PSO as a Meta-Search for Hyper-GA System to Evolve Optimal Agendas for Sequential Multi-Issue Negotiation

Ahmed Kattan and Shaheen Fatima

Abstract— **This paper proposes a new technique based on Hyper Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) to evolve optimal agenda for bilateral multi-issue negotiation. In sequential negotiation the agenda specifies the set of issues included in the negotiation and the order in which they will be discussed. A player's profit from negotiation depends on the agenda. Each player wants to find an agenda that yields the highest profit, i.e., his/her optimal agenda. Our proposed technique identifies the best set of issues to be included in the agenda as well as the best ordering for the issues in a way that increases the player's profit. The proposed technique is comprised of two GA systems. Firstly, we have an outer GA system that searches for the best set of issues to be included in the agenda. Secondly, we have an inner GA system that searches for the best order of the selected issues. PSO is used to automatically adjust the parameters of these two GA systems. Empirical evidence demonstrates that the proposed technique evolves better agendas than standard GA, 1+1 Evolutionary Strategy, Fixed Settings Hyper-GA and a simple random search.**

I. Introduction

Negotiation is a process in which disputing players decide how to divide the gains from cooperation between themselves. Since this decision is made jointly by the players [10], each player can only obtain what the other is prepared to allow them. The simplest form of negotiation involves two players 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 players are likely to differ on the price at which they believe the trade should be, 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.

However, before the players can actually perform such negotiations, they must decide the rules for making offers and counteroffers. These rules are called the negotiation *protocol* or *procedure* [12], [3]. On the basis of this procedure, each player chooses a *strategy* (i.e., what offers to make during the course of negotiation). For competitive negotiations, which are the focus of this work, each player chooses a strategy that maximises their own utility/profit and therefore their *optimal strategy*. For example, buyer-seller negotiations are competitive in nature. For such negotiations, game theory [9] provides methods for analysing the strategic behavior of utility maximising players. 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 the Nash equilibrium [9]. In many buyer-seller negotiations, the players need to settle the price of not one but multiple items. Such negotiations are called multi-issue negotiations [6].

Multiple items/issues can be negotiated as a bundle or sequentially. In this paper, we focus on sequential negotiation. For sequential negotiation, the outcome depends on the set of items/issues chosen for negotiation and the order which they are discussed [3]. This set of issues and its ordering is called the *negotiation agenda*. Different agendas yield different profits to the players [3]. So, a player wants to know what agenda maximises his/her profit and therefore his/her *optimal agenda*. Given a set of m issues, the problem for an agent is to find a subset of $g < m$ issues to negotiate and order in which to negotiation in. There are $C(m,g) = \frac{m!}{g!(m-g)!}$ possible subsets of size g and the elements in each subset can be ordered in $g!$ ways.

In such multi-issue negotiation, the players usually set an agenda of size g (size of the agenda is the number of items/issues within the agenda to be included in the negotiation), and discuss the issues sequentially. It is most likely that in a multi-issue negotiation players have different evaluations regarding the importance of the issues under negotiation. Thus, some issues may be significant to one player while insignificant for the other player and vice versa. Here, the problem for a player is to find an agenda (i.e., of g issues) that maximises his/her profit and is therefore his/her optimal agenda.

In this paper, we will refer to the search space of all possible agendas of size g as $C(m,g)$. Thus, in order to find which agenda is optimal, we need to determine the right issues to be included in the agenda and then the right order of these issues. We call these two problems as A and B, respectively.

This work proposes a new technique that uses swarm optimisation to tune the settings of two GA systems (by settings we mean the population size, number of generations and search operators' rates), where each system is directed to solve the problems (i.e., problem A and problem B). Firstly, we have an outer GA system that searches for the best set of issues to be included in the agenda (to solve problem A). Secondly, we have an inner GA system that finds the best order of the selected issues in such a way to maximise the user's utility function (to solve problem B).[1] Each GA system is associated with numerical weight variables which are used to identify its population size, number of generations

Ahmed Kattan and Shaheen Fatima are with Computer Science Department, Loughborough University, UK, email: A.J.Kattan@lboro.ac.uk, S.S.Fatima@lboro.ac.uk.

