A GP approach for Price-Speed Optimizing Negotiation

Michael Kampouridis and Kwang Mong Sim
School of Computing, University of Kent
Medway, Chatham Maritime, ME4 4AG, UK
Email: {M.Kampouridis, K.Sim}@kent.ac.uk

***Abstract*—This work uses a Genetic Programming (GP) algorithm to co-evolve negotiation strategies of agents that have different preference criteria, namely optimizing price and optimizing negotiation speed. While GP and other algorithms have been extensively used for price-only optimization, the problem of price-speed optimization has not yet received the same amount of attention. In Cloud/Grid computing environments, any delay in acquiring resources will be considered an overhead, hence negotiation agents need to adopt strategies that will enable them not only to optimize resource price but also to reach early agreements. This research is the earliest work to apply a GP algorithm for evolving price-speed optimizing negotiation strategies. An important advantage of the GP is its representation, which allows solutions to be represented in terms of the problem parameters, rather than as binary or real-value code, as it has been the case until now with other algorithms. We apply the GP to different negotiation scenarios and compare its results to other previously published works on the problem of price-speed optimizing negotiation agents. Results show that the GP 1) outperforms the algorithms from these previous works and 2) can evolve to an optimal or near optimal strategy.**

I. Introduction

In multi-agent systems, negotiation is a process where agents make offers to each other, and make concessions to reduce their differences in the hope of eventually reaching agreements. Although designing negotiation agents that only optimize utility can be sufficient for generic game theory [1], [2], [3], [4] and e-commerce applications [5], [6], [7], [8], there are other cases, such as Grid resource management [9], [10], where negotiation agents should be designed in taking into account that speed can be as important as price. For instance, in a Grid computing environment, failure to obtain the necessary computing resources before a deadline, could have an impact on the job executions of some applications.

In the past, there have been quite a few works that have applied evolutionary algorithms to automated negotiation problems [11], [12], [13], [14]. However, none of these works was experimenting with both Price and Speed (P-S) optimization. The only exception to the above, are two previous works of Sim [15], [16], where a Genetic Algorithm (GA), and an Estimated Distribution Algorithm (EDA) were used for P-S optimization. To the best of our knowledge, no other algorithms have been used for P-S optimizing negotiation.

However, a disadvantage of the GA and EDA in the above works was in their representation. In [15], [16], both algorithms used real-coded representation. While this approach has advantages over binary representation [17], [18], an important problem that exists is that this representation might need to be adapted, even if minor changes take place in the bargaining problem parameters [19]. On the other hand, the Genetic Programming (GP) [20], [21] has the advantage that it's a very flexible algorithm. More importantly, it can use parameters of the negotiation problem as leaf nodes, resulting to tree representations that are not affected by changes in the problem that the negotiation agents are dealing with. More information about the GP representation will be given in Section III-C.

Because of the advantages of the GP, we are interested in investigating its performance under diferent experimental setups for P-S optimization. To the best of our knowledge, this is the first time in the literature that GP is used for P-S optimization. This is important, because of the above-mentioned representation advantages of the algorithm. We thus run preliminary experiments for one-to-one P-S optimizing negotiation. We also compare the GP's performance with an EDA and a GA, the only two algorithms in the literature that have been used for P-S optimization. Our aim is to show that the GP is at least as good as these algorithms, or even outperforms them.

The rest of this paper is organized as follows: Section II presents the problem of P-S optimizing negotiation, and Section III discusses the three algorithms used in this paper, which as we have already mentioned are: EDA, GA, and GP. Section IV then presents the experimental setup of this paper, and Section V focuses on the experimental results. Section VI discusses the results. Lastly, Section VII concludes this paper.

II. Price-Speed Optimizing Negotiation

In this process, both Buyer (B) and Seller (S) are sensitive to time and have incomplete information for each other, i.e., they do not know each other's deadline and reserve price. Given an initial price IP_B (respectively, IP_S), reserve price RP_B (respectively, RP_S), deadline τ_B (respectively, τ_S), and time-dependent strategy λ_B, $0 \leq \lambda_B \leq \infty$ (respectively, λ_S, $0 \leq \lambda_S \leq \infty$), one can calculate the Utility function of each agent.

A. Utility function

The U^x of an agent x (B or S) is defined as:

$$U^x = w_p \times U_p^x + w_s \times U_s^x \tag{1}$$

where $U_p^x \in [0,1]$ is the price utility of x, and $U_s^x \in [0,1]$ is the speed utility of x, and $w_p + w_s = 1$. If x cannot reach an agreement before its deadline, then $U_p^x = U_s^x = 0$. Otherwise, U_p^x and U_s^x are defined as follows:

1) Price utility: Given the price P_c that a consensus is reached by both B and S, then if x is a buyer, the price utility $U_p^x(P_c)$ is given as follows:

$$U_p^x(P_c) = u_{min}^p + (1 - u_{min}^p)\frac{RP_B - P_c}{RP_B - IP_B} \quad (2)$$

Alternatively, if x is a seller, its price utility is:

$$U_p^x(P_c) = u_{min}^p + (1 - u_{min}^p)\frac{P_c - RP_S}{IP_S - RP_S} \quad (3)$$

In the above two equations, u_{min}^P is the minimum utility that x receives for reaching a deal at its reserve price.

