Evolving Optimal Agendas for Package Deal Negotiation

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

Ahmed Kattan
Department of Computer Science
Loughborough University
Loughborough LE11 3TU, UK
a.j.kattan@lboro.ac.uk

ABSTRACT

This paper presents a *hyper* GA system to evolve optimal agendas for *package deal negotiation*. The proposed system uses a Surrogate Model based on Radial Basis Function Networks (RBFNs) to speed up the evolution. The negotiation scenario is as follows. There are two negotiators/agents (a and b) and m issues/items available for negotiation. But from these m issues, the agents must choose $g < m$ issues and negotiate on them. The g issues thus chosen form the *agenda*. The agenda is important because the outcome of negotiation depends on it. Furthermore, a and b will, in general, get different utilities/profits from different agendas. Thus, for *competitive negotiation* (i.e., negotiation where each agent wants to maximize its own utility), each agent wants to choose an agenda that maximizes its own profit. However, the problem of determining an agent's optimal agenda is complex, as it requires combinatorial search. To overcome this problem, we present a hyper GA method that uses a Surrogate Model based on Radial Basis Function Networks (RBFNs) to find an agent's optimal agenda. The performance of the proposed method is evaluated experimentally. The results of these experiments demonstrate that the surrogate assisted algorithm, on average, performs better than standard GA and random search.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Game Theory; I.2.8 [**Heuristic Methods**]; I.2.11 [**Intelligent Agents**]; I.2.11 [**Multiagent Systems**]: Electronic Commerce; J.4 [**Economics**]: [Negotiation Agendas]

General Terms

Theory

1. INTRODUCTION

Negotiation is a process in which disputing agents decide how to divide the gains from cooperation between themselves. Since this decision is made jointly by the agents [15], each agent can only obtain what the other is prepared to allow them. The simplest form of negotiation involves two agents and a single-issue. For example, consider a scenario in which a buyer and a seller negotiate on the price of a good. To begin, the two agents are likely to differ on the price at which they believe the trade should take place, but through a process of joint decision-making they either arrive at a price that is mutually acceptable or they fail to reach an agreement. Since agents are likely to begin with different prices, one or both of them must move toward the other, through a series of offers and counter offers, in order to obtain a mutually acceptable outcome.

However, before the agents can actually perform such negotiations, they must decide the rules for making offers and counter offers. These rules are called the negotiation *protocol* or *procedure* [16, 6]. On the basis of this procedure, each agent chooses its *strategy* (i.e., what offers to make during the course of negotiation). For competitive negotiations, which are the focus of this work, each agent chooses a strategy that maximizes its own utility/profit and is therefore its *optimal strategy*. For example, buyer-seller negotiations are competitive in nature. For such negotiations, game theory [14] provides methods for analyzing the strategic behavior of utility maximizing agents. It provides methods for identifying those strategies that are optimal and stable. Strategies that are optimal and stable are said to form an *equilibrium*. There are various notions of equilibrium but the one relevant to our work is Nash equilibrium [14].

Now, in many buyer-seller negotiations, the agents need to settle the price of not one but multiple items. Such negotiations are called multi-issue negotiations [10]. Multiple issues can be negotiated using different *procedures*. These include the *package deal procedure* (PDP), the *sequential procedure* (SQP), and the *simultaneous procedure* (SP). Different procedures are known to result in different outcomes, and the choice of a procedure depends on the characteristics of its outcome. One of the desirable characteristics is *Pareto-efficiency*. Between the PDP, the SQP, and the SP, only the PDP is known to result in Pareto-efficient outcomes. The PDP will therefore be the focus of this work.

For the PDP all the issues are bundled and discussed together as a package [3]. Now, for the PDP, the outcome depends on the set of issues chosen for negotiation. This set is called the *negotiation agenda*. Different agendas yield different profits to the agents [6]. So an agent wants to know what agenda maximizes its profit and is therefore its *optimal agenda*. In many real-world settings, a negotiator has

GECCO'11, July 12–16, 2011, Dublin, Ireland.

the option of choosing an agenda. For example, consider a car dealer who has m cars to sell. A potential buyer may be interested in buying $g < m$ of these. So the buyer must first choose which cars to negotiate the price for (i.e., from all possible subsets of size g, the buyer must choose the one that maximises its utility). Note that here, the buyer has choice over the agenda but the seller does not. Given this, our goal is to determine what agenda will be optimal from the perspective of an individual agent (i.e., the buyer or the seller). We will provide an analysis of this problem form the perspective of the buyer (the analysis from the seller's perspective will be analogous).

In more detail, there are $C(m, g)$ possible agendas of size g. In order to find which of these is optimal for the buyer, we need to determine the buyer's equilibrium profit for each possible agenda and then choose the one that yields highest profit. Thus, we have two problems to solve:

P For a given agenda, determine the buyer's equilibrium utility. This is a constrained nonlinear optimization problem.

