LE11 3TU, UK
A.J.Kattan@lboro.ac.uk

Shaheen Fatima
Loughborough University
Department of Computer Science
Loughborough LE11 3TU, UK
S.S.Fatima@lboro.ac.uk

ABSTRACT

Two key problems in a negotiation are: i) for the players to decide *what issues* to include in a negotiation, and ii) *what strategy* to use for negotiating over the chosen issues. In general, there will be many (say m) issues available for negotiation and the players must choose a subset of $g < m$ issues and negotiate on them. The g issues thus chosen is called the *negotiation agenda*. An agent will choose the agenda that maximizes its utility and is therefore its *optimal agenda*. Once an agenda is chosen, a player must choose an *optimal strategy* since the final deal (and a player's profit from it) will depend on the strategy. In many real-world negotiations, a player's actual utility from a deal will not be defined completely. There may be some information about the utility but this information may not be complete. Such scenarios make the problems of finding optimal agendas and strategies more challenging. In order to overcome this challenge, we present a *multi surrogate-based* GA system. This system is comprised of two GAs and set of Radial Basis Function Network (RBFN) surrogates that work together to find an optimal agenda and also an optimal strategy for that agenda. We compare the performance of this system with a standard GA and random search.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Game Theory; I.2.8 [**Heuristic Methods**]

General Terms

Algorithms

Keywords

Negotiation, Dynamic Fitness Function, Surrogate, RBFN.

1. MOTIVATION

In this work, we focus on negotiations that involve multiple objects/issues such as the following one. A buyer wants to buy more than one car from a second hand car dealer. Assume that the buyer wants to buy g cars and the seller has $m > g$ cars for sale. Clearly, the buyer will want to negotiate a deal that will be best for her/him, i.e., a deal that will maximize her/his profit. In order to get the best deal, the buyer must solve two key problems: **Q)** s/he must identify the best set of g cars to negotiate the price for, and **P)** negotiate the best price for those cars. Here, the buyer can individually decide which of the g cars to negotiate upon. The g cars chosen by the buyer will form the *negotiation agenda* and a price must be negotiated for each issue on the agenda.

The problem of finding optimal agendas and optimal strategies was addressed in [1]. In this work, the authors used GAs with a single surrogate and showed that their approach performs better than a standard GA. However, this work is based on the complete information assumption, i.e., the utility functions are known before negotiation begins. Also, the utility functions are assumed to be *static*, i.e., they do not change with time. This scenario may be enough to ascertain the effectiveness of a surrogate-based approach for solving the problems **P** and **Q**. However, it is necessary to ensure that any approach is also well suited to real-world negotiations.

2. THE PROPOSED SOLUTION

We use a *divide and conquer* approach to solve the problems P and Q. The proposed multi surrogate-based GA system is comprised of the following three main components, *i)* an *outer GA* component for solving the problem Q (Here, Q requires searching the space of $C(m, g) = \frac{m!}{(m-g)!}$ possible agendas to find the one that yields highest equilibrium utility to a player), *ii)* an *inner GA* component for solving the problem P (i.e., for a given agenda, determine the buyer's equilibrium utility – this is a constrained nonlinear optimization problem) and *iii)* a set of n *surrogates* and a *choice mechanism* for choosing one of these, in order to speed up the search in the two GAs.

The population for the outer GA is comprised of individuals that represent agendas. Each individual is a binary string of length m and contains g ones and $m - g$ zeros. Each bit in the string corresponds to a negotiation issue. Within a string, ones indicate that the corresponding issue is included in the agenda. A zero indicates that the corresponding issue is not on the agenda.

The inner GA acts as a fitness evaluator for the outer GA's individuals, i.e., agendas. For a given agenda, the inner GA finds an equilibrium offer/strategy and the associated utility. The utility is calculated by optimising a constrained continuous optimisation problem that represents the equilibrium between the negotiation parties.

This equilibrium utility is the *fitness* (i.e., the profit) of the agenda. Here, the inner GA takes an agenda from the outer GA as input, and evaluates the equilibrium utility for the agenda. In this way, any agenda in the population of the outer GA can be evaluated by running the inner GA.

The population for the inner GA is comprised of individuals that correspond to possible strategies for an agenda (of the outer GA). The fitness of an individual in the inner GA is the utility for that strategy. The individual/strategy with the highest fitness/utility is the optimal strategy for the corresponding agenda. Thus, the best individual in the inner GA is nothing but the optimal strategy for the corresponding agenda. Each individual in the inner GA is a string of m real numbers in the interval $[0, 1]$.

To find an optimal agenda, we must run the whole inner GA for every individual of the outer GA. This must be repeated for a number of generations. Clearly, this can be computationally very expensive. Hence, to speed up the search, we used a *surrogate* system based on RBFNs [2]. This system is comprised of n surrogates.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.