2) Speed utility: If τ is the deadline of an agent x, and t_c is the number of rounds needed for x to reach an agreement, then the speed utility U_s^x of x for reaching a consensus at t_c is:

$$U_s^x(t_c) = u_{min}^s + (1 - u_{min}^s)(1 - \frac{t_c}{\tau}) \quad (4)$$

where U_{min}^s is the minimum utility that an agent receives for reaching a deal at its deadline. U_s^x is larger if x needs fewer rounds to reach an agreement.

B. Negotiation strategy

The proposal P_B for Buyer at time t, $0 \leq t \leq \tau_B$, is determined as follows:

$$P_t^B = IP_B + (\frac{t}{\tau_B})^{\lambda_B}(RP_B - IP_B) \quad (5)$$

Similarly, given an initial price IP_S, reserve price RP_S, deadline τ_S, and time-dependent strategy λ_S, $0 \leq \lambda_S \leq \infty$, the proposal P_S for Seller at time t, $0 \leq t \leq \tau_S$, is determined as follows:

$$P_t^S = IP_S + (\frac{t}{\tau_S})^{\lambda_S}(IP_S - RP_S) \quad (6)$$

C. Negotiation protocol

The negotiation protocol that B and S follow is Rubeinstein's alternating-offer protocol [4]. Thus, B makes an offer first at $t = 0$ and continues making offers at $t = 2, 4, 6, ...$; S makes counter offers at $t = 1, 3, 5, 7, ...$ During negotiation, B makes offers by using Equation 5, while S makes counter offers by using Equation 6. The negotiation terminates if either of the following two cases occur:

- An agreement has been reached by B offering at least the price that S has proposed. In other words, P_B needs to be greater than or equal to P_S. If this happens, the negotiation terminates and the price consensus P_C is B's proposed price P_B.
- The deadline of either B or S has been reached (whichever comes first). At this point S is allowed to make a last counter offer, even if it is not its turn. This is done to take into account the case where B is proposing $P_t^B = RP_S$ at time $t = \tau_S$;[1] in this case, S would have the incentive to accept B's offer. But in order to accept this offer, S should be given a chance to do this. Therefore, when the deadline is reached, S is allowed to make a last counter offer, even if it's not its turn, so that it is given an opportunity to accept B's offer at the level of RP_S.

D. The Problem

Given the above, we are interested in finding a value of λ_B (respectively, λ_S) that would optimise U^x. Taking into account that agents can have different deadlines, different preferences for the time and cost criteria (i.e., different w_s and w_p), and face different opponents (with different parameters, e.g., different deadlines and different strategies), using game-theoretic approaches to compute λ_B (respectively, λ_S) can be a hard problem to solve. In the next section we present how Genetic Algorithm and Genetic Programming can be used, for co-evolving the best strategies for B and S.

III. Algorithms

All algorithms in this paper will be using two sub-populations D^B and D^S, for B and S, respectively. These two populations are going to co-evolve, and individuals in D^B and D^S will negotiate with each other, through random pairing between an individual from D^B and an individual from D^S. The fitness of each individual is determined by its negotiation outcome (i.e., the price and the number of negotiation rounds that it used to reach an agreement with another individual in the other population).

One last thing that we would like to mention, before going into the details of each algorithm, is that the design of the EDA and the GA is the same as in Sim's previous works [15], [16]. This is done for consistency and comparison purposes, since these are the only other published works on the problem of Price-Speed optimizing negotiation agents.

A. Estimation Distribution Algorithm

1) Individual representation: An individual in D^B or D^S represents an agent's strategy λ. In our framework, an individual consists of three genes. Each gene can take random real values between 0 and 1. We are using real-coded representation, because binary coding mechanisms can have disadvantages, such as the existence of Hamming cliffs and the lack of computation precision [17], [18].

The strategy λ is calculated by summing up the values of the first two genes, and then dividing this sum by the number of the third gene. For instance, if an individual was the string [0.5 0.3 1], then $\lambda = \frac{0.5+0.3}{1} = 0.8$. Using the last gene for division purposes has the advantage that it can return small or big λ values, depending on what is necessary. For instance, if the third gene in the above example was 0.05, then $\lambda = \frac{0.5+0.3}{0.05} = 16$. As a result, we do not need to worry about

[1] This example assumes that $\tau_S \leq \tau_B$. Nevertheless, the same principle would take place if $\tau_B \leq \tau_S$.

increasing or decreasing the number of genes of individuals if a problem required very small or very large values of λ.

2) Fitness function: During each generation, every individual x from D^B is randomly paired with an individual y from D^S and they negotiate following the procedure described in Section II-C. The fitness value for x and y is defined as the utility for that agent:

$$f(x) = U^x = w_p \times U_p^x + w_s \times +U_s^x \quad (7)$$

3) Selection method: In every generation, the D^B and D^S populations are sorted according to their fitness (the individual with the highest fitness is placed at the top). Then, the top N individuals are selected to be saved in a new subpopulation (d^B and d^S, respectively).