Q Search the space of $C(m, g)$ possible agendas and choose the one that yields highest equilibrium utility to the buyer. Here the size of search space is combinatorial.

Thus, both P and Q are computationally complex problems. Hence we need a solution method that is computationally feasible. To this end, we propose a *hyper* GA system to solve P and Q. The proposed system uses a Surrogate Model based on Radial Basis Function Networks (RBFNs) to evolve optimal agendas. The system is comprised of two GA systems: an *outer GA* and an *inner GA*. The inner GA solves the problem P while the outer GA solves Q. The outer GA is assisted by a surrogate model based on Radial Basis Function Networks (RBFNs) (Sections 5 and 4 provide details). In the course of evolution, the surrogate's role is to point to the most promising agendas, and thereby speed up the process of evolutionary search.

In order to evaluate the effectiveness of the proposed method, we experimentally compared its performance with that of two other methods: standard GA and random search. This comparison was done on the basis of an agent's profits from the optimal agendas generated by these methods. The results of these experiments demonstrate that the proposed surrogate assisted algorithm, on average, performs better that a standard genetic algorithm and random search.

This paper makes the following main contribution. We present a new method for determining an agent's optimal agenda and the optimal allocations for the agenda, for the PDP. Most of the existing work on negotiation has taken the agenda as given, and dealt with finding effective methods for determining the equilibrium. However, the agenda is a key negotiation parameter and it is crucial in determining the outcome of negotiation. Thus, from the perspective of competitive agents, it is important not just to optimally negotiate over a given set of issues, but also to choose the best agenda before negotiation begins. The proposed method allows agents to perform both these tasks.

The rest of this paper is organised as follows. Section 2 discusses previous work related to this research. Section 4 provides background on RBFNs and its mathematical notation. In Section 5, a detailed description of the proposed model is given. Section 6 describes the experimental evaluation of the proposed model, and Section 7 provides an analysis of the results. Finally, Section 8 draws conclusions.

2. RELATED LITERATURE

We first discuss related work for negotiation and optimal agendas, and then for surrogate models.

2.1 Optimal Agendas

Negotiation has long been studied by game theorists. However, in this work, the analysis of negotiation typically begins with a given set of issues and the parties' utilities for different possible settlements of the issues. Within this framework, theorists have investigated a range of procedures such as the PDP, the SP, and the SQP [6] and shown that different procedures yield different outcomes. Hence, it is important to choose the right procedure. Furthermore, irrespective of the procedure, it is important to choose the right agenda.

Although the importance of agendas has been recognised, most existing work has taken the set of issues as given and analysed the equilibrium for different procedures. For instance, [6, 1, 7] takes the set of issues as given and shows that the order in which they are negotiated is important in determining the outcome. The problem of determining optimal agendas for the PDP was addressed in [5], but in the context of linear utilities. In contrast with [5], the focus of this paper is on non-linear utility functions.

2.2 Surrogate Models

When the objective functions are expensive to evaluate, a single optimisation case can take a very long time and make the optimisation process infeasible. Furthermore, optimization problems are, in many cases, black-box problems, i.e., whose problem class is unknown, and they are possibly mathematically ill-behaved (e.g., discontinuous, non-linear, non-convex). Possible ways of dealing with such optimization problems, include the use of use a high-performance computing technology with multi-threading programming, or an approximation model that approximates a given objective function. Surrogate models are approximation models that have been employed to tackle expensive objective functions. In these models, some of commonly used approximations include, Polynomial Regression (PR), Artificial Neural Networks (ANN), Radial Basis Function Networks (RBFNs) and Support Vector Machines (SVM) [8]. Existing work on surrogate models includes the following.

Lim *et al.* in [12], proposed a generalised surrogate-assisted evolutionary frameworks for optimisation of problems that are computationally expensive to evaluate. The authors introduced the idea of employing several on-line locale surrogate models which are constructed using data points that lie in the vicinity of an initial guess. The improved solutions generated by the local search surrogates are used to replace the original individual. In this work, the framework has been presented with single objective optimisation and multi-objective optimisation.

In [11], proposed an enhancement for GA by using local surrogate search to expedite convergence of GA. The model uses GA to generate a population of individuals and rank them with the real function. Afterwards, gradient-based local search is performed on the surrogate model to find new promising solutions. The GA and local search are alternatively used under a trust-region framework until optimum found. The trust-region framework is used to assure that

the surrogate's solutions are converging toward the original problem.

Recently in [13], Moraglio and Kattan showed that surrogate models can be naturally generalised to encompass combinatorial spaces based in principle on any arbitrarily complex underlying solution representation by generalising their geometric interpretation from continuous to general metric spaces. An illustrative example is given related to Radial Basis Function Networks (RBFNs), which can be used successfully as surrogate models to optimise combinatorial problems defined on the Hamming space associated with binary strings. The authors illustrated the methodology with NK-landscape problem.

