Multi-Agent Multi-Issue Negotiations with Incomplete Information: A Genetic Algorithm Based on Discrete Surrogate Approach

Ahmed Kattan, Yew-Soon Ong and Edgar Galván-López

Abstract— **In this paper we present a *negotiation agent* based on Genetic Algorithm (GA) and Surrogate Modelling for a multi-player multi-issue negotiation model under incomplete information scenarios to solve a resource-allocation problem. We consider a multi-lateral negotiation protocol by which agents make offers sequentially in consecutive rounds until the deadline is reached. Agents' offers represent suggestions about how to divide the available resources among all agents participating in the negotiation. Each agent may *"Accept"* or *"Reject"* the offers made by its opponents through selecting the "Accept" or "Reject" option. The GA is used to explore the space of offers and surrogates used to model the behaviours of individual opponent agents for enhanced genetic evolution of offers that is agreeable upon all agents. The GA population comprises of solution individuals that are formulated as matrices where a specialised three different search operators that take the matrix representation into considerations are considered. Experimental studies of the proposed negotiation agent under different scenarios demonstrated that the negotiations by the agents completed in agreement before the deadline is reached, while at the same time, maximising profits.**

I. 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, each agent can only obtain what the others are prepared to allow them. In resource-allocation problems, each resource can be envisaged as a pie that all agents are trying to decide how to divide among themselves. Each agent is negotiating a set of resources on the basis of maximising a utility function. This utility function is used as a means for expressing high-level objectives. For example, two agents may have different preferences over a set of resources and, thus, each agent may be interested in maximising its share of particular resources. In some negotiation games, agents may have complete information about each other's utility functions. Hence, each agent optimises its own target with respect to its opponents' objectives in order to end the negotiation in agreement. However, in most real-world scenarios, agents do not disclose information about their utility functions which makes the negotiation problem more challenging (i.e., because of incomplete information). In these situations, each agent needs to make careful offers to other agents as set of resources is to be divided among themselves in such a way to satisfy all other agents and reaches an agreement. Negotiation with incomplete information is a common scenario in many real-world problems since the negotiation decision is non-centralised and jointly made by all agents. For example, in an agent-based negotiation system, a group of software agents will negotiate on behalf of their users trying to reach a common agreement. Each agent is trying to reveal the behaviour of its opponents by observing their interactions and subsequently modelling their behaviours via machine learning techniques, for example [1], [2].

Ahmed Kattan is with the AI Real-World Applications Lab, Computer Science Department, UQU, Saudi Arabia. Yew-Soon Ong is with the School of Computer Engineering, Nanyang Technological University, Singapore. Edgar Galván-López is with the Distributed Systems Group, School of Computer Science and Statistics, Trinity College Dublin. email: ajkattan@uqu.edu.sa, asysong@ntu.edu.sg, and edgar.galvan@scss.tcd.ie.

In most real-world scenarios, agents seldom negotiate on single resource only, but rather over a set of multiple resources. Such negotiations have been established as multi-issue negotiations [3]. In this paper, the term *'issue'* refers to a resource item in the negotiation. Multiple issues can be negotiated using different procedures. These include the *Package Deal Procedure* (PDP), *Sequential Procedure* (SQP), and *Simultaneous Procedure* (SP). However, different procedures are known to result in unique outcomes, and the choice of a procedure would depend on the characteristics of its outcome. One of the desirable characteristics is Pareto-efficiency. Among the PDP, SQP, and SP, only the PDP is known to result in Pareto-efficient outcomes [4]. The PDP, will therefore be the focus of this work. For the PDP, all the issues are bundled and discussed together as a package [4]. Thus, in PDP, either all agents agree on how to divide best the resources among themselves or else the negotiation will not reach an agreement.

Classical game theory negotiation models provide detailed theoretical analysis on the optimal outcomes of a negotiation process, under a predefined negotiation protocol (e.g., see [5] [6]). These analytical studies add valuable knowledge to the boundaries of the negotiation process. However, these normative theories fail to advise the courses of actions that a negotiator may follow to reach this optimal outcome. In this paper, we propose a *negotiation agent* to deal with multiple-agent multiple-issue negotiations under incomplete-information settings. We use a multi-lateral negotiation protocol by which agents make offers sequentially in consecutive rounds until the deadline is reached. Agents' offers represent the suggestions on how to divide the available resources among themselves. For example, as illustrated in Figure 1, for n number of agents negotiating on m number of resources, agents' offers can be formulated as a matrix M of size $n \times m$, where each row contain values in the interval $[0, 1]$ to indicate the share of each agent from a particular resource, while each column denotes an offer to the opponent agents. Note the total values of any row in matrix M is 1. Each agent may then respond to offers given by its opponents in the form of *"Accept"* or *"Reject"*.

Surrogate models, also known as response surface models, are approximation models, that mimic the behaviour of a given model as closely as possible while being fast surrogates for

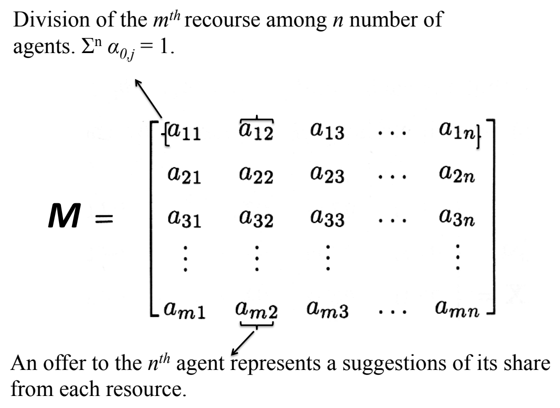
Division of the m^{th} recourse among n number of agents. $\Sigma^n\, \alpha_{0,j} = 1$.
$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$
An offer to the n^{th} agent represents a suggestions of its share from each resource.