[1]Each player has a utility/profit function and the aim is to maximise this function.

and search operators' rates. Thus, the weights of the two GA systems form a six-dimensional vector, which in turn corresponds to an individual/particle in a swarm population. Hence, a swarm population (i.e., collection of weights) is used to optimise the settings of the two GA systems in such a way to find optimal agendas under maximum number of evaluations. In a nutshell, we use PSO to automatically adjust the settings of each system in such a way to keep the whole process under control and get the best possible results. We show that our proposed *Hybrid-GA-PSO* can solve this problem better than standard GA, 1+1 ES, Hyper-GA with fixed settings and simple random search. Despite the simplicity of our current realisation of this idea, the results obtained by our approach have been remarkable in the sense that better agendas have been achieved and further improvement is still possible.

The contributions of this paper are twofold.

1) We propose a system that suggests the best set of issues to be included in the agenda as well as the best ordering of these issues before the actual negotiation process starts. Most existing work has taken the set of issues as given and analysed the equilibrium for different procedures (more details in Section II-A).
2) We show that a swarm optimisation can be effectively used as a meta-search for Hyper-GA's parameters settings to solve optimisation problems.

The rest of this paper is organised as follows. Section II briefly discusses previous work related to this research. Section III presents the negotiation protocol. Section IV presents a standard GA solution. Section V presents a detailed description of the proposed model. Sections VI and VII provide details of the experiments. Section VIII draws conclusions and possible future work.

II. Related Literature

A. Optimal Agenda

Negotiation has long been studied by game theorists. However, 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 *package deal procedure* PDP (where the issues are negotiated together as a bundle), the *simultaneous procedure* SP (where the issues are negotiated simultaneously and independently of each other), and the SQP *sequential procedure* [3] and have 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. This paper is directed toward finding the right agenda for the SQP.

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, [3], [1], [4] take the set of issues as given and show that the order in which they are negotiated is important in determining the outcome.

In [2], the authors proposed a Hyper-GA model to evolve agenda for package deal negotiations. This system consists of an outer GA, an inner GA search and a Surrogate model based on Radial Basis Function Network (RBFN). The outer GA is designed to search the space of all possible agendas. Thus, in the outer GA, each individual (i.e., agenda) is represented as a binary string. For m issues, the length of the string is m. Within the string, ones indicate which issues are included for negotiation. For a given g (i.e., the size of agenda), the string can have only g ones. The inner GA acts as a fitness evaluator for the outer GA's individuals – agendas–. For a given agenda in the outer GA population, the inner GA finds an optimal offer and associated utility. This utility is the fitness value (i.e., the profit for the agenda). To speed up the search process in the inner GA, a RBFN surrogate model was used instead of the inner GA. The surrogate's role is to search for potential agendas using a cheap computational model without the need to invoke an expensive inner GA run. Once the surrogate suggests an agenda, the inner GA is used to verify its real fitness.

Unlike [2] (which dealt with PDP), in this paper we propose a new technique (for the SQP) for not only finding the best issues to be included in the negotiation but also the best order for negotiation these issues.

B. GA-based PSO

The idea of using a search algorithm as a meta-search to search the parameters space of other stochastic search algorithms is not new. In [14], Zadeh envisioned that it is advantageous to employ Computational Intelligence (CI) techniques in combination rather than exclusively.

In [13], GA was used to optimise PSO parameters and enhance the PSO's abilities to solve unconstrained optimsation problems. The performance of the proposed method was evaluated using six standard problems. Results demonstrated that the proposed method can converge to global optimum and obtain better parameter settings for the PSO.

In [11], the author proposed a hybrid method that incorporates concepts from GA and PSO and creates individuals in a new generation not only by crossover and mutation operations as found in GA but also by the mechanisms of PSO. In [5], Juang proposed a new evolutionary learning algorithm based on a hybrid of GA and PSO, called HGAPSO. In HGAPSO, individuals in a new generation are created, not only by crossover and mutation operation as in GA, but also by PSO. The concept of elite strategy is adopted in HGAPSO, where the fittest 50% of the population are regarded as elites. However, instead of being reproduced directly to the next generation, these elites are first enhanced using PSO. The group constituted by the elites is regarded as a swarm, and each elite corresponds to a particle within it. Thus, PSO optimise the elite solutions before copying them into the next generation. The other 50% of the population is generated by performing standard crossover and mutation operation on these enhanced elites.

Unlike other work, here we propose PSO system to automatically optimise the parameters of Hyper-GA system in

such a way to get the best performance within limit number of evaluation (details in Section V).