Traditional surrogate model based optimisation (SMBO) [9, 13, 11, 12, 8] uses generational evolutionary procedure to infer the location of a promising solution (with the assumption that its cost is negligible in comparison to the expensive objective function). In contrast, in the proposed surrogate model, we use a local search process on the surrogate training set (details in Section 5). This local search uses characteristics specific to the problem, in order to speed up the search and to generate better solutions.

3. THE NEGOTIATION MODEL

As mentioned in Section 1, our aim is to effectively solve problems P and Q. In this section, we define the problem P in more detail. This problem requires determining the buyer's/seller's equilibrium utility for a given agenda. Thus, we have two versions of problem P: one for the buyer denoted P_b, and another for the seller denoted P_a. Now, an agent's equilibrium utility for a given agenda depends on the negotiation setting, which is defined as follows.

There are two negotiating agents called a (a seller) and b (a buyer). There is a set $I = \{1, 2, \ldots, m\}$ of m issues/items. The agents are negotiating the price of these items. Each issue is represented as a divisible 'pie' of size one [14]. So the agents are negotiating about how to split each pie between themselves. If we use x_i^a and x_i^b (where $x_i^a \in [0, 1]$ and $x_i^b \in [0, 1]$) to denote a's and b's shares for issue i respectively, then we have the following:

$$x_i^a + x_i^b = 1.$$

The issues are negotiated using a two-round PDP. This procedure is an alternating offers protocol [14] in which one of the agents, say a, starts in the first round ($t = 1$) by offering x^a (where $x_i^a \in [0, 1]$) to b.

Here, x^a denotes a vector that specifies an offer for each of the m issues. Agent b can accept/reject the offer. If it accepts, negotiation ends in an agreement with a getting x^a and b getting $x^b = 1 - x^a$. Otherwise, it goes to the round two ($t = 2$), when b makes an offer. If a accepts this, negotiation ends successfully in an agreement. Otherwise, it ends in a conflict and both agents get zero utility.

Note that an agent is allowed to either accept a complete offer (i.e., the allocations for all the issues) or reject a complete offer. It cannot accept/ reject part of an offer.

For time $t \leq 2$, agent a's cumulative utility/profit from x^a is defined as follows:

$$U^a(x^a, t) = \delta^{t-1} \sum_{i=1}^{m} C_i^a x_i^a \quad (1)$$

where $C_i^a \in R_+$ are real valued constants, and $0 \leq \delta \leq 1$ is the discount factor. The discount factor is a constant that indicates the rate at which utility gets discounted with time. Agent b's cumulative utility/profit is defined with the Rastrigin function as follows:

$$U^b(x^b, t) = \delta^{t-1} \times \left(36g - \sum_{i=1}^{g} C_i^b [(10.24 x_i^b - 5.12)^2 - 10 \cos(2\pi(10.24 x_i^b - 5.12))] \right) \quad (2)$$

where $C_i^b \in R_+$ are real valued constants.

The reason for choosing the Rastrigin function is that it represents a nonlinear landscape with a high degree of ruggedness (i.e., it has the key features of a typical nonlinear utility function). Moreover, the ruggedness can be varied by suitably varying the parameters of this function. Figure 1 shows the landscape for a Rastrigin function in two variables (i.e., two issues for which b's shares are x_1^b and x_2^b).

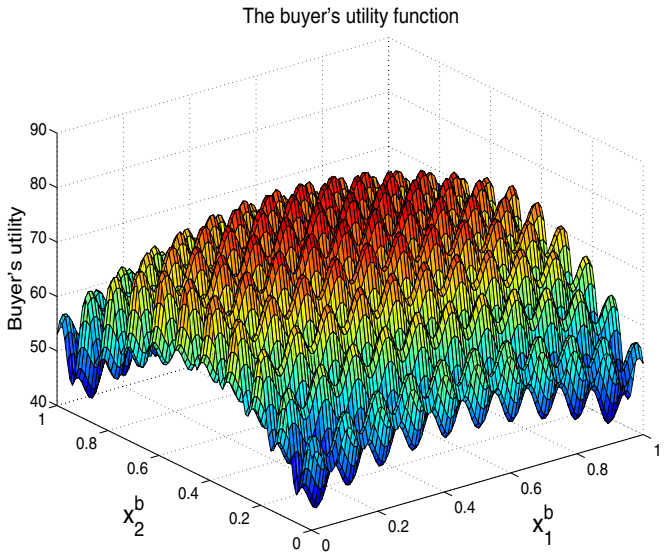

Figure 1: Landscape for the Rastrigin function.

Since it is a two-round PDP, an agent's utility for $t > 2$ is zero. For this model, the equilibrium was given in [4]. Below, we give a brief overview of this equilibrium. We will describe the equilibrium for negotiation over all the m issues. The equilibrium for a $g < m$ can easily be obtained from this.

