9 Heuristic approaches

In the previous chapters, we saw that finding optimal offers can be computationally expensive, especially in the context of solving the trade-off problem for the PDP and for finding optimal agendas. Because software agents have limited computational resources at their disposal, heuristic approaches can be useful for the design of negotiating agents in such situations. In such approaches, a heuristic is used to generate counter offers that are "good enough" (i.e., close to optimum) but not necessarily optimal.

Dealing with the bounded rationality of agents is just one of the reasons for using a heuristic approach. Another reason is that an opponent might not always play equilibrium strategies. In such cases, a strategy that is a best reply to the opponent's strategy must be played rather than an equilibrium one.

Irrespective of the reason for using this approach, heuristics can be designed for a number of purposes. For example, they can be designed to generate optimal counter offers, to predict information about an opponent, or to find optimal agendas. Research into the design of heuristics has been growing continuously and several different approaches – including local search methods, approximation methods, and machine learning methods – have been adopted. Below, we introduce a number of representative works in terms of their key underlying ideas, goals, and main results.

9.1 Generating counter offers

The approach used for generating counter offers will depend on the negotiation protocol. For bilateral negotiations, the alternating offers protocol (Osborne and Rubinstein, 1994) is by far the most general and widely studied protocol. In previous chapters, we provided a strategic analysis of this protocol. Here, we study a number of heuristic approaches for generating counter offers for this protocol.

In the context of utility-maximising agents, counter offers must be generated to maximise individual agent utility. We have seen how an agent's utility from an agreement depends not just on the agreement but also on the time at which it is reached. It is therefore crucial to take this dependence into consideration for generating counter offers that maximise utility. To this end, Faratin *et al*. (1998) designed a time-dependent heuristic model based on *negotiation decision functions*. In terms of these functions, they defined three types of strategies for counter-offer generation:

1. time-dependent strategies;
2. resource-dependent strategies;
3. behaviour-dependent or imitative strategies.

A time-dependent strategy is defined as a function that takes "time" as an input parameter and returns an offer in such a way that concessions are small (big) at the beginning of negotiation but increase (decrease) as the deadline approaches. A family of such functions is defined that vary the concessions made during the course of negotiation. Thus, an agent can use one of these functions as a negotiation strategy. For example, let $j \in \{1, 2, \dots, m\}$ be an issue under negotiation and assume that it represents the “price” of a product. Then, the price offered by Agent a at time t is given by the following equation:

$$x^j_{a\to b}(t) = \begin{cases} \min^a_j + \alpha^a_j(t)(\max^a_j - \min^a_j) & \text{if } a\text{'s utility decreases with price} \\ \min^a_j + (1-\alpha^a_j(t))(\max^a_j - \min^a_j) & \text{if } a\text{'s utility increases with price} \end{cases}$$

where $\max^a_j$ and $\min^a_j$ denote Agent a’s reserve prices. In the above equations, $\alpha^a_j(t)$ is a function such that $0 \le \alpha^a_j(t) \le 1$, $\alpha^a_j(0) = \kappa^a_j$ (Agent a’s initial offer is defined in terms of κ^a), and $\alpha^a_j(t^a_{max}) = 1$. So at the beginning of negotiation (i.e., $t = 0$), the offer will be a point in the acceptable region $(\min^a_j, \max^a_j)$, and when the deadline (t^a_{max}) is reached, the offer will be her reserve price $\max^a_j$. For example, in negotiation over price, the buyer’s acceptable price region might be (£0, £20) and she might start negotiation by offering £10. The seller might have the range (£15, £30) and she might make an initial counter offer of £25.

A family of time-dependent strategies can be defined by varying the definition of the function $\alpha^a_j(t)$. For instance, $\alpha^a_j(t)$ could be defined as a polynomial function parameterised by $\beta \in \mathbb{R}_+$ as follows:

$$\alpha^a_j(t) = \kappa^a_j + (1-\kappa^a_j)(\min(t, t^a_{max})/t^a_{max})^{1/\beta}.$$

An infinite number of possible strategies can be obtained by varying β. One can easily verify that, for small β (such a strategy is called *Boulware* (Pruitt, 1981; Raiffa, 1982)), the initial offer is maintained until time almost approaches the deadline and then rapid concessions are made to the reserve value. On the contrary, for large β (such a strategy is called *Conceder*), the offer quickly goes to the reserve value. The parameter β can also be adjusted suitably to obtain a *linear* strategy. These strategies are illustrated in Figure 9.1.