III. Negotiation Protocol

The issues on a given agenda are negotiated using a *sequential* protocol. In this protocol, the issues are negotiated one at a time in a given *order*. Negotiation on an issue does not begin until the previous issue is successfully negotiated. For example, if there are m issues on the agenda, then negotiation on issue i ($1 \leq i \leq m$) begins after all the previous $i-1$ issues are agreed upon. The agreement on an issue is implemented as soon as it is agreed.

In a typical negotiation, the agents will have different utility functions. Also, these functions will, in most cases, be *time dependent*. That is, an agent's utility from an issue will depend not just on the issues itself but also on the time at which it is agreed upon. So an agent's cumulative utility (which is the sum of the utilities for the individual issues on the agenda) depends not just on the set of issues on the agenda, but also on the order in which they are negotiated. A negotiating agent must therefore solve the following two problems: (A) From a given set of m issues choose $g < m$ issues and (B) For the g issues that are chosen, find an ordering that will maximise the agent's cumulative utility.

IV. A GA Solution

A possible GA solution to the problems A and B (as defined in Section I) is to use standard GA to explore the search space of all possible agendas. Thus, individuals in a GA population represent agendas of size $g < m$ where g is the number of items/issues to be included in the negotiation process and m is the total number of available issues. Each individual is represented as a sequence of integer values to indicate which issue is selected from m available issues. In order to ensure that each individual has a correct syntax, the following rules can be used. Firstly, the search operators should not allow the same issue to be used in the agenda more than once. Secondly, the search operators should allow freedom to the system to explore different orders for the issues within any agenda. To this end, two types of mutation operators are implemented; *1) Swap Mutation*: which randomly selects two issues from an agenda and swaps their locations, allowing the system to explore different permutations for the same set of issues within the agenda and *2) One-point Mutation*: which randomly selects an issue from an agenda and replaces it with an other issue that dose not exist in the agenda. The advantage of this system that it searches for both the right issues and the right order of the selected issues in the same process, hence, it is easy to implement. However, the limitation of this standard GA system is that it requires large number of evaluations to be able to evolve competitive agendas. Moreover, this system can easily get trapped in local optima which prevent it from exploring superior solutions in the $C(m, g)$ search space (results of this standard GA system are presented in Section VII). To solve these problems we present the following technique.

V. Proposed Methodology

This technique is based on GA and PSO. The proposed technique comprised of two GA systems working together to solve the whole problem in cooperation; *1) Outer GA* and *2) Inner GA*. The outer GA system searches for the best set of issues to be included in the agenda. For this task, the outer GA uses two types of mutation operator, namely, *swap mutation* and *one-point mutation*. Note that using these two types of search operators, as in the standard GA system described earlier in Section IV, allows the outer GA to not only find the right issues of the agenda, but also to explore the potential of the agenda when reordering its issues (inner GA will further explore different permutations of the agenda). Once the outer GA evolves an agenda AG, it passes it to the inner GA system. The search space of the inner GA is all possible permutations of AG (i.e., $g!$). The inner GA searches for the best order of the selected issues from $g!$ possible permutations in such a way as to maximise the negotiator's utility function. Thus, the initial population of the inner GA is comprised of different sequences of AG which were received from the outer GA. The inner GA uses a single search operator to explore different permutations within AG, namely, *swap mutation*.

The fitness measure for both GA systems (i.e., outer GA and inner GA) is defined by a utility (or profit) function that receives an agenda AG as an input and returns a profit value (real number) as an output. The utility function is a non-linear function (a detailed description of the utility functions used in our experiments is provided in Section VI).

In order to use two GA systems, it is important to find the ideal settings for each system in such a way to get the best results of both systems. One possibility is to set the population size and number of generations in each system to large numbers; however, this approach is computationally expensive especially if the user's utility function expensive to evaluate (which is not uncommon is many real world problems). Hence, we use PSO to automatically adjust the size of population and number of generations in each system as well as the search operators' rates in such a way as to keep the whole process under control and get the best possible results. For this task, each GA system is associated with a set of numerical weights $W = \{w_0, ..., w_i\}$, where $w_i \in i = \{0, ..., 5\}$, which is used to identify its settings. Table I illustrates the description of each variable.