Let $\text{SA}(I, t)$ ($\text{SB}(I, t)$) denote a's (b's) equilibrium strategy for time t for the issues in I. At $t = 2$, the offering agent proposes to keep a 100% of all the pies and the other agent accepts [4]. In the previous round, $t = 1$, the offering agent (say b) offers (x^a, x^b) such that a's cumulative utility from it is what a would get from its own offer for $t = 2$. If there is more than one such (x^a, x^b), then b must choose the one that maximizes its own utility.

If we let Q_t^a (Q_t^b) denote a's (b's) equilibrium utility for t,

and $\mathbf{0}$ ($\mathbf{1}$) denote a vector of m zeros (ones) we get:

$$Q_2^a = \delta \sum_{i=1}^{m} C_i^a$$

and

$$Q_2^b = U^b(\mathbf{1}, 1).$$

So b must solve the following trade-off problem (called $P_b(I,1)$) at $t = 1$:

$$P_b(I,1): \quad \text{MAX} \quad U^b(x^b, 1)$$
$$\text{s.t.} \quad U^a(x^a, 1) \geq Q_2^a \qquad x_i^a \in [0,1];\ x_i^b = 1 - x_i^a$$

For agent a, the trade-off problem is defined as follows:

$$P_a(I,1): \quad \text{MAX} \quad U^a(x^a, 1)$$
$$\text{s.t.} \quad U^b(x^b, 1) \geq Q_2^b \qquad x_i^a \in [0,1];\ x_i^b = 1 - x_i^a$$

Both P_a and P_b are nonlinear optimization problems. The agents' equilibrium strategies are defined in terms of P_a and P_b as follows:

$$\text{SB}(I,2) = \begin{cases} \text{OFFER} \ \ (\mathbf{1},\ \mathbf{0}) & \text{If } b\text{'s turn to offer} \\ \text{ACCEPT} & \text{If } b\text{'s turn to receive} \end{cases}$$

For $t = 1$, the strategies are:

$$\text{SB}(I,1) = \begin{cases} \text{OFFER} \ \ P_b(I,1) & \text{If } b\text{'s turn to offer} \\ \text{If} \ \ (U^b(x^b,1) \geq Q_2^b) & \text{If } b \text{ receives } (x^a, x^b) \\ \text{ACCEPT else REJECT} & \end{cases}$$

Agent a's strategy (SA) is defined analogously in terms of P_a. The above strategies form a Nash equilibrium and result in an agreement at $t = 1$.

Given the above equilibrium, we can determine each agent's utility for a given agenda, and, on the basis os these, determine what agenda will maximize b's utility.

Before closing this section, we will formally define the terms *agenda* and *optimal agenda*.

3.1 The Negotiation Agenda

The term agenda is defined as follows:

DEFINITION 1. ***Agenda**: Given the set I of m issues and an integer $g \leq m$, an agenda A^g is a set of g issues, i.e., $A^g \subseteq I$ where $|A^g| = g$.*

Let AG^g denote the set of all possible agendas of size g. Then, an agent's optimal agenda is defined as follows:

DEFINITION 2. ***Optimal agenda**: Given the set I of m issues and an integer $g \leq m$, an agenda (AB^g) is agent b's optimal agenda if:*

$$AB^g = \underset{X \in AG^g}{arg\ max} \ \ U^b(\text{SB}(X,1), 1).$$

An optimal agenda for a is defined analogously.

We showed how to find equilibrium for the set of m issues in I. Given this equilibrium, the problem Q is to find AB^g. Sections 4 and 5 describe our approach for solving the problems P and Q.

4. RADIAL BASIS FUNCTION NETWORKS

The evaluation of the functions in P_a and P_b (described in Section 3) requires an optimisation process which may be time consuming. Thus, instead of evaluating these functions while exploring the space of possible agendas $C(m,g)$, we want to replace it with a surrogate model that approximates U^b. If U^b approximation is promising, then we pass it to the real function described in Section 3 to calculate its real value.

There are a number of known approaches to learn a function belonging to a certain class of functions from existing data-points (i.e., finding a function in that class that interpolates and best fits the data-points according to some criteria). These include Genetic Programming, Radial Basis Function Network Interpolation, and Gaussian Process Regression.

Genetic programming is very powerful framework for approximating unknown functions. However, its direct implementation is not suitable to be used as surrogate because of its expensive learning process. Gaussian Process Regression has been often used as an approximation model with a solid theoretical foundation, which not only can make a rational extrapolation about the location of the global optimum, but also gives an interval of confidence about the prediction made. Radial Basis Function Network Interpolation is conceptually simpler than Gaussian Process Regression and can extrapolate the global optimum from the known data-points. In this paper, we use the RBFNs because of its simplicity and effectiveness [2]. As we will show in the section on experiments, this approach works well. The following sub-sections provide a description RBFNs and how we use them.