4) Estimation of the probability distribution: After the creation of the sorted subpopulations d^B and d^S, new populations D^B and D^S are going to be sampled from a probability distribution. As in [15], a normal distribution is assumed for each gene of the λ strategies. Thus, there is one probability distribution from which all first genes of the strategies are going to be sampled, another distribution for all second genes, and another one for all third genes.[2] The probability distribution $f^B(X)$ for each X gene learnt from d^B can be estimated as follows:

$$f^B(X) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(X-\mu)^2}{2\sigma^2}}$$

Thus, to estimate $f^B(X)$ is to estimate the value of μ and σ. In each generation, the maximum likelihood estimates of μ and σ are determined by:

$\hat{\mu} = \frac{1}{N}\sum_{r=1}^{N} \lambda_r$, and $\hat{\sigma} = \sqrt{\frac{1}{N}\sum_{r=1}^{N}(\lambda_r - \hat{\mu})^2}$

The same method is used to estimate $f^S(X)$ for X from d^S.

5) Stopping criteria: The EDA algorithm stops once the maximum number of generations has been reached. Figure 1 summarizes the EDA process.

B. Genetic Algorithm

The GA follows the same individual representation, fitness function, and stopping criteria, as the EDA above. The difference is in the selection method, where tournament is used. In addition, elitism, one-point crossover and one-point mutation are the three genetic operators used during the evolutionary procedure.

C. Genetic Programming

1) Individual representation: The idea behind the use of GP is to use the flexibility of its structure and calculate λ as a representation of the problem parameters, instead of just creating different λ values, as it happens with the EDA and GA. This has the advantage that λ *is now calculated dynamically*. As a result, the GP's representation and behavior is not affected by changes in the negotiation problem parameters, such as changes in the values of τ_b or τ_S. Since the representation is in the form of the problem parameters, the same evolved trees could be used even if the above values had changed. This, on the other hand, could not happen with the EDA and the GA, where a new λ value would have to be evolved every time.

To further explain this, we present Equation 8, which is the strategy that allows Buyer B to achieve maximum utility, as it was proven in [22].[3] Thus, B achieves maximum utility when it adopts the strategy

$$\lambda_B = log_{\frac{\tau_S}{\tau_B}} \frac{RP_S - IP_B}{RP_B - IP_B} \quad (8)$$

Similarly, S achieves maximum utility when it adopts the strategy

$$\lambda_S = log_{\frac{\tau_B}{\tau_S}} \frac{IP_S - RP_B}{IP_S - RP_S} \quad (9)$$

Hence, we are interested in creating λ values that are a direct result of the above parameters found in the optimal strategies in Equations 8 and 9. Thus, the GP terminal set consists

Begin
Pop D^B <= **InitializePopulation (Pop);** */* Randomly create a population of strategies λ for Buyer */*
Pop D^S <= **InitializePopulation (Pop);** */* Randomly create a population of strategies λ for Seller */*
Evaluate (D^B)**;** */* Calculate fitness of each strategy λ in* D^B **/*
Evaluate (D^S)**;** */* Calculate fitness of each strategy λ in* D^S **/*
Repeat
  Select (d^B)**;** */* New subpopulation with* $N \leq M$ *individuals from* D^B *via the selection method. M is the population size of* D^B*. */*
  Select (d^S)**;** */* New subpopulation with* $N \leq M$ *individuals from* D^S *via the selection method. M is the population size of* D^S*. */*
  Estimate ($f^B(X)$)**;** */* Estimate the probability distribution* $f^B(X)$ *of each* X *gene from* d^B*. */*
  Estimate ($f^S(X)$)**;** */* Estimate the probability distribution* $f^S(X)$ *of each* X *gene from* d^S*. */*
  Create (D^B)**;** */* Create new population* D^B *of* M *individuals sampled from* $f^B(X)$*. */*
  Create (D^S)**;** */* Create new population* D^S *of* M *individuals sampled from* $f^S(X)$*. */*
Until (TerminationCondition( )) */* Determine if we have reached the last generation, and return the best individual (highest fitness) from* D^B *and* D^S*, respectively. */*
End

Fig. 1. Pseudo code for the procedure that the EDA follows.

[2]The terms 'first', 'second', and 'third' gene refer to the position of the gene in an individual λ strategy. For instance, if we have the individual [0.5 0.3 1], then the first gene refers to the value 0.5, the second gene to the value 0.3, and the third gene to the value 1.

[3]This theoretical optimum can only be calculated for Price-only optimization problems, and not if Speed optimization is involved. Nevertheless, such a theoretical optimum can act as a very useful experimental benchmark.

of all of the above problem parameters: IP_B, IP_S, RP_B, RP_S, τ_B, and τ_S. Furthermore, the function set consists of the following functions: ADD (Addition), SUB (Subtraction), MUL (Multiplication), DIV (Division), and LOG (Logarithm).

Therefore, the GP can, through its evolutionary process, create complex trees that represent the trading strategies of the agents, and also return trees similar to the ones from Equations 8 and 9. Figure 2 presents an example of a GP individual, which is essentially the GP representation of Equation 9.

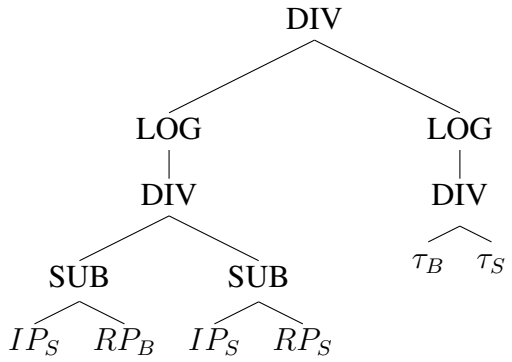

Fig. 2. Sample tree generated by the GP

The rest of the GP algorithm is the same as the GA. The fitness function is calculated in exactly the same way as the GA's fitness (see Equation 7). Also, tournament is again used as the selection method, and the GP operators are again elitism, one-point crossover and one-point mutation. The GP algorithm stops once the maximum number of generations has been reached. Figure 3 summarizes the process of the GA and GP algorithms.