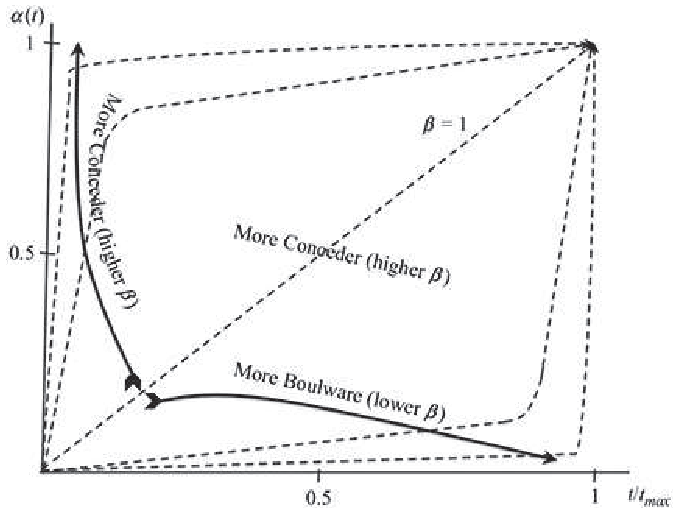

Figure 9.1 Conceder, linear, and Boulware strategies.

Resource-dependent strategies generate counter offers depending on how a particular resource is being consumed. These strategies are similar to timedependent ones; a time-dependent strategy can be seen as a type of resourcedependent strategy in which the only resource-considered is time. Whereas time vanishes constantly up to its end, other resources may have different patterns of usage. Resource-dependent strategies are modelled in the same way as time-dependent ones, that is, by using the same functions, but with one of the following two variations:

1. Making the value of t^a_{max} dynamic. A dynamic value of t^a_{max} represents a heuristic about how many resources are in the environment. The scarcer the resource, the more urgent the need for an agreement. For example, in some application domains, the most important resource to model will be the number of agents negotiating with a given agent and how keen they are to reach agreements. On the one hand, the greater the number of agents who are negotiating with Agent *a* for a particular service *s*, the lower the pressure on *a* to reach an agreement with any specific individual. On the other hand, the longer the negotiation, the greater the pressure on *a* to come to an agreement. Hence, representing the set of agents negotiating with Agent *a* at time *t* as: $NA(t) = \{i | \mathscr{X}^t_{i \leftrightarrow a}$ is active$\}$, the dynamic time deadline for Agent *a* is defined as follows:

$$t^a_{max} = \tau^a \frac{|NA^a(t)^2|}{\sum_i |\mathscr{X}^t_{i \leftrightarrow a}|},$$

where τ^a denotes the time Agent *a* considers reasonable to negotiate with a single agent and $|\mathscr{X}^t_{i \leftrightarrow a}|$ the length of the negotiation between *i* and *a*. Here, time is directly proportional to the number of agents and inversely proportional to the average length of negotiation.
2. Making the function α depend on an estimation of the amount of a particular resource. These strategies generate counter offers depending on how a particular resource is being consumed. A resource could be money transferred among agents, the number of agents interested in a particular negotiation, or, in a similar way as before, time. One might want an agent to become progressively more conciliatory as the quantity of resource diminishes. The limit when the quantity of the resource approaches nil is to concede up to the reservation value for the issue(s) under negotiation. When there is plenty of resource, a more Boulware behaviour is to be expected. Formally, this can be modelled by having a different computation for the function α:

$$\alpha^a_j(t) = \kappa^a_j + (1 - \kappa^a_j) e^{-\text{resource}^a(t)},$$

where the function resource$^a(t)$ measures the quantity of the resource at time *t* for Agent *a*. Some examples of this function are:
    1. resource$^a(t) = NA^a(t)$. Here, the number of negotiating agents is the resource. The more agents negotiating, the less the pressure to make concessions.
    2. resource$^a(t) = \tau^a |NA^a(t)|^2 / \sum_i |\mathscr{X}^t_{i \leftrightarrow a}|$.Here, time is modelled as a resource in a similar way as in the previous case. The more agents, the less pressure and the longer the negotiations, the more pressure.
    3. resource$^a(t) = \min(0, t^a_{max} - t)$. Here too time is modelled as a resource, but in this case the quantity of resource decreases in a linear fashion with respect to time.