4.1 RBFN Representation

RBFNs are a variant of artificial neural network that uses radial basis functions as activation functions [2]. They have been used in function approximation, time series prediction, and control [2]. A radial basis function (RBF) is a real-valued function of the following form:

$$\phi : \mathbf{R}^n \rightarrow \mathbf{R}$$

whose value depends on the distance from some point c, called a *center*, so that

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x_q} - \mathbf{c}\|).$$

The point c is a parameter of the function and the point x_q is the query point to be estimated. The norm is usually Euclidean, so $\|\mathbf{x} - \mathbf{c}\|$ is the Euclidean distance (radius) between c and x. Since we use a generalised RBFN [13], the Euclidean distance has been replaced with a metric distance that naturally encompasses the GA representation of our optimisation problems P_a and P_b (see Section 4.3). The most commonly used types of radial basis functions are the Gaussian functions of the form

$$\phi(x) = \exp(-\beta \|\mathbf{x} - \mathbf{c}\|^2)$$

where $\beta > 0$ is the width parameter. Radial basis functions are typically used to build function approximations of the form:

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{N} w_i\, \phi(\|\mathbf{x} - \mathbf{c}_i\|) \qquad (3)$$

Thus, $y(x)$ is used to approximate U^b. The approximating function $y(x)$ is represented as a sum of N radial basis functions, each associated with a different center c_i, a different width β_i, and different weight w_i, plus a bias term w_0. In principle, any continuous function can be approximated with arbitrary accuracy by a sum of this form, if a sufficiently large number N of radial basis functions is used. The bias w_0 is set to the mean of the values of the known data-points from the training set that are used to train the surrogate model, or set to 0.

4.2 Training

Training the RBFNs amounts to find three parameters:

i) the centres $\mathbf{c}_i$,

ii) the weights w_i, and

iii) the RBF width parameters β_i.

These parameters must be found in such a way that the predictions on the training set minimises any errors. We choose the centers so as to coincide with the known data-points. The values of β can either fixed for all N linear RBFs (global), or they may be different for different RBFs (local). Here, we use local values for β for each RBF. Specifically, β is $1/D^2$ where D is the mean pairwise swap distance between the query data-point and its n closest neighbours in the training set.[1]

The value of β controls the radius of each RBF to spread on the space to cover all other centres so that each known function value at a center can potentially contribute significantly to the prediction of the function value of any point in space, and not only locally to function values of points near the given center. In order to find the best value for n, we conducted experiments. The results of these experiments showed that the best value of n is 40% of the size of the training set.

Finally, the weights vector is calculated by solving the system of N simultaneous linear equations in w_i obtained by requiring that the unknown function interpolates exactly the known data-points

$$y(\mathbf{x}_i) = b_i,\ i = 1 \ldots N$$

By setting

$$g_{ij} = \phi(||\mathbf{x}_j - \mathbf{x}_i||),$$

the system can be written in matrix form as follows:

$$\mathbf{Gw} = \mathbf{b}$$

where $\mathbf{b}$ is a vector of the true fitness values of the data-points that are used to train the surrogate. The matrix $\mathbf{G}$ is non-singular if the points $\mathbf{x}_i$ are distinct and the family of functions ϕ is positive definite (which is the case for Gaussian functions). Thus solving $\mathbf{w} = \mathbf{G}^{-1}\mathbf{b}$ gives the weights w.

The value of the bias term w_0 in Equation 3 is set to the mean value of the known data-points, i.e., the mean of vector $\mathbf{b}$. So the predicted function value of a point which is out of the influence of all centres, is by default set to the average of their function values.

[1]The swap distance between two strings is the minimum number of interchanges needed to transform one string to the other.

4.3 Interpolation

The RBFNs can be naturally generalised from continuous spaces to any representation [13]. This is done by considering distances defined directly on the underlying representation. The generalisation is possible because the representation of the RBFN, its training, and prediction does not depend directly on the representation, but only on the distances between solutions [13]. Once the parameters of the RBF are determined, the model is ready to estimate the fitness of any unseen point. Thus, the fitness $f(x)$ of unknown point x_q in the search space is predicted by weighted linear combination of:

$$f(x) = w_0 + \sum_{i=1}^{N}[w_i * \phi(d(\mathbf{x_q}, \mathbf{c}_i))]$$

where, w_i is a vector of weights that are calculated during the training phase and ϕ is the kernel function which is defined in Section 4.1. Finally, $d(\mathbf{x}, \mathbf{c}_i)$ is the swap distance between the new point x_q and the training points c_i.

Traditionally, Hamming distance is used to measure the distance associate between binary strings. However, the reason we use the (adjacent) swap distance instead of the Hamming distance is that the former is more natural to binary strings constrained to have the same number of bits set to one (i.e., the number of issues on the agenda).

5. THE SURROGATE-ASSISTED GA

As mentioned previously, we have two problems to solve: find the equilibrium utility for an agenda, i.e., solve the problem P_b (as defined in Section 3), and finding an optimal agenda, i.e., solve problem Q (as defined in Section 1).