IV. Experimental Setup

The purpose of the experiments is to evaluate the performance of the EDA, GA, and GP algorithms, in co-evolving negotiation strategies for bilateral negotiation between two agents B and S, where both B and S have incomplete information (they do not know each other's initial price, reserve price, deadline, and strategy). For statistical purposes, each algorithm is run for 50 times. The EDA, GA and GP parameters are presented in Table I. The first part of the table presents the evolutionary parameters that are present in all three algorithms. The second part refers to EDA only, and informs us that the subpopulations are going to be the top 50% (in terms of fitness) of the D^B and D^S populations. The value of 50% was selected after experimenting with 10 different percentages, from 10-100%. Furthermore, the third part of the table presents parameters that are present in the GA and the GP. The value of these parameters was again selected through experimental tuning. Lastly, the bottom part of the table presents the function and terminal sets, and also the maximum depth of the trees. These parameters are GP-related only.

In addition, Table II presents the experimental parameters related to the negotiation problem. As we can observe, the initial and reserve prices are set to $IP_B = 5, RP_B = 85, IP_S = 95$, and $RP_S = 15$, respectively. Through experimental tuning, a deadline of around 30-35 iterations is considered short, whereas a deadline of around 100 iterations is considered long. As it is noted in [15], for deadlines below 30 time units, agents do not have sufficient time to reach agreements; also, for deadlines longer than 100 time units, the negotiation outcomes are generally similar to those obtained for deadlines in range [90,100]. Therefore, in this paper, we are using deadlines in the range from 30 up to 100. We consider three different cases: (i) B has very little bargaining advantage in terms of time over S, and both agents have little time to negotiate ($\tau_B = 35, \tau_S = 34$), (ii) B has a significant bargaining advantage in terms of time over S

Begin
Pop D^B $<=$ **InitializePopulation (Pop);** */* Randomly create a population of strategies λ for Buyer */*
Pop D^S $<=$ **InitializePopulation (Pop);** */* Randomly create a population of strategies λ for Seller */*
Evaluate (D^B)**;** */* Calculate fitness of each strategy λ in* D^B **/*
Evaluate (D^S)**;** */* Calculate fitness of each strategy λ in* D^S **/*
Repeat
Pop D^B $<=$ **Elitism** (D^B) **+ Crossover** (D^B) **+ Mutation** (D^B)**;** */* New population is created after genetic operators of elitism (which reproduces M*PrE individuals), crossover (which creates M*PrC individuals), and mutation (which creates M*(1-PrC) individuals). PrE and PrC denote the elitist and crossover probabilities, respectively. M is the population size. */*
Pop D^S $<=$ **Elitism** (D^S) **+ Crossover** (D^S) **+ Mutation** (D^S)**;**
Evaluate (D^B)**;** */* Calculate the fitness of each strategy λ in* D^B **/*
Evaluate (D^S)**;** */* Calculate the fitness of each strategy λ in* D^S **/*
Until (TerminationCondition( )) */* Determine if we have reached the last generation, and return the best individual (highest fitness) from* D^B *and* D^S*, respectively. */*
End

Fig. 3. Pseudo code for the procedure that the GA and the GP follow.

TABLE I
Experimental Parameters for EDA, GA, and GP

Parameter	Value
Generations (all)	100
Population (all)	1000
Subpopulation Percentage (EDA)	0.5
Tournament Size (GA and GP)	4
Crossover Probability (GA and GP)	0.9
Mutation Probability (GA and GP)	0.1
Elitism Probability (GA and GP)	0.01
Function Set (GP)	ADD, SUB, MUL, DIV, LOG
Terminal Set (GP)	$IP_B, IP_S, RP_B, RP_S, \tau_B, \tau_S$
Maximum Depth (GP)	4

TABLE II
NEGOTIATION PARAMETERS

Parameter	Value
IP_B	5
RP_B	85
IP_S	95
RP_S	15
Deadlines $[\tau_B\ \tau_S]$	[35 34] , [100 30] , [100 95]
Weights $[w_p\ w_s]$	[0.1 0.9], [0.5 0.5], [0.9 0.1], [1 0]
u^P_{min}	0.01
u^S_{min}	0.1

($\tau_B = 100, \tau_S = 30$), and (iii) B has very little bargaining advantage in terms of time over S, and both agents have plenty of time to negotiate ($\tau_B = 100, \tau_S = 95$). Lastly, we are interested in testing the above deadlines under different utility function weight arrangements. Thus, each of the above three deadlines is tested under four different pairs of price and speed weight, w_p and w_s, respectively: (i) Both B and S are speed-optimising agents, and very little emphasis is given on price-optimization ($w_p = 0.1, w_s = 0.9$),[4] (ii) B and S are price-speed optimizing agents, placing equal emphasis on both price and speed ($w_p = 0.5, w_s = 0.5$), (iii) Both B and S are price-optimizing agents, and very little emphasis is given on speed optimization ($w_p = 0.9, w_s = 0.1$), and (iv) Both B and S are price-optimizing agents, and no emphasis is given on speed ($w_p = 1, w_s = 0$).