During the optimisation process of the particles, two constraints are applied to ensure the validity of all settings. Constraint 1, to prevent the PSO from setting a large number for the population size and for number of generations (that potentially may be computationally infeasible) the total number of evaluations of both GA systems are not allowed to exceed a predefined maximum number of evaluations. Hence, $(w_0 \times w_1) + (w_2 \times w_3) \leq max\_explorations$. The $max\_explorations$ is the upper limit of evaluations for both GA systems. Thus, the total number of evaluations performed by the inner GA and outer GA in any single run should not exceed this upper limit. The reason of setting this upper

TABLE I

PSO'S PARTICLES REPRESENTATION

Variable	*Description*	Value
w_0	Population size of the outer GA	Integer (1...max_explorations)
w_1	Number of generations in the outer GA	Integer (1...max_explorations)
w_2	Population size of the inner GA	Integer (1...max_explorations)
w_3	Number of generations in the inner GA	Integer (1...max_explorations)
w_4	Swap Mutation rate of the outer GA	Real number (0...1)
w_5	One-Point Mutation rate of the outer GA	Real number (0...1)

limit of evaluations is to allow the user to define an upper boundary of the computational costs. For constraint 2, the total rates of Swap mutation and One-Point mutation is 1, thus $\sum_{i=4}^{5} w_i = 1$. The weights of the two GA systems are forming a six-dimensional vector $V = \{w_0, ..., w_5\}$, which in turn corresponds to an individual in a swarm population $P_{pso} = \{V_0, V_1, ..., V_n\}$, where n is the size of the swarm. Hence, swarm population (i.e., collection of weights) is used to optimise the setting of the two GA systems in such a way to find optimal agendas under a limited number of evaluations.

A. Procedures

The proposed technique works as follows. Firstly, the system randomly initialises a swarm population of size n, according to the constraints mentioned in Section V. For each individual/particle V_j in P_{pso} the system runs the two GA systems (i.e., outer GA and then inner GA) using the settings from the V_j.

Once the system evaluates all V_j particles in P_{pso}, it allocates the best particle (i.e., the one which provided the best adjustment to the outer and inner GAs in such a way to return the best agenda) as a center and all other particles in the P_{pso} move toward the center using a velocity value $V_j[w_i]$. Kennedy and Elberhart proposed some velocity control equations in [7]. After several preliminary experiments, we found the best way to adjust the particles' velocities in our application is according to their difference from the best known location for each w_i in V_j particle as follows:

$if(center_{w_i} < V_j[w_i])$

$$V_j[w_i] = V_j[w_i] - vFactor \times rand_1 \times [|center_{w_i} - V_j[w_i]|]$$

else if(center$_{w_i} > V_j[w_i]$)

$$V_j[w_i] = V_j[w_i] + vFactor \times rand_1 \times [|center_{w_i} - V_j[w_i]|] \quad (1)$$

Thus, each particle in P_{pso} remembers the globally best position (which is found by a member in the flock). Each particle moves into the six-dimensional weights space (or parameters space) which denoted in the equation above as $V_j[w_i]$, where V_j represents the j^{th} particle and w_i represents the i^{th} dimension. The vFactor is a predefined constant and used to set the max step of any V_j particle in the P_{pso}. Preliminary experiments show the best value for $vFactor$ is 0.1. At each move, the system checks whether the particles' values satisfy the constraints mentioned in Section V. If the particles are not satisfying these constraints then a new $rand_1$ value is placed in Equation 1 until all constraints are satisfied. The process of velocity adjustment iterates until the maximum number of moves.

Algorithm 1 broadly outlines the whole process. Thus, in **line 1**, the system generates P_{pso}, of size n particles, according to the constraints defined in Section V. In **lines 3 - 11**, the system iterates over P_{pso}. For each V_j, it calculates the population size, number of generations, and search operators' rates for the outer and inner GAs (**lines 4 - 9**). Thereafter, the system runs the outer GA based on its new settings (**line 10**). The best agenda found by the outer GA stored into a variable called AG and then passed to the inner GA, where the system starts to explore different permutations for AG (**line 11**). In **line 12**, the best V_j treated as a center and all other individuals in P_{pso} update their velocity values (as explained in Equation 1) to move toward the center (**line 13**).