To solve these problems, the proposed surrogate assisted GA uses a hyper GA system. This system is comprised of two GA systems, one to solve the problem P_b, and the other to solve the problem Q. These two GA systems are as follows:

- an 'outer' GA to solve the problem Q, and
- an 'inner' GA to solve the problem P_b.

These two GAs work as follows. The *outer* GA searches the space of possible $C(m, g)$ agendas to solve Q. Each agenda is represented as a binary string. A one in the string indicates that the corresponding issues is included in the agenda. A zero means the issue is not on the agenda. Since there are m available issues, the size of a chromosome is equal to m. Also, since we want to find an optimal agenda of size g, a chromosome can have only g ones. Thus, all individuals in the population must have the same number of ones.

Then, we have the *inner* GA to solve P_b. For a given individual in the outer GA population, the inner GA optimises U^b, i.e., finds b's equilibrium utility. In other words, the inner GA serves as a fitness evaluator for the outer GA, and allows the individuals for the outer GA to be ranked on the basis of their utilities/fitness. An individual for the inner GA is a vector of m real numbers in the interval $[0, 1]$. The element i represents x_i^b (where x_i^b is as defined in Section 3).

Clearly, the hyper GA method is expensive as it requires each individual in the outer GA population to invoke an inner GA run to evaluate its fitness. Also, due to the noisy nature of evaluating the population (i.e., evaluating the same agenda twice with the inner GA may produce a slightly different results), and hence the noisy nature of the landscape

it is not always easy to get an optimum (or more precisely near optimum) solutions. To solve this problem, we used the surrogate model based on RBFNs (described in Section 4) to speed up the outer search.

The surrogate model is built solely from available known values of the expensive objective function evaluated on a set of solutions.[2] We refer to the pair (solution, known objective function value) as data-point. Thus, data-points form a sample of the expensive objective function and the surrogate model builds a function that approximates these data-points.

The outer GA, assisted by the surrogate model, explores as many agendas as possible and ranks them based on their predicted fitness using the surrogate model. The most promising data-points (i.e., the points that have better estimated fitness than the best known data-point) are re-examined through the inner GA evaluation to calculate their real profits, update the list of existing data-points, and update the surrogate model (as described in Section 4.2).

Algorithm 1: Surrogate-Assisted GA search.

```
1  Surrogate-TrainingSet =
   Generate-Solution(initial-set-size);
2  Evaluate-Inner-GA(SampleDataPonits, U^b);
3  Surrogate.Train(Surrogate-TrainingSet);
4  while Expensive-evaluation-budget Not Finished do
5      Promising-Point =
       Tournament-selection(Surrogate-TrainingSet);
6      Mutation(Promising-Point);
7      Surrogate.Predict(Promising-Point);
8      if Promising-Point >
       Best-item(Surrogate-TrainingSet) then
9          Evaluate-Inner-GA(SampleDataPonits, U^b);
10         Surrogate-TrainingSet.add(Promising-Point);
11         Surrogate.Train(Surrogate-TrainingSet);
12     if Maximum exploration elapsed then
13         Generate a new random data-point and
           update the surrogate.
```

The surrogate's search procedure is described in Algorithm 1. To begin, some initial solutions are generated using the real objective function. These form a sample of data-points that are used to train the surrogate model as explained in Section 4.2 (see Lines 1-3 in Algorithm 1). Once the surrogate model is trained, it becomes ready to the predict fitness of unseen points in the search space. The system applies a simple mutation operator on a selected data-point from the surrogate's training set using Tournament selection (we refer to this process as *local search*) (see Lines 5-6). The newly mutated point is then evaluated using cheap surrogate evaluation as explained in section 4.3 to obtain its estimated fitness (Line 7).

Note that the role of the local search procedure is to infer the location of a promising solution of the problem using the surrogate model, and not to directly apply it to the original problem with the expensive objective function. Also, unlike other surrogate models where number of explorations is limited, our model has the freedom to explore as many solutions as it requires until it finds promising solutions.

If the predicted fitness value of the mutated point is better than the best known fitness value of the known data-points, then it means that the model succeeded in extrapolating from the data (see Line 8). Otherwise, it means that the mutation operator failed at suggesting a promising solution which improves over the best known point. If this happens, we repeat the tournament selection and mutation process. If the prediction is higher than the best point in the training set, then the point is promising, and we evaluate it with the real fitness function (Line 9). Thereafter, the new promising point, with its true fitness, is added to the sample of data points (Line 10). The resulting sample is then used to re-train the surrogate model (Line 11).

To avoid the system from getting caught at a local optima, the surrogate training set is updated at a uniformly generated random data-point and evaluated with the expensive objective function. This allows us to gather more data about under-sampled regions of the problem and improve the accuracy of the surrogate model to help subsequent searches on the model (Lines 12-13).

The outer GA, is designed to operate in conjunction with a surrogate model. Thus, the training set is considered as the GA population. The outer search process keeps increasing the population size (i.e., adding new promising data-points) until the maximum number of expensive evaluations has been reached. The use of tournament selection and mutation operator allows the newly added data-points to converge toward optimum fitness.