V. EMPIRICAL RESULTS

Tables III-V present the results of the three algorithms over the different deadlines of $[\tau_b, \tau_S]$ = [35, 34], [100, 30], and [100, 95]. Each table is going to be looking into the four different Price-Speed weight settings.

A. Case 1: $[\tau_b, \tau_S]$ = [35, 34]

In this scenario, the buyer agent has a very slight bargaining advantage over the seller agent in terms of deadline and both agents have very little time for negotiation. Table III presents the mean and the max results of λ_B and Fitness of Buyer.[5] Let us first start with the case where $[w_p, w_s] = [1, 0]$, where we are only interested in Price optimization. In this case, we can can compare the table's result with the theoretical optimum, which can be calculated from Equation 8:

$$\lambda_B = log_{\frac{\tau_S}{\tau_B}} \frac{RP_S - IP_B}{RP_B - IP_B} = log_{\frac{34}{35}} \frac{15-5}{85-5} = 71.736$$

As we can observe from the max values, all three algorithms have managed to find the optimal $\lambda_B = 71.736$.

Let us now move to the results under the different weights. Since now the optimization is P-S, we cannot refer to the optimal λ to judge the results. Thus, we have to check the mean and max values of the Fitness.[6]

In terms of mean results, we compared the average fitness of each algorithm, for each weight scheme, under one-way ANOVA test. The software we used for this test was MATLAB, where we used the built-in $anova1$ command. The test's significance level was 5%. The null hypothesis H_0 was that all three groups (fitness values over 50 runs of EDA, GA, GP) have a common mean; the alternative H_1 was that they do not. In cases where the null was rejected, we had to further investigate which algorithm was better. In order to do this, we used the multiple comparison tests, which are also provided by MATLAB, via the built-in command $multcompare$. Therefore, when an algorithm is significantly better (at 5% level) than at least one more algorithm, we denote this in Table III by putting its average fitness in bold fonts. When an algorithm is significantly better than both of the other two algorithms, then we put its average fitness value in both bold and underlined fonts. For instance, in the case of $[w_p, w_s] = [0.1, 0.9]$, the GA's fitness value is in bold font, and the GP's fitness is both bold and underlined. This means that they are both significantly better than the EDA's average fitness. In addition, the GP is also significantly better than the GA.[7]

What we can thus observe from the *mean* results of Table III, is in most cases the GP performs as well as the GA and the EDA. More specifically, there are no significant differences for the three algorithms, for the cases of $[0.1, 0.9], [0.9, 0.1]$, and $[1, 0]$. On the other hand, in the case of $[w_p, w_s] = [0.5, 0.5]$, we can observe that not only the GA outperforms the EDA, but also that the GP outperforms both of the GA and EDA (at 5% significance level).

With respect to the *max* values, we follow the same presentation principle, of bold and underlined values. The difference here is that since we are dealing with best values, we cannot use a statistical test. Thus, for the purposes of this paper we consider an improvement as important when there is an increase of at least 1%. As we can observe from Table III, GP returns either as good results as the other two algorithms, or improved results. In the $[0.5, 0.5]$ scenario, the GP outperforms again both the GA and EDA, while the GA only outperforms the EDA. What is worth-mentioning here is the scale of the improved results, which is quite significant: the GP has improved the best fitness of EDA and GA (0.5422 and 0.7399, respectively) to 0.8984. That's quite remarkable, as it is an improvement of about 36 and 16%, respectively. Lastly, the GP outperforms both the GA and EDA in the $[0.9, 0.1]$ scenario.

B. Case 2: $[\tau_b, \tau_S]$ = [100, 30]

The buyer agent has a significant bargaining advantage over the seller in terms of deadline. Table IV presents the results of

[4]Setting $w_s = 1$, thus having the agents not being interested about price optimization, is unrealistic; for that reason we are not interesting in testing this scenario.

[5]Since Buyer has the negotiation advantage, we are only interested in his results. The same principle applies to the rest of our paper, since Buyer always has the negotiation advantage.

[6]As explained earlier in Section III, the theoretical optimum can only be calculated for Price-only optimizing negotiation.

[7]The same principle of setting the significant improved results in bold and underlined fonts applies to the rest of the tables of this paper.

TABLE III
RESULTS UNDER $[\tau_B, \tau_S] = [35, 34]$

		EDA		GA		GP	
$[w_p, w_s]$		λ_B	Fitness	λ_B	Fitness	λ_B	Fitness
$[0.1, 0.9]$	Mean	0.6216	0.9138	0.6963	0.9171	0.7158	0.9178
	Max	0.6907	0.9169	0.7242	0.9181	0.7264	0.9182
$[0.5, 0.5]$	Mean	70.307	0.4892	3.0381	**0.5846**	8.0331	**0.6319**
	Max	71.736	0.5422	71.726	**0.7399**	71.736	**0.8984**
$[0.9, 0.1]$	Mean	71.736	0.7986	71.726	0.7986	66.138	0.7993
	Max	71.736	0.7986	71.736	0.7986	71.736	**0.812**
$[1.0, 0.0]$	Mean	71.736	0.8762	71.728	0.8762	71.735	0.8762
	Max	71.736	0.8762	71.736	0.8762	71.736	0.8762

this negotiation scenario. Let us again first examine the case where $[w_p, w_s] = [1, 0]$. The optimal lambda is:

$$\lambda_B = log_{\frac{\tau_S}{\tau_B}} \frac{RP_S - IP_B}{RP_B - IP_B} = log_{\frac{30}{100}} \frac{15-5}{85-5} = 1.7271$$