Behaviour-dependent strategies are for those situations where there is no pressure in terms of time or resources. Here, a counter offer is computed based on the previous attitude of the negotiation opponent. In other words, an agent simply imitates her opponent's strategy in order to protect herself from being exploited by her opponent. There is a range of behaviour-dependent strategies and the main difference between them is the type of imitation they perform. One type imitates proportionally, another in absolute terms, and the last one computes the average of the proportions in a number of previous offers. Hence, given a sequence of offers $(\ldots, x_{b\to a}^{t_{n-2\theta}}, x_{a\to b}^{t_{n-2\theta+1}}, \ldots, x_{a\to b}^{t_{n-1}}, x_{b\to a}^{t_n})$, with $\theta \geq 1$, the following are some of the possible strategies:

1. *Relative tit-for-tat*. Here, an agent imitates, in percentage terms, her opponent's behaviour $\theta \geq 1$ steps ago to generate the following counter offer:

$$x_{a\to b}^{t_{n+1}}(j) = min\left(max\left(\frac{x_{b\to a}^{t_{n-2\theta}}(j)}{x_{b\to a}^{t_{n-2\theta+2}}(j)} x_{a\to b}^{t_{n-1}}(j), min_j^a \right), max_j^a \right).$$

   In order to apply this strategy, we must have $n \geq 2\theta$ (see index for t).
2. *Random absolute tit-for-tat*. The same as before but in absolute terms. This means that if the other agent decreases her offer by say £2, then the next response should be to increase by the same amount. Here, a random component may be added that modifies that behaviour by increasing or decreasing (depending on the value of parameter s) the offer by a random amount. This can be done to enable an agent to escape from local minima. $\mathscr{M}$ is the maximum amount by which an agent can change her imitative behaviour. The condition of applicability is again $n \geq 2\theta$:

$$x_{a\to b}^{t_{n+1}}(j) = min\Big(max\Big(x_{b\to a}^{t_{n-1}}(j) + (x_{b\to a}^{t_{n-2\theta}}(j) + x_{b\to a}^{t_{n-2\theta+2}}(j)) + (-1)^s \mathscr{R}(\mathscr{M}), min_j^a \Big), max_j^a \Big),$$

   where

$$s = \begin{cases} 0 & \text{if } a\text{'s utility is decreasing} \\ 1 & \text{if } a\text{'s utility is increasing} \end{cases}$$

   and $\mathscr{R}(\mathscr{M})$ is a function that generates a random integer in the interval $[0, \mathscr{M}]$.
3. *Averaged tit-for-tat*. Here, an agent computes the average of percentages of changes in a window of size $\gamma \geq 1$ of her opponent's history when determining her new offer. When $\gamma = 1$, we have the relative tit-for-tat strategy with $\theta = 1$. The condition of applicability for this strategy is $n \geq 2\gamma$:

$$x_{a\to b}^{t_{n+1}}(j) = min\left(max\left(\frac{x_{b\to a}^{t_{n-2\gamma}}(j)}{x_{b\to a}^{t_n}(j)} x_{a\to b}^{t_{n-1}}(j), min_j^a \right), max_j^a \right).$$

Again, there is a family of functions that differ in terms of the specific aspect of the opponent's behaviour that is imitated and the degree to which it is imitated. See Faratin *et al*. (1998) for more details on these strategies and how they can be combined together.

The above heuristic approaches are mostly suitable for negotiations with monotonic utility functions. For the more general case where there are multiple issues and the utility functions are nonlinear, finding an individual player's optimal offers, especially for the PDP, can be computationally hard. To address this problem, Fatima *et al*. (2009) used an approximation heuristic that replaces the actual nonlinear utility functions with their linear approximations. They then compared the outcomes generated by the PDP and the simultaneous procedure in terms of the utilities of the individual players, the social welfare, and the Pareto optimality.

A similar approximation approach was used by Hindriks *et al*. (2010) to deal with nonlinear utilities and interdependent issues. They provide experimental results to show how these approximations affect the negotiation outcome.

9.2 Predicting opponent preferences and generating counter offers

We have seen that information plays a key role in negotiation. But in many situations, an agent may not have complete information about her opponent. Moreover, in competitive negotiations, an agent might not be willing to reveal information about herself. In such cases, it becomes important for an agent to have the ability to learn about her opponent and make predictions about her behaviour in order to generate offers that are better either from her own perspective (i.e., improve her individual utility) or from a system-wide perspective (i.e., improve social welfare or attain Pareto-optimal outcomes). Information about an opponent can be learnt by observing the offers she rejects and those that she proposes during the process of negotiation. A number of heuristic approaches have been developed for learning and making predictions. The following is an overview of the key approaches from the literature.