6. EXPERIMENTAL ANALYSIS

The aim of the experiments is to evaluate the performance of the proposed surrogate assisted GA model in a range of settings. This requires determining 'how optimal' are the agendas generated by this model. Here, we look at optimal agendas from the buyers perspective (the same analysis applies to the seller as well). The higher the buyer's utility for an agenda, the better the agenda is.

In order to evaluate the proposed model, we compared its performance with two other models:

- a standard GA (i.e., without surrogate assistance), and
- a random search.

The reason we included random search is that, although evolutionary algorithms may generally be better than random search, they may not be so for small search spaces. It should be noted that, for the random search, the agendas are generated randomly, but their profit is fully evaluated with the inner GA.

6.1 The Setting

The following experiments were conducted for $\delta = 0.5$ assuming that both negotiators have complete information about the negotiation parameters. The remaining parameters are set as shown in Table 1. Recall that the surrogate model uses the RBFNs to fit the available data-points using the learning procedure described in Section 4.2. So the parameters were set partly by using values commonly found in the surrogate literature, and by performing a variety of

[2]We refer to the inner GA evaluation as expensive evaluation because it requires a complete GA run to evaluate the fitness of a single individual in the outer search.

Table 1: Surrogate Parameter Setting

Setting	Value
Expensive Evaluation	$m \times g$
Initial sample size	2
Tournament selection	50% of Training Set size
θ Explorations	$(g \times m)^2$

Table 2: Inner GA Parameter Setting

Setting	Value
Population	1000
Generations	100
Tournament selection size	10
Mutation	100%

preliminary experiments and selecting the values that gave us good results while keeping the processing time under control.

Our aim is to find the best solution (i.e., an optimal agenda) in linear time with respect to m and g. As mentioned earlier, θ is the maximum number of explorations for detecting that no further improvement is happening. If this happens, we update the model with a randomly generated data-point. Thus, θ is also related to the problem complexity with respect to the number of possibilities to mutate data-points in the training set. The use of tournament selection with high pressure (i.e., 50%) allows the system to favour the best data-points in the training set to locate new promising solutions. And training points with inferior fitness still have a chance to contribute to the process of finding new promising points.

The performance of the model was evaluated for 120 independent runs. The runs were divided into four sets. Each set involved finding the best agenda for $m = 10$, $m = 20$, $m = 30$, and $m = 40$. For each m, three different values of g, where $g = m/5$, $g = m/2$, and $g = m - 3$ were used. For each m and g, we conducted 10 independent runs.

Recall that we use standard GA and random search for comparison. To allow a fair comparison, the standard GA was applied directly to the problem with the expensive objective function. Also, the the standard GA and the random search were given exactly the same number of expensive evaluations and the inner GA engine (see inner GA settings in Table 2).

In standard GA, the GA has a population of size m and runs for g generations. We used tournament selection of size 2 and a mutation operator. Each individual in the population invokes the inner GA engine to evaluate its fitness.

The reason for conducting 10 independent runs for each m and g, is that the experiments are very time consuming. So, we balanced the experiments to show enough information about the performance of the model while keeping the whole process under control. For this setting, Section 7 shows the best evolved agendas.

7. RESULTS AND ANALYSIS

Table 3 summarises the results of experiments. For each m and g, the table illustrates the average profit for an agenda over 10 independent runs. These results show that the surrogate produced better agendas in all cases. Also, the profit difference increased with the problem complexity. Thus, surrogate performs better with complex problems (i.e., those with large m and g) than easier ones (i.e., those with small m and g).

Table 3: The average profit (over 10 runs) for each m,g combination.

m	g	Surrogate	Standard GA	Random Search
10	2	**65.8**	63.9	64.5
10	5	**168.7**	166.3	165.4
10	7	**209.6**	209.1	209.1
20	4	**137.9**	130.5	132.6
20	10	**309.9**	299.4	297.6
20	17	**452.8**	432.6	433.4
30	6	**204.3**	197.6	193.6
30	15	**460.9**	445.8	435.2
30	27	**750**	739.05	738.1
40	8	**275.4**	262.2	258.1
40	20	**617.8**	596.7	577.2
40	37	**1070**	1024.5	1023.4

*Numbers in **boldface** represent the highest generated profits.

Table 4: The average improvement (over 10 runs) for each m,g combination.

m	g	Average Improvement	Best Improvement
10	2	0.54%	4.80%
10	5	0.98%	1.80%
10	7	0.05%	0.5%
20	4	3.69%	7.10%
20	10	2.39%	4.90%
20	17	4.40%	9.90%
30	6	3.17%	5.40%
30	15	3.39%	4.90%
30	27	1.40%	1.60%
40	8	4.63%	7.40%
40	20	3.42%	5.30%
40	37	4.40%	4.50%
Average		**2.7%**	**4.8 %**

This is further illustrated in Table 4. This table shows the best in all 10 runs for each m, g combination. For $m = 10$, random search produced better or similar agendas relative to the standard GA and the surrogate. This is no surprise, because standard GA can get easily trapped in local optima, whereas the solution found by random search exhibits a large variance in quality so the best solution found can be competitive by a "stroke of luck", especially with small sample size and in small problems (such as $m = 10$). However, as shown in Tables 3 and 4, on average, the surrogate always comes in the first place while standard GA and random search take the second and third places respectively.