As we can observe from the table, the three algorithms have again managed to find the optimal λ. Moreover, in the rest of the results the three algorithms are performing equally well. The only exception is again the $[w_p, w_s] = [0.5, 0.5]$ scenario. Here the GA significantly outperforms the average fitness of the EDA. In addition, the GP significantly outperforms the average fitness of both GA and EDA. It is also worth noting that the GP's average fitness has approximately increased the EDA and GA's fitness by 11% and 8%, respectively. Both of these improvements should be considered as significant.

Lastly, there is a similar picture for the best results, where the GP is again performing as well as the other two algorithms for the cases of $[0.1, 0.9]$, $[0.9, 0.1]$ and $[1, 0]$, and significantly outperforms both the EDA and GA for $[w_p, w_s] = [0.5, 0.5]$. Also, both the GA and GP have improved the EDA's best fitness by approximately 13%.

TABLE IV
RESULTS UNDER $[\tau_B, \tau_S] = [100, 30]$

		EDA		GA		GP	
$[w_p, w_s]$		λ_B	Fitness	λ_B	Fitness	λ_B	Fitness
$[0.1, 0.9]$	Mean	0.4929	0.9613	0.5027	0.9618	0.5271	0.9631
	Max	0.5180	0.9627	0.5285	0.9632	0.5313	0.9633
$[0.5, 0.5]$	Mean	1.7271	0.7986	1.3245	**0.8273**	0.5366	**0.9094**
	Max	1.7271	0.7986	1.7271	**0.9224**	1	**0.9217**
$[0.9, 0.1]$	Mean	1.7271	0.8607	1.7269	0.8607	1.7257	0.8605
	Max	1.7271	0.8607	1.7271	0.8607	1.7271	0.8607
$[1.0, 0.0]$	Mean	1.7271	0.8762	1.7270	0.8762	1.7259	0.8761
	Max	1.7271	0.8762	1.7271	0.8762	1.7271	0.8762

C. Case 3: $[\tau_b, \tau_S]$ = [100, 95]

The buyer agent has a slight bargaining advantage over the seller agent in terms of deadline. Also, both agents have plenty of time to negotiate. Table V presents the results of this negotiation scenario. Let us again first examine the case where $[w_p, w_s] = [1, 0]$. The optimal lambda is:

$$\lambda_B = log_{\frac{\tau_S}{\tau_B}} \frac{RP_S - IP_B}{RP_B - IP_B} = log_{\frac{95}{100}} \frac{15-5}{85-5} = 40.5402$$

We can see that none of the three algorithms has found this λ. However, this only happens because the price utilities resulting from the optimal λ above and the λ of the GP, are exactly the same. To be more specific: under the $\lambda_B = 40.5402$, our experiments showed that the agreement will take place in round 95, and the agreed price will be the $RP_S = 15$. Interestingly enough, the GP's best $\lambda_B = 33.607$ will result into an agreement in round 94; of course, since we are only optimizing price, reaching to an agreement one round in advance does not affect the agent's utility. What is of interest, however, is the fact that the agreed price is also 15. We can see that the two agreed prices are the same, which thus explains why the utilities are the same. The offered prices for the Buyer are demonstrated at the Appendix of this paper.

As a result, both all three algorithms seem to be focusing on λ values around 33.6, which *gives the highest fitness anyway*. We can thus conclude that all three algorithms can find λ values that result to a fitness equal to the optimal one.

Regarding the other price and speed weight parameters, there is a similar picture as before, in Case 1: $[\tau_b, \tau_S]$ = [35, 34]. In terms of mean results, there are no significant differences in the cases of $[w_p, w_s] = [0.1, 0.9]$ and $[0.9, 0.1]$. The case of $[0.5, 0.5]$ returns very interesting results, with the GP outperforming both the EDA and GA by 26% and 13%, respectively.

Lastly, in terms of max results, the $[0.5, 0.5]$ scenario returns again very impressive results. More specifically, the EDA's best fitness is only 0.68. On the other hand, the GA has had a best fitness of 0.8984, and the GP a best fitness of 0.9131. These are improvements of about 30%. Lastly, both the GA and GP outperform the EDA in the $[0.9, 0.1]$ scenario.