Algorithm 1: PSO tuning Hyper-GA settings

```
1  Random-Generate-Swarm(V_j)[w_0, ..., w_5], n)
2  repeat
3      foreach V_j do
4          OuterGA_Pop = w_0;
5          OuterGA_Gen = w_1;
6          InnerGA_Pop = w_2;
7          InnerGA_Gen = w_3;
8          OuterGA_Swap_Mutation = w_4;
9          OuterGA_One-Point_Mutation = w_5;
10         Agenda AG = Run(OuterGA);
11         Agenda Best_AG = Run(InnerGA, AG);
12     center_best[w_0, ..., w_5] = Best_AG
13     Update-All(V[w_0, ..., w_5])
14 until max-moves;
```

The size of the swarm population and number of moves is related to the problem difficulty (more details on the parameters' setting are provided in Section VI).

VI. EXPERIMENTS SETUP

Experiments were conducted on various agenda sizes g out of m different number of issues to be included in the negotiation. The main aim of the experiments is to evaluate the performance of the model and to assess its behaviour. As

a reference of performance, we compared the model against *1) Standard GA* to measure whether our Hyper-GA system based on PSO managed to outperform a simple and standard solution of this problem, *2) 1+1 Evolutionary Strategy (ES)*, based on one-point mutation, to compare our method against standard search algorithm, *3) a Hyper-GA with fixed setting* to ensure that parameters search improves the performance of the system and finally, *4) a simple Random Search (RS)*. The reason we included a random agenda generator in the comparison is because, whereas it is safe to assume that in practise evolutionary algorithms are better than a random search under normal circumstances, however, with small samples random search can do relatively well. In addition, we included in the comparison a simple baseline method which is used to estimate the global optimum of the given $C(m, g)$ search space. This baseline method applies a very large GA search (i.e., the standard GA described in Section IV) using a population size of 1000 in 1000 generations and we run it for 100 independent runs and reported the best results of all runs.[2]

A utility function can be any real-valued function $f : \mathbf{R}^g \rightarrow \mathbf{R}$ from the space of functions. Therefore, in principle, any function can be used as a utility function. In our experiments we evaluated the system's performance using three different well-known benchmark problems. Namely, Rastrigin function, Dixson & Price function and, finally, Michalewics function [8] (see mathematical notations in Appendix). The reason for choosing these three functions in particular is because they represent different landscapes with different levels of difficulty. The Rastrigin function has the overall structure of a hyper-parabola with many bumps, whereas the other two functions have a smooth landscape but deceptive (the best optima are on different sides of the search space). In other words, they have key features of a typical utility function. For each function, we investigate the performance of the system under five different values of g. Thus, $g = 20, 30, 40, 50$ and 60. For all experiments $m = 100$. For each g, m combination we tested the system using 20 independent runs. Numerical results were summarised to conclude the system's behaviour (see Section VII). Table II illustrates the settings of GA systems used in the experiments.

Since the problem complexity is related to the g value, here, we relate the value of $max\_explorations$ (the upper limit of evaluations for both GA systems as explained in Section V) to problem complexity. Thus, $max\_explorations = [g \times 5]^2$. So, essentially our aim is to find the best solution to the problem the algorithm can produce in quadratic time out of an exponential number of candidate solutions in the $C(m, g)$ space. The size of P_{pso} is equal to $\frac{g \times 2}{10}$ and the number of moves for each particle in the swarm is 10. Thus, each particle in the P_{pso} effectively runs the outer and inner GAs 10 times in a process of tuning their parameters. Note that the total number of agendas evaluations performed by both GAs (i.e., the outer and inner GAs) in any single evaluation of V_j will not exceed the $max\_explorations$ value. Hence, to allow a fair comparison, when each individual in the P_{pso} runs the two GA systems, we run all systems in the comparison using the same number of evaluations (i.e., $max\_exploration$) and report the best results.

[2]This baseline method is computationally expensive which makes it impractical solution. Here, we used this method as a reference of performance only.

TABLE II
SETTINGS USED IN THE EXPERIMENTS

Operator	**Hyper-GA fixed parameters** *Outer GA*	*Inner GA*	*Standard GA*	1+1 ES	Random Search
One-Point Mutation	70%	0%	70%	100%	N/A
Swap Mutation	30%	100%	30%	0%	N/A
Tournament size	2	2	2	N/A	N/A
Population Size	$\frac{g \times 5}{2}$	$\frac{g \times 5}{2}$	$g \times 5$	1	$[g \times 5]^2$
Generations	$\frac{g \times 5}{2}$	$\frac{g \times 5}{2}$	$g \times 5$	$[g \times 5]^2$	N/A

VII. RESULTS

We compared the performance (both in terms of finding the best and the average of the best solutions) of our proposed approach (Hybrid GA-PSO) versus four different approaches (Standard GA, (1+1) ES, Hyper GA with fixed parameter settings and simple Random Search). Tables III,IV and V report these results.