Fig. 1. Offer matrix representation. Each row represents the suggested share of each agent for a particular resource. While, each column represents an offer to a particular agent.

time-consuming computation. In a nutshell, surrogate models work by running simulations on a set of points and fitting response surfaces to the resulting input-output data. Some commonly used approximation functions include the Gaussian Process (also called Kriging), Artificial Neural Networks, Radial Basis Function Networks (RBFNs) and Support Vector Machines [7].

Since the main focus of this paper lays on negotiation with incomplete information settings, we used surrogate models for the proposed agent to map offers to their responses. To this end, the proposed agent observes the interaction of it's opponents in order to predict an agreement zone among all agents. For n number of agents, we use $n - 1$ number of surrogates (i.e., one for each opponent) and update the surrogates after each round with new offers and their responses (i.e., Accept or Reject). The proposed agent generates offers for opponents using a Genetic Algorithm (GA) engine to evolve a matrix offer (as represented in Figure 1). The GA is used to explore the offers space in such a way to reach an agreement zone and maximise profit of our agent (we will further explain this in Section IV).

The reason why we used surrogate models is because they have the ability to model an unknown function from small number of sample points. This allows us to explore the offers space of the opponents in such a way to reach superior solutions and derive negotiation strategies that satisfy all negotiators. As will be seen in Section II, most of the existing works for agent-based negotiation systems, under incomplete information settings, used the notion of distance where the main idea is to generate offers that minimise the Euclidean distance to the most recent counter offers given by the opponents. While this seems to be a good way to make offers that satisfy the opponents, however, it focuses the search on a very small area in the offers space (i.e., because it searches for solutions around the most recent counter offer). However, in our work, we use surrogate to model the opponents behaviour and then freely explore their offers space, thus, achieving better solutions.

The structure of this paper is as follows. Section II presents some related work. Section III presents the negotiation protocol used in this work. Section IV provides a detailed description of the proposed negotiation agent. Section V covers an empirical analysis of the proposed agent. Finally, this papers ends in Section VI where some conclusive remarks and future work directions are presented.

II. Related Work

A. Multi-Agent Negotiation

In the past few decades, there has been noticeable development of multi-agent systems and automated negotiation to solve coordination and cooperation for resource allocation problems in complex environments. Lau [2] proposed an adaptive negotiation agent based on GA for multi-agent multi-issue negotiations. The author assumed that agents can change their utility functions during the negotiation process itself. To this end, GA is used to evolve offers that maximise profits, while at the same time, minimises the weighted Euclidean distance between the evolved offer and the most recent counter offers thrown by the opponents.

Rubenstein-Montano et. al. [8] treated multi-agent negotiation as a constrained multi-objective optimisation problem in which they used a GA to evolve offers in the agreement zone. The GA used a weighted sum approach to handle the multiple objectives of each negotiation participant. The authors proposed a customised search operator called *trade*. This trade operator simulates a concession making mechanism that is often used in negotiation systems. Matwin et. al. [9] used a GA to evolve negotiation rules instead of offers for two-agent multi-issue negotiations. These rules represent the offers and counter offers.

Kattan and Fatima [10] proposed a hyper-GA to evolve optimal agendas for bilateral multi-issue sequential negotiation, where the order of issues being negotiated influences the negotiation outcome. The hyper-GA comprised an outer-GA that searches for the optimal set of issues to be included in the agenda and an inner-GA that searches for the optimal order of the selected issues in a manner that increases agents' profits. In addition, Particle Swarm Optimisation is used to balance the computational budget between the two GAs.

To date, the use of surrogate models in agents-based negotiation systems has not been well explored thus far. In [3], the authors presented a hyper-GA system to evolve optimal agendas for package deal two-agent negotiation under complete-information settings. The proposed hyper-GA uses a surrogate model based on Radial Basis Function Networks (RBFNs) to speed up the evolution. The hyper-GA comprises 1) an outer-GA that searches the space of possible agendas and 2) an inner-GA that optimises the shares of each agent from the selected issues (encoded in the evolved agenda) in such a way as to maximise the utility functions of both agents. An RBFN surrogate is used to predict the profits of evolved agendas and reduce the computational costs of the inner-GA. Later, Kattan and Fatima [11] presented a multi-surrogate based GA that deals with a dynamic utility function for two-agent negotiation. The proposed system assumed that the agent's opponent may switch between several pre-defined utility functions during the negotiation process. To this end, several surrogate models are trained for each possible utility function. The authors proposed a choice mechanism to select the most appropriate surrogate model each time the system tries to evolve an offer to reach an agreement zone.

Unlike other works, the proposed agent (for multi-agent multi-issue negotiations under incomplete-information settings) is designed to deal with real-world scenarios in real-time for the Package Deal Procedure. The proposed agent learns the preferences of its opponents through their interactions. One of the major challenges in this work is that we use a surrogate to model discrete responses (i.e., accept or reject) coming from the agent's opponents to previously thrown offers. While, it is known that surrogate models are used to model continuous spaces [12]. To the best of our knowledge, this is the first work where surrogate is used to map discrete spaces in agent-based negotiation systems.

The main assumption, in this paper, is that all participating agents do not disclose any information about their utility functions. Moreover, we assume that agents' utility functions are fixed until the end of the negotiation process.