TABLE V
RESULTS UNDER $[\tau_B, \tau_S] = [100, 95]$

		EDA		GA		GP	
$[w_p, w_s]$		λ_B	Fitness	λ_B	Fitness	λ_B	Fitness
$[0.1, 0.9]$	Mean	0.4791	0.9605	0.5039	0.9619	0.5244	0.963
	Max	0.5151	0.9625	0.529	0.9632	0.5307	0.9633
$[0.5, 0.5]$	Mean	32.945	0.5096	3.4544	**0.6172**	1.5335	**0.7644**
	Max	30.607	0.68	33.604	**0.8984**	18	**0.9131**
$[0.9, 0.1]$	Mean	33.607	0.8022	32.311	0.8042	23.983	0.8093
	Max	33.607	0.8022	33.607	**0.8634**	33.607	**0.8663**
$[1.0, 0.0]$	Mean	33.607	0.8762	33.603	0.8762	33.606	0.8762
	Max	33.607	0.8762	33.607	0.8762	33.607	0.8762

VI. DISCUSSION

The above allow us to argue that all three algorithms are doing extremely well; however, it seems that overall the GP outperforms both the EDA and the GA. It should also be noted that the max results are of particular interest to us, because in real-life situations we would be running the

algorithms multiple times, and then select the best result as the agents' trading strategy. Thus, while average results are always important for research comparison purposes, best/max results have real-life value.

To get a clearer picture of these results, we present Tables VI and VII, which are 'league' tables for the mean and best values of the three algorithms. For every scenario from Sections V-A, V-B and V-C, we compare the three algorithms. The algorithm with the significantly[8] highest fitness receives 2 points, and the second highest receives 1 point. No points are allocated for the worse algorithm. For instance, in the case of $[\tau_B, \tau_S] = [35, 34]$, $[w_p, w_s] = [0.5, 0.5]$, the EDA's best fitness is 0.5422, the GA's 0.7399, and the GP's 0.8984. Thus, the GP would get 2 points in this instance, the GA 1 point, and the EDA 0 (zero) points. We apply the same principle to all scenarios presented in the previous section. Lastly, in cases where two algorithms have the same fitness value, they both receive 2 points each. When all three algorithms have the same fitness, then they all receive 2 points.

TABLE VI
LEAGUE TABLE FOR MEAN RESULTS

Algorithm	Points
EDA	16
GA	21
GP	24

TABLE VII
LEAGUE TABLE FOR BEST RESULTS

Algorithm	Points
EDA	14
GA	18
GP	22

Tables VI and VII summarize what we saw earlier in Section V: the GP is the best algorithm in both mean and max results. The GA takes the second position in both mean and max results, and the EDA is last.

Furthermore, it should be reminded that all three algorithms were able to find the theoretical optimum λ value, under the Price-only optimizing scenarios, for the cases of $[\tau_B, \tau_S] = [35, 34]$ and $[\tau_B, \tau_S] = [100, 30]$. Also, as we explained earlier, in the $[100, 95]$ scenario, even though none of the algorithms was able to find the optimal strategy, all three algorithms were able to locate another soluction, which *returned equally high fitness*, as the one of the theoretical optimum. Thus, we can conclude that the GP, as well as the other two algorithms are able to find optimal or near-optimal solutions.

The above results are very important, because they suggest that the GP is able to not only find optimal/near-optimal solutions, but also *outperform the only other published algorithms in the literature*, namely the EDA and the GA. Furthermore, in some cases the GP was able to show impressive improvements, particularly in the scenario of $[w_p, w_s] = [0.5, 0.5]$ (e.g., improvements in the range of 15%-30%). It seems that because in this scenario the agents have to put equal efforts for optimizing Price and Speed, the EDA and the GA have difficulties doing this in an effective manner. This is likely because of the nature of the search space, which under the [0.5, 0.5] scenario is much more complex than in the other scenarios. On the other hand, we believe that the GP's flexible representation has allowed it to search the space in such ways that can result to such highly performing trading strategies. The above is of particular significance, because the $[w_p, w_s] = [0.5, 0.5]$ scenario is a true Price *and* Speed optimization problem. In all other scenarios, where higher weight was given in either Price *or* Speed optimization, all algorithms have more or less managed to do similarly well. However, the actual difficulty lies with the task of optimizing both Price and Speed. As it was explained at the beginning of this paper, while there are numerous works that have focused on optimizing Price, it's only lately that it has been argued that Speed is equally important. And as we have seen, in such an important problem the GP is able to offer solutions of great quality. For that reason, we believe that the GP is an extremely valuable tool for the problem of Price-Speed optimization. Lastly, it is also important to note that the GP was never outperformed in neither mean nor max results, by any of the EDA or the GA. This is also significant, because *it demonstrates the overall superiority of the GP in the experiments of this paper*.

One final note that we would like to make is that we are aware that there are different schools and ways to implement the above three algorithms, which could potentially lead to different results. Nevertheless, this falls outside the scope of this paper. We are not arguing an overall superiority of the GP over the EDA and GA. Our claim is that the GP, as a result of its flexible representation, is able to outperform the only other published works (i.e., EDA and GA) on the problem of Price-Speed negotiating agents. For this reason, the GP should be considered as a very useful algorithm in the above field and more research should take place towards this direction.

VII. CONCLUSION

To conclude, this paper applied Genetic Programming for Price-Speed (P-S) optimizing negotiation. P-S optimizing negotiation is a very important category of bargaining, with application in real-life situations, such as Grid computing environments. In this type of negotiations, we are not only interested in finding the optimal price for the participating agents, but also to ensure that the agreement takes place as soon as possible.

P-S optimizing negotiation is an area that has not received much attention yet. This work was the first to apply GP to co-evolve trading strategies. The novelty that the GP offers is that it can represent solutions as part of the experimental parameters, rather than just producing arbitrary real numbers, as other methods have done until now.