	Proposed Method			**Standard GA**			**Random Search**		
(m,g)	*Average*	*Max*	*StdDev*	*Average*	*Max*	*StdDev*	*Average*	*Max*	*StdDev*
20,4	**246.49**	280.25	25.72	195.72	224.35	26.35	156.49	193.98	22.64
20,10	**308.03**	358.83	18.39	197.95	252.55	19.80	199.40	250.55	18.75
20,15	**396.72**	407.85	6.07	241.33	250.14	5.29	239.07	242.99	3.45
30,6	**381.90**	416.85	20.00	309.48	367.70	31.00	272.29	347.43	33.55
30,15	**507.48**	533.01	15.89	353.91	409.77	21.62	328.04	349.14	14.93
30,20	**720.08**	759.76	16.15	511.63	537.95	22.08	500.68	535.29	30.41
40,8	**814.39**	909.84	39.86	613.01	669.34	35.79	609.64	651.30	40.81
40,20	**834.93**	970.73	123.40	566.63	711.96	94.77	557.61	721.83	100.61
40,25	**886.55**	988.71	56.10	611.18	660.46	42.77	605.14	675.04	46.04
50,10	**695.28**	764.04	34.89	544.97	616.24	53.43	531.93	609.10	49.91
50,25	**983.34**	1104.17	51.13	734.70	863.68	79.74	813.73	909.12	73.92
50,30	**975.55**	1045.55	37.87	701.70	771.35	48.83	751.31	796.40	38.28

*Bold numbers are the highest

Table 1: A summary of the utilities for 120 independent runs.

We have n surrogates because there are n possible utility functions. Each surrogate corresponds to a possible utility function.

The surrogates are trained using the given possible utility functions. Thus, the first surrogate is trained using the first possible utility function, the second surrogate is trained using the second possible utility function, and so on for all the n surrogates.

Initially, for a given utility function, a *training set* is created by running the inner GA without the surrogate. A training set is comprised of a set of agendas and the associated utilities/fitnesses. This set is then used to train the surrogate. Once the initial training phase is over, the surrogate is can be used to *predict* the fitness of an agenda without using the inner GA. We do this initial training for all the n surrogates.

Once a surrogate is trained for a given utility function, it can be used instead of the inner GA. However, although we know the possible utility functions, we do not know exactly which one of these it will actually be. Thus, the problem is to choose the right surrogate (from n available surrogates) without knowing the actual utility function.

At the beginning, the system first randomly selects any surrogate out of n available ones and uses it instead of the inner GA. If the fitness predicted by this surrogate is higher than the fitness of all the elements in the training set, then[1] we evaluate the true fitness using the inner GA and the current utility function. If this true fitness is higher than fitness of all the elements in the training set, then we consider that the current surrogate is the right one to use with the current utility function. Otherwise, if the true fitness is not better than the best fitness in the training set, then we consider that this surrogate is not the right one for the given utility function. In this case, we apply all the n surrogates to predict the fitness, and choose the surrogate with the lowest *prediction error*. The selected surrogate will be used in the next prediction unless it the system decides to change it.

3. EVALUATION

We compared the performance of the proposed system with a standard GA (i.e., without the assistance of a surrogate) and a random search method. The comparison with a standard GA (i.e., without surrogate assistance) will allow us to check whether the proposed surrogate system can actually explore the search space better than a standard GA search for our negotiation application. The comparison with a random search method will show that the proposed surrogate model is better than a random search. Here, it is important to note that, in random search, the agendas are generated randomly but their utility/fitness is fully evaluated by running the inner GA.

The inner GA is comprised of a population of 1000 individuals and is run through 100 generations. For the inner GA, tournament selection is used with 10 as the tournament size. The outer GA is comprised of m individuals and runs through g generations. For the outer GA, we used *mutation* as the search operator. In order to produce syntactically correct offspring, we ensure that the mutation operator always maintains g ones in every agenda.

Basically, the negotiation utility function can be any $U_i : \mathbf{R}^g \rightarrow \mathbf{R}$. Here, we used three possible functions for U_i: the Rastrigin function, the Dixon and Price function and, the Michalewics function [3]. These three functions keep switching every $G = 3$ generations of the outer GA (in non-acyclic order).

We performed 120 independent runs where the runs were divided into four sets. Each set involved testing the model to find an optimal agenda of size $m = 20, 30, 40$ and 50 and the associated negotiation strategy. For each m, we used $g = m/5, m/2$ and $m/2 + 5$. For each m and g, we performed 10 independent runs.

Both comparison algorithms were given exactly the same number of expensive evaluations (i.e., evaluations done using the inner GA without surrogates) and the same inner GA. In the standard GA, we used a population of size of m and ran it for g generations. The tournament size was 2, and the mutation operator was used 100% of the time. Each individual in the population invoked the inner GA engine to evaluate its fitness. In the standard GA, the system switches utility every $G = 3$ generations. These changes act as a noise on the evolutionary process and prevent the GA from converging to an optimal solution. Results are presented in Table 1. The standard GA comes in the second place in almost all the cases.

4. CONCLUSION

This paper addressed the problem of finding optimal agendas and negotiation strategies for environments where utility functions are not fully known. The utility function can be defined in one of n possible ways. To solve this problem, we proposed a *multi surrogate-based GA system* to evolve optimal agendas and strategies for package deal negotiation. We compared this approach with a standard GA and random search.

5. REFERENCES

[1] Shaheen Fatima and Ahmed Kattan, “Evolving optimal agendas for package deal negotiation”, in *GECCO ’11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, Dublin, Ireland, 12-16 July 2011, pp. 505–512, ACM.

[2] Adrian G. Bors, “Introduction of the radial basis function (rbf) networks”, Tech. Rep., Department of Computer Science, University of York, UK, 2001.

[3] Marcin Molga and Czeslaw Smutnick, “Test functions for optimization needs”, *Test functions for optimization needs*, 2005.

[1] If the fitness predicted by this surrogate is not higher than the fitness of all the elements in the training set, then we try the next available surrogate, and so on. If the fitness predicted by all the surrogates is not higher than the fitness of all the elements in the training set, then we discard that individual.