B. Surrogate Modelling

Surrogate models (SMs) used in evolutionary frameworks, typically known as response surface models or meta-models, are approximation models that mimic the behaviour of the simulation model as closely as possible while being fast surrogates for time-consuming objective functions. In a nutshell, SMs work by running simulations at a set of points and fitting response surfaces to the resulting input-output data. To date, many data centric approximation methodologies have used to construct surrogates. These include the polynomial regression, support vector machines, artificial neural networks, radial basis functions, Gaussian process [13] and surrogate ensembles [14] are among the most commonly investigated [7]. Early approaches have focused on building global surrogates [15] that attempts to model the complete problem fitness landscape. However, due to the effects of the curse of dimensionality [16], many have turned to local surrogate models [17] [18] or their synergies [19] [20] or ensembles.

III. Negotiation Protocol

The negotiation protocol used in this work aims to study solutions for $n > 2$ agents to allocate $m > 1$ resources amongst themselves through negotiation. Thus, let the set $G = \{g_0, g_1, ..., g_n\}$ denote the set of agents participating in the negotiation process. Each agent is trying to maximise its utility function U^{g_i}. We assume that this utility function is hidden and different agents may act on different utility functions. The negotiation process moves round by round under a fixed deadline. All agents have to reach an allocation agreement, otherwise no resource will be allocated for any agent. In each r round, agents make offers sequentially. The order in which agents make offers in each round is decided randomly. For each offer, all opponents evaluate the given offer with respect to their utility functions and respond accordingly. Agents' offers represent an allocation of the resources to all agents (see Figure 1). Each agent can respond to a given offer as *Accept* or *Reject*. For any given offer, if all opponents are satisfied (i.e., all of them responded with Accept) then the negotiation ends in an agreement, otherwise, the negotiation will proceed to the next agent to send a new offer, and so on. The worst possible outcome of a negotiation is that agents don't reach an agreement until the deadline, and then the negotiation is declared to be failed.

An agent accepts an offer if and only if its profit is more than its reservation value, otherwise, it will be rejected. The reservation value of each agent is a predefined percentage of the maximum possible profit from its utility function. More formally, if U^{g_i} is the utility of agent g_i, where U^{g_i} is a function that receive the allocated shares of each resource as parameters. Thus, $U^{g_i}(\alpha_0, \alpha_1, ..., \alpha_m)$ determines agent g_i's profit, where $\alpha_i \in [0, 1]$ represents a proposed offer to allocate of the i^{th} item and $o^{g_i}_{r_t} = \{(\alpha_i)|i = 1, ..., m\}$ denote an offer given to g_i in round r at time t. The reservation value is calculated as follows:

$$ReservationValue = Max(U^{g_i}) \times minimum\_accepted\_profit.$$

where $minimum\_accepted\_profit$ is a number from the interval $[0, 1]$. Thus, agents responses to given offers can be formulated as:

$$Response(g_i) = \begin{cases} if \mathrm{U}^{g_i} > ReservationValue & , Accept \\ else & , Reject \end{cases}$$

IV. Negotiation Agent

In a nutshell, our proposed agent builds a library of observations about its opponents' interactions and then uses this information as the training set for surrogate modelling (i.e., one per opponent). To this end, the proposed agent uses a form of a continuous learning process throughout the negotiation, where it accumulates observations and increases its knowledge about preferences of its opponents. Hence, it collects information about the offers sent to each agent along with their responses. Each observation can be represented as a triplet of $< g_i$, *offer, response*$>$, where an offer can be denoted $o^{g_i}_{r_t} = \{\alpha_0, \alpha_1, ..., \alpha_m\}$. For the responses, we encode each accepted offer as 1 and each rejected offer as 0. Thus, for each agent, a vector v^{g_i} of observations can be formulated. The proposed agent uses each collected vector as the training set to build a surrogate model for each opponent. This process can be formulated as follows: Let set $V = \{v^{g_0}, v^{g_1}, ..., v^{g_n - 1}\}$ be the set of observations collected for each agent at particular round, while let set $S = \{s^{g_0}, s^{g_1}, ..., s^{g_n - 1}\}$ be the set of surrogates used to model the agents' behaviours. Hence, $\forall s^{g_i} \in S$, v^{g_i} is used as the training set to model the function $s^{g_i} : \mathbf{O} \rightarrow \mathbf{R}\{0, 1\}$. Note that the set V is updated with the new observations after each round while set S comprises of built surrogates of the newly updated observations accordingly.

As will be shown in the experiments section, we investigated our agent's performance on two form of machine learning approximation methodologies, namely, RBFN and Kriging [7]. With the set of surrogate models available, a GA engine is then used to evolve offers in the agreement zone. If the proposed agent managed to evolve an acceptable offer for all agents, then we declare that it managed to successfully end the negotiation in agreement. Otherwise, it will wait for its turn in the next round to make a new offer. Meanwhile, new observations will be accumulated and surrogate models will possess increasing knowledge about the offers space of each opponent. The following sub-section will present, in details, the process of evolving offers.