[8]Regarding significance, the principles explained earlier in Section V apply: for mean results, we have used the ANOVA and multicomparison tests to specify which algorithms are significantly better than others. Regarding the best results, the rule of an improvement of at least 1% applies.

We compared the GP with an EDA and a GA, the only other algorithms in the literature that had been used for P-S optimizing negotiation. Experiments took place over a number of different scenarios, and results showed that the GP can always perform at least as good as the other two algorithms. In addition, in many situations it can even outperform them. Lastly, the GP was able to find the optimal or near-optimal solutions. These results, in addition to the advantages of the GP representation, allow us to characterize this algorithm as a significant contribution to the field of Price-Speed optimizing negotiation.

Future work will focus on experiments with more complex setups, as one-to-many and many-to-many negotiating agents. This paper, being our first attempt with a GP, focused on bilateral negotiations only (one buyer, one seller). However, it is important to experiment with more complex scenarios, because this will make our framework more realistic and offer us better understanding of the bargaining problem when dealing with Grid computing environments.

APPENDIX

Here we demonstrate the bargaining process for Buyer, under two different λ_B values. The first value, which was calculated from the theoretical optimum, is 40.5402. The second value, which was the GP's evolved result, is 33.60691642865712, which can be re-written to 33.607; the latter has been used in this paper for presentation purposes. Nevertheless, it should be noted that the experiments used the full number and not its "shortened" version.

TABLE VIII
NEGOTIATION PRICES OF BUYER (P_B), UNDER TWO DIFFERENT λS: 40.5402, AND 33.607.

Time	$\lambda_B = 40.5402$	$\lambda_B = 33.607$
0	5	5
1	5	5
2	5	5
...	...	...
92	7.7230	9.8541
93	9.2207	11.9807
94	11.5116	15
95	15	19.2708

As we can observe, both λs result to an offer of Seller's Reserve Price $RP_S = 15$; thus, even though $\lambda_B = 33.607$ is not the theoretical optimum, it can return exactly the same fitness as the theoretical optimum ($\lambda_B = 40.5402$), for Price-only optimizing agents.

REFERENCES

[1] J. F. Nash, "The bargaining problem," *Econometrica*, vol. 18, pp. 155–162, 1950.

[2] M. J. Osborne and A. Rubenstein, "Tbargaining and markets," *Academic*, 1990.

[3] A. Rubenstein, "A bargaining model with incomplete information about time preferences," *Econometrica*, vol. 53 (5), pp. 1151–1172, 1985.

[4] ——, "Perfect equilibrium in a bargaining model," *Econometrica*, vol. 50 (1), pp. 97–109, 1982.

[5] A. Lomuscio, M. Wooldrigde, and N. Jennings, "A classfication scheme for negotiation in electronic commerce," *Group Decision and Negotiation*, vol. 12, pp. 31–56, 2003.

[6] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge, "Automated negotiation: Prospects, methods and challenges," *Group Decision and Negotiation*, vol. 10 (2), pp. 199–215, 2001.

[7] K. Sim and C. Choi, "Agents that react to changing market situations," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 33 (2), pp. 188–201, 2003.

[8] K. Sim, "A market-driven model for designing negotiation agents," *Computational Intelligence*, vol. 18 (4), pp. 618–637, 2002.

[9] ——, "G-Commerce, market-driven G-negotiation agents and grid resource management," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36 (6), pp. 1381–1394, 2006.

[10] R. Lawley, M. Luch, K. Decker, and T. Payne, "Automated negotiation between publishers and consumers of grid notifications," *Parallel Processing Letters*, vol. 13 (4), pp. 537–548, 2003.

[11] B. Rubenstein-Montano and R. Malaga, "A weighted sum genetic algorithm to support multiple-party multi-objective negotiations," *IEEE Transactions on Evolutionary Computation*, vol. 6 (4), pp. 366–377, 2002.

[12] R. Lau, M. Tang, O. Wong, S. Milliner, and Y.-P. Chen, "An evolutionary learning approach for adaptive negotiation agents," *International Journal of Intelligent Systems*, vol. 21 (1), pp. 41–72, 2006.

[13] N. Jin, "Equilibrium selection by co-evolution for bargaining problems under incomplete information about time preference," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2005, pp. 2661–2668.

[14] N. Matos, C. Sierra, and N. Jennings, "Determining successful negotiation strategies: an evolutionary approach," in *Proceedings of the International Conference on Multi-Agent Systems (ICMAS)*, 1998, pp. 182–189.

[15] K. M. Sim, "An evolutionary approach for p-s-optimizing negotiation," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 1364–1371.

[16] J. Gwak and K. M. Sim, "Co-evolving best response strategies for p-s-optimizing negotiation using evolutionary algorithms," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2010.

[17] F. Herrera, M. Loxano, and J. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, pp. 265–319, 1998.

[18] K. Deb, K. Sindhya, and T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the 9th annual conference on Genetic and Evolutionary Computation*, 2007, pp. 1187–1194.

[19] N. Jin, "Constraint-based co-evolutionary genetic programming for bargaining problems," Ph.D. dissertation, Department of Computer Science, University of Essex, 2007.

[20] J. Koza, *Genetic Programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press, 1992.

[21] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[22] K. M. Sim, Y. Guy, and B. Shi, "Adaptive bargaining agents that negotiate optimally and rapidly," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007, pp. 1007–1014.