The reason why the GA assisted by the surrogate is better is that it is able to infer promising solutions by making rational use of the knowledge of the location and the real fitness of previously sampled solutions.

As mentioned previously, the system has the freedom to explore as many agendas as it requires, before it updates the model with new promising point, or alternatively, with a randomly generated point if no promising point can be found. Thus, if the system updates the model at random

Table 5: The average number of random updates for the surrogate for each m and g combination.

m	g	Random Updates
10	2	8.5%
10	5	13.4%
10	7	15.7%
20	4	0.12%
20	10	0.6%
20	17	0%
30	6	0%
30	15	0%
30	27	0.04%
40	8	0.03%
40	20	0%
40	37	0%
Average		**3.19%**

*The table shows the average over 10 runs for each m and g combination.

points on a large norm, the whole process turns into something close to random search. Note that, for small spaces, this may not necessarily be bad. Thus, in Table 5, we show the number of times (in terms of the percentage over the total number of expensive evaluations budget) the system updates the model at random points.

As shown, when the problem space is small (e.g., $m = 10$) the system injects the surrogate model with some random samples to have a better approximation. However, for bigger search spaces, only in rare cases the system updates the model at random points. In fact, in 97 out of the total 120 runs the system did not add any random points to update the surrogate. This indicates that the local search process guided by the mutation operator and tournament selection (see Section 5) did a good job at pointing out the most promising solutions.

8. CONCLUSIONS

This paper proposed a hyper GA system to evolve optimal agendas for package deal negotiation. The agenda is important in the context of negotiation because it effects the agents' profits. Thus, the agents want to know what agenda will optimize their profit. But the problem of finding such an agenda is complex, especially in the context of nonlinear utilities. To solve this problem, we presented a hyper GA system that uses a surrogate model based on Radial Basis Function Networks (RBFNs) to assist the GA search. The model speeds up the search by guiding the evolution in the direction of promising agendas.

The method was evaluated experimentally. The results demonstrate that the proposed model, on average, finds better agendas in comparison with a standard GA (applied directly without surrogate assistance), and random search.

This paper has made two main contributions: proposed a new method for finding an optimal negotiation agenda, and showed that it works well in the context of a real-world problem.

This research can be extended in many different ways. In the future we will explore ways of extending the model to those situations where the utility functions change with time. Another extension would be to test the proposed system with other optimization problems.

Acknowledgements

This research was supported by the EPSRC under grant EP/G000980/1.

9. REFERENCES

[1] M. Bac and H. Raff. Issue-by-issue negotiations: the role of information and time preference. *Games and Economic Behavior*, 13:125–134, 1996.

[2] A. G. Bors. Introduction of the Radial Basis Function (RBF) networks. Technical report, Department of Computer Science, University of York, UK, 2001.

[3] S. Fatima, M. Wooldridge, and N. Jennings. Multi-issue negotiation with deadlines. *Journal of AI Research*, 27:381–417, 2006.

[4] S. S. Fatima, M. Wooldridge, and N. R. Jennings. Approximate and online multi-issue negotiation. In *Proc. 6th Int. J. Conference on Autonomous Agents and Multi-agent Systems*, pages 947–954, 2007.

[5] S. S. Fatima, M. Wooldridge, and N. R. Jennings. On optimal agendas for multi-issue negotiation. In *Proc. 12th Int Workshop on Agent-Mediated Electronic Commerce*, pages 155–168, 2010.

[6] C. Fershtman. The importance of the agenda in bargaining. *Games and Economic Behavior*, 2(3):224–238, 1990.

[7] R. Inderst. Multi-issue bargaining with endogenous agenda. *Games and Economic Behavior*, 30:64–82, 2000.

[8] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.*, 9(1):3–12, 2005.

[9] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21:345–383, December 2001.

[10] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: John Wiley, 1976.

[11] Y. Lian, M. sing Liou, and A. Oyama. An enhanced evolutionary algorithm with a surrogate model. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.130.4692, 2008.

[12] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff. Generalizing surrogate-assisted evolutionary computation. *Evolutionary Computation, IEEE Transactions on*, 14(3):329 –355, 2010.

[13] A. Moraglio and A. Kattan. Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In *EvoCop*, Lecture Notes in Computer Science. Springer, 2011.

[14] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[15] D. G. Pruitt. *Negotiation Behavior*. Academic Press, 1981.

[16] T. Schelling. An essay on bargaining. *American Economic Review*, 46:281–306, 1956.