A. *Evolve Offers*

Our proposed agent uses a GA engine to explore the space of offers. For the purpose of offers representation, each offer to our agent's opponents is represented as a matrix of size $m \times n$ (as illustrated in Figure 1) where each row represents the division of m^{th} resources among n number of agents and each column represents the n^{th} agent's share from each resource. Thus, the GA population comprises of solution individuals that are formulated as matrices. For the GA, we consider three different operators that take the matrix representation into considerations (more details in subsection IV-B). All search operators are designed to maintain correct syntax (i.e., the summation of the values in any row should be 1). The fitness function of the GA is made up of two functions to evaluate the quality of each offer-matrix in the population and a penalty scheme to assure convergence of the evolution toward accepted offers. The first function is formulated as follows:

$$F1 = \sum_{i=1}^{n-1} s^{g_i}(\mathbf{Offer_{x_q}}) \tag{1}$$

where $s^{g_i}(\mathbf{Offer_{x_q}})$ is the prediction outcome of the surrogate that was trained by observations collected for agent g_i. Note that we round the surrogate's output to 0 or 1. Thus, the GA is searching for offers that maximise their possibility of being accepted. The second function is simply the utility function of our agent. Hence,

$$F2 = U^{g_0}(\alpha_0, \alpha_1, ..., \alpha_m) \tag{2}$$

where U^{g_0} represents the utility function of our proposed agent. Finally, the penalty term is the outcome of a K-Nearest Neighbour (KNN) algorithm. The KNN is measuring the Euclidean distance between an evolved offer and previous offers to each agent. The aim is to maximise the distance to previously rejected offers and minimise the distance to previously accepted offers. The use of KNN in the fitness measure assures that the search converges toward an agreement zone. The penalty term is calculated as follows:

$$Penalty = \sum_{i=1}^{n-1} \frac{\sum_{j=1}^{k} Distance(\mathbf{Offer_{x_q}}, g_i(Accepted\_offer_j))}{\sum_{z=1}^{k} Distance(\mathbf{Offer_{x_q}}, g_i(Rejected\_offer_z)) + 1} \tag{3}$$

where $Distance(\mathbf{Offer_{x_q}}, g_i(Accepted\_offer_j))$ is the Euclidean distance between an evolved offer ($\mathbf{Offer_{x_q}}$) and the j^{th} previously accepted offer for the g_i agent. The $g_i(Rejected\_offer_z)$ is used to denote the z^{th} rejected offer by the same agent. Note that good solutions are on the borderline between accepted and rejected offers and there where the search should focus. To purely maximise the distance to rejected offers makes it difficult to stay on this borderline. Hence, Equation 3 is treated as a penalty term (which is mainly should be minimised) while Equations 1 and 2 are meant to be maximised.

As will be seen in the experimental sections, in order to validate the significance of the KNN in the fitness measurement, we study the performance of the agent with the use of KNN and without.

The final fitness function is the summation of Equations 1 and 2, in addition Equation 3 used as a penalty term. The outputs of Equations 1 and 2 have been scaled to values in the interval $[0, 1]$. Thus, the fitness value of any individual in the GA population is:

$$Fitness = \frac{F1}{n-1} + \frac{F2}{Max(U^{g_0})} - Penalty \tag{4}$$

The objective is thus to maximise the fitness function or the quality of the evolved offers. GA undergoes a search process, guided by the fitness function in Equation 4, to explore the space of offers and return the most promising offer that is most likely to be accepted by our agent's opponents and maximise profit as much as possible. It should be noted that the surrogate is used to model the unknown behaviours of our agent's opponents and not to save computational costs.

B. *Search Operators*

As mentioned previously, the search operators are designed to maintain proper syntax of the individual's matrix representation. Here, we consider three operators to explore the offers space. First, we used a crossover operator in which the two matrices exchange a row, randomly. Second, an aggressive mutation operator that replaces an entire randomly selected row with a randomly generated row. Note that any randomly generated row shall adhere to the defined syntax imposed in which the summation of its values equal to 1. Finally, we used a smooth mutation operator where an element of a random selected row is mutated into a new value. In this mutation, the difference between mutated values and the old value is traded with the remaining elements in the same row in such a way as to maintain the total summation of the elements within the same row to 1.

C. *Surrogate Learning*

In principle, the proposed agent can apply any surrogate model to approximate the offers space of its opponents. To verify this, we investigated the performance of the proposed agent using two different surrogate models (i.e., RBFN and Kriging). It is worth noting that in principle, all surrogate models are implicitly or explicitly spatial models as their predictions involve exploiting some assumed spatial smoothness or continuity between the values of the function at a query point whose value is unknown and has to be predicted, and the known solutions in the search space. This makes surrogates naturally suited to continuous function optimisation. However, remember that our interest is on surrogates that models given input offers to discrete responses as output due to the discrete nature of the agents responses.

The point $o^{g_i}_{r_t}$ represents an offer given to the agent g_i in round r at time t. Thus, we store all offers $o^{g_i}_{r_t}$ in each round along with their responses (i.e., Accept= 1, Reject= 0) and then use this information as training points for the s^{g_i} surrogate. The number of training point increases in each round to allow the surrogate collect more information about the undersampled offers space of the g_i agent. Remember the proposed agent builds an independent surrogate model for each opponent.

The next two sub-sections will describe the mathematical notation of both RBFN and Kriging model.

1) RBFN: A radial basis function (RBF) is a real-valued function $\phi : \mathbf{R}^n \to \mathbf{R}$ whose value depends only on the distance from some point $o^{g_i}_{r_t}$, called a *center*, so that $\phi(\mathbf{Offer}_{\mathbf{x_q}}) = \phi(\|\mathbf{Offer}_{\mathbf{x_q}} - o^{g_i}_{r_t}\|)$. The point $o^{g_i}_{r_t}$ is a parameter of the function and $\mathbf{Offer}_{\mathbf{x_q}}$ is a new offer in the offers space of the agent g_i (i.e., query point).