Our proposed approach, Hybrid GA-PSO, outperformed its competitors in almost all cases. For instance, if we focus our attention on the system's behaviour when Rastrigin function is the fitness measure. Clearly, it outperforms its competitors in all five g values (indicated in boldface in Table III) compared with the other four approaches. If we further continue analysing these results, we can see that our approach still outperform its competitors on average on all cases of Rastrigin function. The fixed Hyper-GA comes in the second place in all g values and then the standard GA in the third place. 1+1 ES and Random Search come in the fourth and fifth places, respectively. This indicate that using PSO to tune the parameters of the Hyper-GA has added extra flexibility to the system and sequentially improved its performance. Also, the results' differences between the fixed Hyper-GA and Hybrid GA-PSO shows that parameters' settings of a search algorithm is an essential part its success or failure.

If we, now, turn our attention on the system's behaviour when Dixon & Price function is the fitness measure – which has smoother landscape than Rastrigin function– it is clear that Hybrid GA-PSO still outperforms its competitors and achieves the best results in all cases (indicated in boldface in Table IV). Moreover, Hybrid GA-PSO managed to achieve the estimated global optimum (using the baseline method) in $g = 30$. This is a remarkable result in the sense that Hybrid GA-PSO using $5g^2$ evaluations managed to achieve the same results as a large GA search that uses $1000 \times 1000 \times 20$ evaluations in this particular case. Similar to Rastrigin function, the fixed Hyper-GA comes in the second place in all g values and then the standard GA is in the third place.

TABLE III

SUMMARY OF 100 INDEPENDENT RUNS WITH THE RASTRIGIN FUNCTION (20 RUNS FOR EACH g VALE).

	Standard GA	*Hybrid GA-PSO*	*1+1 (ES)*	*Fixed Hyper-GA*	*Random Search*
g= 20					
Estimated Global Opt. = 687.932					
Mean	628.47	**643.15**	521.65	635.96	437.62
Std	11.58	36.22	19.06	6.30	15.58
Best	651.47	**678.97**	579.08	649.17	479.07
g= 30					
Estimated Global Opt. = 996.834					
Mean	878.89	**921.69**	732.22	882.19	633.60
Std	7.29	5.11	21.36	5.42	21.24
Best	890.35	**927.03**	801.51	895.73	677.48
g= 40					
Estimated Global Opt. = 1271.59					
Mean	979.86	**1033.68**	812.76	992.55	641.80
Std	4.67	1.73	21.63	6.68	24.73
Best	989.50	**1034.79**	868.42	1006.54	694.45
g= 50					
Estimated Global Opt. = 1582.32					
Mean	1091.63	**1127.55**	953.57	1101.13	729.63
Std	8.47	2.84	28.36	6.11	18.34
Best	1111.93	**1131.78**	1018.65	1111.02	764.52
g= 60					
Estimated Global Opt. = 1895.41					
Mean	1234.32	**1275.61**	1175.52	1245.37	927.56
Std	4.86	0.06	26.60	3.99	23.22
Best	1243.71	**1275.76**	1226.08	1252.92	992.58

*__Bold__ numbers are the highest. Underlined numbers are the second best.

Finally, looking at the system's behaviour when the Michalewics function is the fitness measure, it can be seen that Hybrid GA-PSO managed to achieve the best agendas in all five g values. However, Standard GA has slightly outperformed Hybrid GA-PSO averages in $g = 20$. Also, surprisingly, unlike the previous two functions, results of the Standard GA comes in the second place after Hybrid GA-PSO and fixed parameters Hyper-GA in the third place. This indicate that the standard settings which have been used in Fixed settings Hyper-GA (see table II) was not suitable to solve this particular problem effectively.

Another surprising observation, is that results show that random search achieved better solutions than 1+1 ES in $g = 20$ and $g = 30$. This is probably because random search exhibits a larger variance than 1+1 ES and with a "stroke of luck" it can be better.