For bilateral negotiations over multiple discrete issues, Oliver (1996) showed how to evolve player strategies using *genetic algorithms*. In this approach, although there is no explicit model of the opponent's preferences, a player is able to adapt her strategy to that of her opponent through an evolutionary process. The agreements that result from the evolved strategies are compared and shown to be better than those that result from the strategies used at the beginning of evolution. This comparison is done in terms of the Pareto optimality of the agreements, the utilities they yield to the individual players, and the resulting social welfare.

The strategies used by Oliver (1996) are simple in that an offer is accepted if it exceeds a certain threshold. A more elaborate range of strategies was considered later by Matos *et al*. (1998) in order to handle different types of negotiation settings such as those with time or resource constraints. They used *genetic algorithms* to evolve buyer–seller strategies for service-oriented negotiations. A range of time, resource, and behaviour-dependent strategies were used in the evolutionary process. The strategies that perform best against different types of opponents are shown.

Zeng and Sycara (1998) introduced a general *Bayesian learning* framework for situations where multiple agents bargain over multiple items. Initially, the agents have an *a priori* distribution over their opponents. As negotiation progresses, this *a priori* information is updated and a posterior distribution is computed using Bayes' rules. The approach is shown to improve the players' joint utility measured in terms of a Nash solution (Luce and Raiffa, 1957).

Then, for multi-issue negotiations where the issues are independent, Faratin *et al*. (2002) introduced a heuristic computational model for making trade-offs and generating counter offers that lead to an increased social welfare of the system. Given an agent's beliefs about her opponent's weights for the issues, the algorithm operates by using the notion of *fuzzy similarity* to approximate the opponent's preferences and then employs a *hill-climbing* technique to explore the space of possible trade-offs for the one that is most likely to be acceptable.

Luo *et al*. (2002) developed a new *fuzzy approach* for bilateral multi-issue negotiations. This approach employs the notion of *prioritised fuzzy constraints* to determine what counter offers to send and what offers to accept (Luo *et al*., 2003). The authors demonstrate that the approach leads to Pareto-optimal agreements.

Jonker and Robu (2004) also considered bilateral multi-issue negotiations in the context of independent issues and linear utility functions. For an alternating offers protocol, they showed how information about an opponent's preferences can be learnt on the basis of her offers. They also show how this information can then be used to reach Pareto-optimal agreements.

Coehoorn and Jennings (2004) introduced a method for learning the opponent's weights for different issues using *kernel density estimation*. Kernel density estimation is a statistical method for finding structure in data sets (Wand and Jones, 1995). The authors show how this method can be used to improve the utilities of both parties.

In the context of multiple concurrent multi-issue negotiations between a buyer and multiple sellers, Nguyen and Jennings (2004) introduced a heuristic approach for generating counter offers. Here, the players have deadlines and hence it is assumed that they will play a time-dependent strategy such as linear, Boulware, or Conceder. The players have probabilistic beliefs about the type of strategy used by the opponent and these beliefs are updated using Bayesian updating. A counter offer is then generated to maximise individual agent utility. This heuristic strategy is shown to generate higher utility than a range of other benchmark strategies.

In the context of a single issue, An *et al*. (2009) derived Nash equilibrium strategies for one-to-many and many-to-many negotiations by assuming discrete time and focusing primarily on perfect-information settings.

For multiple concurrent negotiations, Williams *et al*. (2012) extended the work of Nguyen and Jennings (2004) to more realistic situations such as those encountered in the ANAC competition. Williams *et al*. (2012) consider realtime negotiations with uncertainty about payoffs and opponent types and demonstrate that their agent outperforms Nguyen and Jennings (2004) even if the latter is given more prior information than the former.

For a slightly different environment involving six parties, He *et al*. (2006) employed a *fuzzy reasoning* approach for the design of a negotiating agent. The environment is the *Trading Agent Competition for Supply Chain Management* (TAC SCM) (Collins *et al*., 2005; Trading Agent Competition). In this competition, the agents are simulations of small manufacturers, who must compete with each other for both supplies and customers, and manage inventories and production facilities to fulfil customer orders for assembled PCs. On each of the 220 simulation days of the game, agents receive new requests for quotes (RFQs) and actual orders (if offers they have previously sent have won) from customers. From the supplier side, an agent receives offers to deliver particular quantities and types of components at particular prices in response to the RFQs that it sent on the previous day. Thus, in each day of the game, the agent must decide on the following: (i) which new RFQs to send and which offers to accept; (ii) which RFQs to respond to, and at what price; and (iii) how to schedule the production of PCs given the availability of components, the limited capacity of the factory, and the delivery deadlines of pending orders. Here, an agent spends money on buying the components, and earns money by selling PCs. The success of an agent is measured in terms of its profit. The agent designed using fuzzy reasoning (He *et al*., 2006) not only succeeded in the competition (it came second in the semi-finals and was runner-up in the finals), but also performed well in a more controlled experimental setting.