The norm is usually Euclidean, so $\|\mathbf{Offer}_{\mathbf{x_q}} - o^{g_i}_{r_t}\|$ is the Euclidean distance between the offer $o^{g_i}_{r_t}$ and the $\mathbf{Offer}_{\mathbf{x_q}}$. Other norms are also possible and have been used. Commonly used types of radial basis functions include Gaussian functions, multi-quadric, poly-harmonic splines, and thin plate splines. The most frequently used, and the one we used on this paper, are Gaussian functions of the form:

$$\phi(x) = \exp(-\beta\|\mathbf{Offer}_{\mathbf{x_q}} - o^{g_i}_{r_t}\|^2)$$

where $\beta > 0$ is the width parameter.

Radial basis functions networks are typically used to build function approximations of the form:

$$y(\mathbf{Offer}_{\mathbf{x_q}}) = w_0 + \sum_{t=0}^{N} w_i \, \phi(\|\mathbf{Offer}_{\mathbf{x_q}} - o^{g_i}_{r_t}\|).$$

Therefore the approximating function $y(\mathbf{Offer}_{\mathbf{x_q}})$ is represented as a sum of N radial basis functions, each associated with a different center $o^{g_i}_{r_t}$, a width β, and weighted by an appropriate coefficient w_i, plus a bias term w_0. Any continuous function can in principle be approximated with arbitrary accuracy by a sum of this form, if a sufficiently large number N of radial basis functions is used.

2) Kriging: Kriging or Gaussian Process can be seen as a general case of the RBFN where the approximation of an unseen point take the following form:

$$y(\mathbf{Offer}_{\mathbf{x_q}}) = exp(-\sum_{t=0}^{N} \beta_t |\mathbf{Offer}_{\mathbf{x_q}} - o^{g_i}_{r_t}|^{p_{t+1}})$$

Similar to RBFN, Kriging is based on the idea that the value at a query point ($\mathbf{Offer}_{\mathbf{x_q}}$ in our case) should be the average of the known values at its neighbours weighted by the neighbours' distance to the query point. Unlike the RBFN, however, Kriging uses a vector $\beta = \{\beta_0, ..., \beta_t\}^T$ allow the width to vary from variable to variable [21].

V. Experiments

Experiments were designed to investigate the proposed agent under different scenarios so as to bring insights on the potential benefits and limitations.

TABLE I
Parametric settings of the algorithms considered settings in the experiments.

Parameter	*Proposed Agent's GA*
Aggressive Mutation	15%
Smooth Mutation	15%
Crossover	70%
Tournament size	5
Population Size	100
Generations	100

Parameter	*Selfish GA*
One-Point Mutation	30%
Crossover	70%
Tournament size	5
Population Size	100
Generations	100

A. Experimental Settings

The present experimental work has been divided into two different studies. In the first study, we considered four different versions of our agent against a standard selfish GA. The selfish GA basically tries to optimise the shares from each resource in such a manner as to maximise the profit of the agent it serves and divide the remaining share of each resource among the remaining agents equally. The fitness function of the selfish GA is the utility function of its agent. Usually, this selfish GA fails to reach an agreement. However, the aim of the first experimental study is to validate the efficacy of the different agent types. Remember that our agent builds a surrogate model for each opponent and involves a GA that evolves offers in the agreement zone (see Section IV). The four configurations of our agent considered here are listed as follows:

- Version 1: We used *Kriging* as the surrogate model *with KNN* as part of the GA fitness measurement to ensure convergence of the search (see Equation 3).
- Version 2: We used *Kriging* as the surrogate model *without KNN* in the GA fitness measurement.
- Version 3: We used *RBFN* as the surrogate models *with KNN* as part of the GA fitness measurement.
- Version 4: We used *RBFN* as the surrogate models *without KNN* as part of the GA fitness measurement.

Since surrogate are mainly suited for continuous spaces, it would be interesting to observe the behaviour of the two state-of-the-art approximation methodologies for the present application (i.e., discrete space). The KNN heuristic was switched on and off to validate the effects on GA search. The parametric settings of the selfish GA and the GA used by our agent for evolving agent offers are tabulated in Table I. The settings were optimised using a trial-and-error procedure during preliminary experiments in a way that maximise the performance of all systems in the comparison.

We investigate each configuration of our agent (versions 1-4) against four agents to form a negotiation game that makes up of 5 competing agents. Each agent is supplied with a unique utility function and preference of resources in the negotiation as defined in the Appendix of Section VI. Each competitor agent uses the selfish GA to generate its offers. The experimental study include negotiations over $10, 20, 30, 40, 50, 100$, and 120 different resources. For each of resource size, 20 independent simulation runs were conducted. The deadline for all negotiations was set to 10 rounds.

In the second experimental study, we allow each of the four different agent versions (versions 1-4) to compete against each other in the same game. In addition, we added a benevolent agent that uses a standard GA to optimise its shares from each resource in such a way to get the reservation value only and divide the remaining share of each resource equally among the remaining agents. Thus, we formed a negotiation game of 5 agents where each agent uses different negotiation techniques. Similar to the first experimental study, we allowed the 5 agents to negotiate over $10, 20, 30, 40, 50, 100$, and 120 different resources. Once again, for each resource size, we tested the agents through 20 independent runs at a deadline of 10 rounds for negotiations.