We noticed that in all experiments the Hybrid GA-PSO produces a higher standard deviation than its competitors in some cases and a lower standard deviation in other cases. This is expected because PSO individuals are searching for the best settings of the two GA systems (i.e., outer and inner GAs) in the parameters' space. Thus, depending on the PSO initial population, each V_j individual explores different area in the search space, and thus, each V_j tries different settings which is not necessarily to the best in a process of optimising the solution. To further verify the significance of our results, a Kolmogorov-Smirnov two-sample test was performed on the test case results produced by the best evolved agenda in each run for all pairs of systems under test and for all g test cases. Table VI reports the *p-value* for the tests. As can be seen in the table, the proposed approach is statistically significantly superior to its competitors in all g cases for all three functions at the standard 5% significance level.

VIII. CONCLUSIONS

This paper proposes a new technique based on GA and PSO to evolve an optimal agenda for sequential multi-issue negotiations. Our proposed technique identifies the best set of issues to be included in the agenda as well as the best order of the issues in such a way to increase the players' profit. For this task, we propose a new technique that uses swarm optimisation to automatically adjust the settings of two GA systems, where each system is directed to solve part of the problem. Firstly, we have an outer GA system that searches for the best set of issues to be included in the agenda. Secondly, we have an inner GA system that finds the best order of the selected issues in such a way to maximise the user's utility function.

There are two main contributions in this paper. Firstly, we propose a system that suggests the best set of issues to

TABLE IV

SUMMARY OF 100 INDEPENDENT RUNS WITH THE DIXON & PRICE FUNCTION (20 RUNS FOR EACH g VALE).

	Standard GA	*Hybrid GA-PSO*	*1+1 (ES)*	*Fixed Hyper-GA*	*Random Search*
g= 20					
Estimated Global Opt. = 6019.80					
Mean	5721.65	**5811.93**	4572.87	5786.51	3988.72
Std	51.80	164.75	119.27	42.19	181.26
Best	5815.65	**6036.46**	4789.11	5851.56	4551.84
g= 30					
Estimated Global Opt. = 11631.30					
Mean	11122.02	**11552.93**	9190.08	11323.14	7905.44
Std	30.64	57.14	372.72	63.35	192.18
Best	11184.60	**11631.30**	9836.07	11397.10	8330.38
g= 40					
Estimated Global Opt. = 21959.10					
Mean	20882.64	**21619.96**	16886.37	21328.87	14831.70
Std	101.53	21.00	178.76	10.32	303.70
Best	21077.80	**21638.70**	17215.30	21334.40	15158.40
g= 50					
Estimated Global Opt. = 32973.90					
Mean	30172.31	**31000.48**	25700.93	30698.66	22100.88
Std	600.13	656.87	1046.31	572.12	781.61
Best	31454.70	**32499.70**	28940.30	32001.50	23850.20
g= 60					
Estimated Global Opt. = 42371.70					
Mean	38215.93	**38934.80**	34207.72	38837.48	28316.05
Std	862.22	777.30	1151.23	850.66	1022.37
Best	39015.80	**39424.20**	36613.40	39364.10	29550.90

*__Bold__ numbers are the highest. Underlined numbers are the second best.

be included in the agenda as well as the best ordering of these issues before the actual negotiation process starts. To the best of the authors' knowledge, most existing work has taken the set of issues as given and analysed the equilibrium for different procedures. Secondly, we show that a swarm optimisation can be effectively used as a meta-search for Hyper-GA's parameters settings to solve continuous optimisation problems.

Three different standard benchmark optimisation functions were used as utility functions to test the proposed technique. Results demonstrate that the proposed technique evolves better agendas than standard GA, 1+1 ES, Hyper-GA with fixed parameters and simple random search in all experimental cases. Kolmogorov-Smirnov two-sample shows that the proposed approach is statistically significantly superior to all of its competitors in all g cases for all three functions at the standard 5% significance level. In addition, results shows that the performance margins between Hyper-GA with fixed parameters and Hyper-GA with tuned parameters using PSO is significant, which indicates that the settings of a search algorithms is essential part of it's success or failure.

There are many directions were we can extend this work. For example, we can allow PSO to tune the GAs settings based on their populations' diversity rather than using their direct fitness. Also, in the current version of the system, the whole swarm converges toward a single leader. Thus, a potential an extension of this work is allow the swarm converge to multiple leaders in the population.