B. Results

Table II shows the results of our first experimental study. As mentioned previously, we investigate each of the four versions of our agent against 4 other selfish GA agents. We used utility function U^{g_0} (see Appendix VI) to represent our agent while the utility functions U^{g_1}, U^{g_2}, U^{g_3}, and U^{g_4} represent our agent's opponents. The reservation value was set to be $0.5 \times Max(U^{g_i})$. For each of 20 independent runs, we collected the number of times that our agent won the negotiation over the opponents (i.e., managed to end the negotiation in agreement before the deadline). In addition, we present the average number of rounds (across the 20 runs) taken by our agent to end the negotiation in agreement. Moreover, we show our agents' profit (i.e., outcome of U^{g_0}). Finally, also reported the quality of each version as:

$$Quality = \frac{Wins}{Avg.Rounds} \times profit$$

To this end, a high *Quality* value reflects an agent that achieves high number of wins (across the different negotiation games) fast, i.e., before the deadline is reached, while achieving high profit. As can be seen from Table II, RBFN with KNN achieved the highest quality when the agents were negotiating over 10 and 20 resources. When the negotiation problem got tougher (i.e., agents were negotiating over more resources) the RBFN (without KNN assistance) exhibited the highest quality. In fact, the RBFN is found to win all games when agents were negotiating over 50 resources. The large disparity in the quality of RBFN and Kriging thus seems to imply that both models the discrete landscapes rather differently. Interestingly, Kriging did not manage to achieve an agreement in any game (out of the 20 runs) in two cases, i.e., when 10 and 30 resource sizes are considered. The results in the table also indicated that RBFN has always been able to achieve the highest profit. It, however, could not achieve an agreement in the early rounds relative to the other versions. This thus indicates that RBFN may be more suitable for negotiation games with a longer deadline where the user is willing to take a higher risk as a trade-off for profits. Both Kriging with KNN and RBFN with KNN also exhibited the smallest average rounds necessary to reach an arrangement in most cases. This indicates that they could be more suitable for users who are unwilling to take risk for sake of possibly higher profits.

The results in Table II are further supported in Table III, where the performance for the 4 different versions of our proposed agent are summarised. Clearly, RBFN always achieve the highest profit. However, it does so at the expenses of higher number of rounds incurred before reaching an agreement. Kriging on the other hand incurred the smallest number of rounds but has a small number of wins (it won only 38.6% of the negotiation games in the experiments). Agent with Kriging and KNN has second smallest number of rounds at high number of wins, winning 66.4% of the negotiation games in the experiments.

TABLE II
RESULTS OF EACH PROPOSED AGENT (VERSIONS 1-4).
EACH RESOURCE SIZE WAS SIMULATED FOR 20 INDEPENDENT RUNS.

	Kriging_KNN	Kriging	RBFN_KNN	RBFN
Resources = 10				
Wins	12.000	0	15.000	8.000
Avg. Round	4.833	N/A	4.800	5.000
Profit	0.948	N/A	0.908	0.997
Quality	2.354	N/A	**2.837**	1.595
Resources = 20				
Wins	11.000	1.000	19.000	17.000
Avg. Round	4.727	8.000	4.053	4.824
Profit	0.854	1.000	0.836	0.982
Quality	1.987	0.125	**3.921**	3.460
Resources = 30				
Wins	7.000	0	12.000	10.000
Avg. Round	3.857	N/A	5.833	4.700
Profit	0.817	N/A	0.902	0.998
Quality	1.483	N/A	1.856	**2.124**
Resources = 40				
Wins	10.000	7.000	11.000	14.000
Avg. Round	3.600	4.429	5.091	4.357
Profit	0.821	0.998	0.753	1.000
Quality	2.282	1.578	1.627	**3.212**
Resources = 50				
Wins	17.000	14.000	16.000	20.000
Avg. Round	4.588	3.786	3.500	3.850
Profit	0.770	0.995	0.760	0.999
Quality	2.852	3.681	3.473	**5.188**
Resources = 100				
Wins	16.000	14.000	13.000	16.000
Avg. Round	4.063	4.071	5.231	3.750
Profit	0.797	0.982	0.886	0.984
Quality	3.141	3.378	2.202	**4.200**
Resources = 120				
Wins	20.000	18.000	20.000	17.000
Avg. Round	3.600	4.278	4.050	3.235
Profit	0.781	0.946	0.756	0.930
Quality	4.337	3.979	3.732	**4.888**

* Bold numbers are the best qualities.
* N/A means the agent didn't end any negotiation in agreement.

Table IV illustrates the results of the second experimental study, where the agents versions 1-4 compete. In addition, a benevolent agent is included in the game. We used utility function U^{g_0} to represent the Kriging with KNN agent, utility function U^{g_1} to represent the Kriging without KNN agent, utility functions U^{g_2} to represent the RBFN with KNN agent, utility functions U^{g_3} to represent RBFN without KNN agent, and finally the utility function U^{g_4} to represent the benevolent agent. The results in Table IV indicate that both Kriging with KNN and RBFN with KNN attained the highest number of wins. Notably, Kiring with KNN dominated all negotiations when the number of resources available increased, i.e., in the large resource sizes. Also, we noted that the benevolent agent did not win any game in all experiments. This can be explained by the unique preferences for the resource items. As such, a fair distribution of the extra shares can never satisfy all agents. We noticed that in the case of 10 resources size the agents failed reach an agreement in 4 out of 20 games. With a size of 100 resources the agents failed to reach an agreement in only 1 out of 20 games. With a size of 120 resources the agents failed to reach an agreement in 6 out of 20 games.

TABLE III
SUMMARISED PERFORMANCES OF THE PROPOSED AGENT (VERSIONS 1-4). EACH RESOURCE SIZE WAS SIMULATED FOR 20 INDEPENDENT RUNS.

Summary	Kriging KNN	Kriging	RBFN KNN	RBFN
Avg. Wins	66.4%	38.6%	**75.7%**	72.9%
Avg. Rounds	4.181	3.509	4.651	4.245
Avg. Profit	0.827	0.703	0.829	**0.984**
Quality	2.634	1.820	2.807	**3.524**

* Bold numbers are the highest.

TABLE IV
RESULTS OF 5 AGENTS COMPETITION IN THE SAME GAME. EACH RESOURCE SIZE WAS TESTED 20 DIFFERENT TIMES.

	Kriging KNN	Kriging	RBFN KNN	RBFN	Benevolent Agent
Resources = 10					
Wins	4.000	0	**8.000**	4.000	0
Avg. Round	4.000	N/A	3.875	1.250	N/A
Resources = 20					
Wins	**10.000**	0	9.000	1.000	0
Avg. Round	4.600	N/A	3.778	2.000	N/A
Resources = 30					
Wins	8.000	0	**11.000**	1.000	0
Avg. Round	3.625	N/A	2.000	2.000	N/A
Resources = 40					
Wins	**11.000**	1.000	7.000	1.000	0
Avg. Round	3.182	3.000	2.286	3.000	N/A
Resources = 50					
Wins	**11.000**	2.000	7.000	0	0
Avg. Round	5.818	3.000	5.000	N/A	N/A
Resources = 100					
Wins	**11.000**	1.000	7.000	0	0
Avg. Round	2.000	9.000	5.571	N/A	N/A
Resources = 120					
Wins	**12.000**	1.000	1.000	0	0
Avg. Round	2.917	8.000	3.000	N/A	N/A

* Bold numbers are the highest wins.
* N/A means the agent didn't end any negotiation in agreement.

C. Discussion

There is a natural trade off between profit and number of rounds until agreement. From tables II and III, the inclusion of KNN led to accelerate the convergence of the agent's search for the agreement zone. However, this acceleration came at the expense of a lower profit. This thus suggests that one can present a risky agent through the exclusion of the KNN or a safe agent with the inclusion of KNN based on user preference, accordingly. This conclusion shed light on the efficiency of the proposed agent, where Table III shows that the Kriging requires on average 3.5 rounds before it reaches an agreement, while the Kriging with KNN requires a higher average of 4.1 rounds before it ends the negotiation in agreement. With a maximum number of rounds was configured to 10 the proposed agent is able to reach an agreement within 50% of the negotiation's maximum time budget available.

It has been observed from the experimental results, the complexity of the utility functions used for our agent's opponents are non-trivial. As a result of the strategy adopted by our proposed agent in building a surrogate that models the behaviour of for each opponent it becomes possible to predict the offers by of each opponent separately and, thus, end negotiation in agreement more oftenly. This indicate that the proposed agent can work well with several agents regardless of the complexity of their utility functions. We can safely assume that in real-world negotiation the agents may use functions of the same level of complexity if not more complex.

It is fair to report one major limitation of the proposed agent is that it requires some initial observations are necessary in order to collect sufficient training data for training surrogate models that generate reasonably good offers prediction. Thus, it cannot be the first to provide an offer in the negotiation process. If the first turn in the first round were allocated to either of the proposed agents then a random offer has to be made.

VI. CONCLUSIONS

This paper presents a *negotiation agent* for a multi-agent multi-issue negotiation model under incomplete-information scenarios to solve a resource-allocation problem. Unlike other works, the proposed agent is designed to work with Package Deal Procedures where all agents have to agree on sharing all issues in the negotiation (before a deadline) or the negotiation ends in disagreement. We use a multilateral negotiation protocol, by which agents make offers sequentially in consecutive rounds until a deadline. Agents' offers represent suggestions about how to divide the available resources among all agents participating in the negotiation. We used a continuous learning strategy where the proposed agent learns the preferences of its opponents through their interactions. To this end, the proposed agent is building a library of observations about its opponents' interactions and uses this information as training set for surrogate models (i.e., one for each opponent).

One of the major challenges in this work is that we use surrogate to model discrete responses (i.e., accept or reject) coming from the agent's opponents to previous offers. In our experimental settings, we proposed four different versions of our agent where we tested each version individually (against a selfish GA system) and allowed all versions to compete against each other. In addition, we compared our agent against a benevolent agent. Results show that the proposed negotiation agent can ends the negotiation in agreement before the deadline while maximising profits. Moreover, our results indicates that Kriging and RBFN surrogate treat the same discrete landscape differently.

One major weakness of our agent is that it can not take the first turn in the first round to make an offer for its opponents. This is because the proposed methodology requires the agent to collect some information about its opponents in order to make sensible offers.

In future work, we will develop our agent to be able to deal with dynamic environment in which opponents can change their utility functions during the negotiation process. In addition, we will explore the performance of our agent when the number of opponents is large.

Acknowledgments

The third author wishes to acknowledge Science Foundation Ireland (SFI) under the Principal Investigator research program 10/IN.1/I2980 "Self-organizing Architectures for Autonomic Management of Smart Cities" and by SFI grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) for their support to conduct this research.

Appendix

We use the value α_j to refer to the agent's share from the j^{th} resource. Each α_j is multiplied with a value $p_j^{g_i} \in [0,1]$ to indicate the agent's preference of the j^{th} resource. Hence, the set $P^{g_i} = \{p_0^{g_i}, p_1^{g_i}, ..., p_j^{g_i}\}$ represents the agent g_i preferences of the resource items in the negotiation. The value of each $p_j^{g_i}$ is randomly selected in each run.