APPENDIX

AG refers to an agenda of size g. Here, AG is a vector that contains the negotiation items which found by GA search. The $f(x)$ represents a utility function.

Rastrigin function:
$f(AG) = 36g \times \sum_{i=1}^{g} AG[i] \times cos([10.24 \times 0.5^i - 5.12]^2) - ([10.24 \times 0.5^i - 5.12]^2) \times 2\pi) \times 10.$
[8]

Dixson & Price function:
$f(AG) = [AG[0] - 1]^2 + \sum_{i=1}^{g} i \times [AG[i] \times 2]^2 - AG[i] - 1.$
[8]

Michalewics function:
$f(AG) = \sum_{i=1}^{g} sin(AG[i]^2) \times sin(\frac{i \times AG[i]^2}{\pi})^{2m}$
where $m = 10$. [8]

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] S. Fatima and A. Kattan. Evolving optimal agendas for package deal negotiation. In N. Krasnogor and P. L. Lanzi, editors, *GECCO*, pages 505–512. ACM, 2011.

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

TABLE V

SUMMARY OF 100 INDEPENDENT RUNS WITH THE MICHALEWICS FUNCTION (20 RUNS FOR EACH g VALE).

	Standard GA	*Hybrid GA-PSO*	*1+1 (ES)*	*Fixed Hyper-GA*	*Random Search*
g= 20					
Estimated Global Opt. = 17.3794					
Mean	**14.27**	14.02	7.51	12.94	7.75
Std	0.20	2.04	0.24	0.47	0.41
Best	14.58	**16.69**	7.92	13.93	8.69
g= 30					
Estimated Global Opt. = 26.3191					
Mean	21.50	**23.00**	10.54	18.88	10.94
Std	0.16	2.61	0.34	0.16	0.09
Best	21.94	**25.64**	10.90	19.26	11.03
g= 40					
Estimated Global Opt. = 34.1278					
Mean	28.33	**30.72**	12.98	25.52	12.09
Std	0.42	0.31	0.64	0.44	0.67
Best	29.17	**31.37**	13.92	26.28	13.06
g= 50					
Estimated Global Opt. = 39.4839					
Mean	30.77	**34.73**	15.56	28.79	11.69
Std	0.02	0.13	0.09	0.16	0.00
Best	30.78	**34.76**	15.58	28.83	11.69
g= 60					
Estimated Global Opt. = 47.0512					
Mean	37.63	**43.80**	17.23	34.46	14.84
Std	0.66	1.41	1.22	0.12	0.17
Best	38.82	**44.63**	18.41	34.63	15.12

*__Bold__ numbers are the highest. Underlined numbers are the second best.

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

[5] C.-F. Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):997 – 1006, april 2004.

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

[7] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942 –1948 vol.4, nov/dec 1995.

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

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

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

[11] N. Ru and Y. Jianhua. A ga and particle swarm optimization based hybrid algorithm. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1047 –1050, june 2008.

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

[13] J.-Y. Wu. Real-coded genetic algorithm-based particle swarm optimization method for solving unconstrained optimization problems. In *Electronics and Information Engineering (ICEIE), 2010 International Conference On*, volume 1, pages V1–194 –V1–198, aug. 2010.

[14] L. A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Commun. ACM*, 37:77–84, March 1994.

TABLE VI

KOLMOGOROV-SMIRNOV TEST.

G	Hyper GA-PSO vs. GA	Hyper GA-PSO vs. 1+1 ES	Hyper GA-PSO vs. Random Search	Hyper GA-PSO vs. Fixed parameters Hyper-GA
Rastrigin Function				
20	**0.0026**	**4.7406e-09**	**5.5466e-10**	**0.05091**
30	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
40	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
50	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
60	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
Dixon& Price Function				
20	**0.0082**	**5.5466e-10**	**5.5466e-10**	**0.0232**
30	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
40	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
50	**1.5295e-06**	**5.5466e-10**	**5.5466e-10**	**0.0082**
60	**2.4894e-07**	**5.5466e-10**	**5.5466e-10**	**0.05091**
Michalewics Function				
20	**0.0082**	**5.5466e-10**	**5.5466e-10**	**0.0082**
30	**0.0082**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
40	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
50	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**
60	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**	**5.5466e-10**

*__Bold__ numbers are lower that 5% significance level.