- Utility of Agent 0 (Dixson & Price function): $U^{g_0}(\alpha_0 : \alpha_m) = [\alpha_0 - 1]^2 + \sum_{j=1}^{m} j \times [\alpha_j \times 2]^2 - \alpha_j - 1$. [22].
- Utility of Agent 1 (Rastrigin function): $U^{g_1}(\alpha_0 : \alpha_m) = 36 \times \sum_{j=1}^{m} \alpha_j \times cos([10.24 \times 0.5^j - 5.12]^2) - ([10.24 \times 0.5^j - 5.12]^2) \times 2\pi) \times 10$. [22].
- Utility of Agent 2 (Michalewics function): $U^{g_2}(\alpha_0 : \alpha_m) = \sum_{j=1}^{m} sin(\alpha_j^2) \times sin(\frac{j \times {\alpha_j}^2}{\pi})^{2k}$ where $k = 10$. [22].
- Utility of Agent 3 (Rosenbrock function): $U^{g_3}(\alpha_0 : \alpha_m) = \sum_{j=1}^{m} [100(\alpha_j^2 - \alpha_{j+1})^2 + (\alpha_j - 1)^2]$ [22].
- Utility of Agent 4 (Griewank function): $U^{g_4}(\alpha_0 : \alpha_m) = \sum_{j=1}^{m} \frac{\alpha_j^2}{4000} - \prod_{j=1}^{m} cos(\frac{\alpha_j}{\sqrt{i}}) + 1$ [22].

References

[1] E. Galvan, C. Harris, I. Dusparic, S. Clarke, and V. Cahill, "Reducing electricity costs in a dynamic pricing environment," in *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, 2012, pp. 169–174.

[2] R. Y. K. Lau, "Towards genetically optimised multi-agent multi-issue negotiations," in *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 1 - Volume 01*, ser. HICSS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 35.3–. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2005.637

[3] S. Fatima and A. Kattan, "Evolving optimal agendas for package deal negotiation," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 505–512. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001646

[4] S. S. Fatima, M. Wooldridge, and N. R. Jennings, "Multi-issue negotiation with deadlines," *J. Artif. Int. Res.*, vol. 27, no. 1, pp. 381–417, Nov. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622572.1622583

[5] F. Abedin, K.-M. Chao, and N. Godwin, "An agenda based multi issue negotiation approach," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2012. [Online]. Available: http://dx.doi.org/10.1007/s12652-012-0123-1

[6] M. Wu, M. Weerdt, H. Poutre, C. Yadati, Y. Zhang, and C. Witteveen, "Multi-player multi-issue negotiation with complete information," in *Innovations in Agent-Based Complex Automated Negotiations*, ser. Studies in Computational Intelligence, T. Ito, M. Zhang, V. Robu, S. Fatima, T. Matsuo, and H. Yamaki, Eds. Springer Berlin Heidelberg, 2011, vol. 319, pp. 147–159. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15612-0_8

[7] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation." *Soft Comput.*, vol. 9, no. 1, pp. 3–12, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/soco/soco9.html#Jin05

[8] B. Rubenstein-Montano and R. Malaga, "A weighted sum genetic algorithm to support multiple-party multiple-objective negotiations," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 366 – 377, aug 2002.

[9] S. Matwin, T. Szapiro, and K. Haigh, "Genetic algorithms approach to a negotiation support system," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, no. 1, pp. 102 –114, jan/feb 1991.

[10] A. Kattan and S. Fatima, "Pso as a meta-search for hyper-ga system to evolve optimal agendas for sequential multi-issue negotiation," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, june 2012, pp. 1 –8.

[11] A. Kattan and F. Shaheen, "Evolving optimal agendas and strategies for negotiation in dynamic environments: a surrogate based approach." in *GECCO (Companion)*, T. Soule and J. H. Moore, Eds. ACM, 2012, pp. 1435–1436. [Online]. Available: http://dblp.uni-trier.de/db/conf/gecco/gecco2012c.html#KattanF12

[12] A. Moraglio and A. Kattan, "Geometric generalisation of surrogate model based optimisation to combinatorial spaces," in *Proceedings of the 11th European conference on Evolutionary computation in combinatorial optimization*, ser. EvoCOP'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 142–154. [Online]. Available: http://dl.acm.org/citation.cfm?id=2008339.2008352

[13] F. H. Lesh, "Muti-dimensional least-squares polynomial curve fitting," *Communications of ACM*, vol. 2, no. 9, pp. 29–30, 1959.

[14] T. Goel, R. T. Haftka, W. Shyy, and N. V. Queipo, "Ensemble of surrogates," *Structural and Multidisciplinary Optimization*, vol. 33, no. 3, pp. 199–216, 2007.

[15] H. Ulmer, F. Streichert, and A. Zell, "Evolution strategies assisted by Gaussian processes with improved preselection criterion," *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, vol. 1, pp. 692–699, 2004.

[16] D. L. Donoho, "High-dimensional data analysis: the curses and blessings of dimensionality," *American Mathematical Society Conference on Math Challenges of the 21st Century*, August 2000.

[17] K. C. Giannakoglou, "Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence," *Progress in Aerospace Sciences*, vol. 38, no. 1, pp. 43–76, 2002.

[18] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modelling," *American Institute of Aeronautics and Astronautics Journal*, vol. 41, no. 4, pp. 687–696, 2003.

[19] Z. Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee, "Memetic algorithm using multi-surrogates for computationally expensive optimization problems," *Soft Computing Journal*, vol. 11, no. 11, pp. 957–971, 2007.

[20] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Transactions on Systems, Man and Cybernetics (SMC), part C*, vol. 37, no. 1, pp. 66–76, 2005.

[21] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. Wiley, 2008.

[22] M. Molga and C. Smutnick, "Test functions for optimization needs," *Test functions for optimization needs*, 2005.