Another heuristic approach is to base the negotiation strategy on the helpfulness of other negotiators. Talman *et al*. (2005) studied negotiations in which players needed to exchange resources to achieve their goals, but did not have information about each others' resources. Furthermore, agreements were not enforceable. They described a model in which an agent's helpfulness is characterised in terms of her cooperation and reliability. An agent chooses a negotiation strategy based on her estimate of the others' degree of helpfulness given the dependency relationships that hold between the agent and others. The proposed model was evaluated in the Coloured Trails (CT) game (Grosz *et al*., 2004; Gal *et al*., 2010) with games of four players. Results showed that agents using the model could identify and adapt to others' varying degrees of helpfulness even while the other agents were constantly changing their strategy. Moreover, the agents that varied their degree of helpfulness depending on their estimate of others' helpfulness outperformed those agents who did not, as well as increased the social welfare of the group.

Other related work that deals with forming heuristics for predicting an opponent's behaviour includes Brzostowski and Kowalczyk (2006). By assuming that the opponent uses a weighted combination of time and behaviourdependent strategies, a prediction is made about her future behaviour. On the basis of this prediction, a counter offer aimed at improving the offering agent's utility is generated.

Then, in the context of single-issue negotiations, Narayanan and Jennings (2006) developed a model for learning the opponent's strategy in the context of *non-stationary* environments, that is, environments where the opponent's strategy changes with time. Here, Markov chains are used to model the process of negotiation and Bayesian learning is used to learn the opponent's strategy. Based on this learning, counter offers are generated to maximise an agent's individual utility. The authors demonstrate that this approach allows a strategy to converge quickly to an optimal one.

Rather than individual utility, Lai *et al*. (2008) developed a heuristic for attaining Pareto-optimal agreements. This approach is based on an alternating offers protocol. In this protocol, in each time period, the proposing agent sends a set of offers rather than a single one. Based on these offers, the receiving agent employs a heuristic to generate a set of counter offers. The heuristic works as follows. An offer that lies on the indifference curve that is closest to the best offer made by the opponent is chosen. This offer is then used as a seed for randomly choosing several other offers in the neighbourhood of the seed. It is shown how this method can lead to near Pareto-optimal agreements.

Hindriks and Tykhonov (2008) and Hindriks *et al*. (2009) considered an opponent that uses a concession-based strategy (such as one that can be generated using negotiation decision functions) and developed a Bayesian learning model for learning the opponent's preferences for multiple independent issues. Under some assumptions on the structure of utility functions, the approach is shown to generate Pareto-optimal agreements.

More recently, a novel approach that takes into consideration both deadlines and discount factors was proposed by Williams *et al*. (2011, 2012). In this work, two concession strategies were developed for negotiations with real-time deadlines. One of these uses a fast, least-squares regression approach to predict the future concession of an opponent and accordingly propose a counter offer for multi-issue negotiations. The other uses a slower Gaussian process regression technique and, in addition to prediction, also provides a measure of the confidence in that prediction. The strategies have the following key features:

1. they do not depend on knowledge about the opponent's preferences, and
2. they can be used for negotiating over discrete issues.

Furthermore, the strategies allow one-to-one and many-to-many negotiations, making them easily applicable to real-world negotiations. Evaluation of these strategies against a number of state-of-the-art benchmarks showed that they not only perform well in terms of individual agent utilities but also result in close to Pareto-optimal agreements.

A common feature of the work discussed so far is that it deals with negotiations involving multiple independent issues. In contrast to this, negotiation over interdependent issues was dealt with by Ehtamo *et al*. (1999), Klein *et al*. (2003), Robu *et al*. (2005), Ito *et al*. (2007).

In their model, Ehtamo *et al*. (1999) used the following protocol:

Step 1 The mediator sends a constraint to the two negotiating parties. This constraint is formed by the mediator in such a way that the offers generated in Step 2 are close to Pareto optimality.

Step 2 Each party solves an optimisation problem to generate an offer that satisfies the constraint specified in Step 1 and is also optimal for her.

Step 3 The offers generated in Step 2 are sent to the mediator.

Step 4 If the mediator receives offers that are close to each other, it means a Pareto-optimal solution has been found. Otherwise, the mediator uses a heuristic based on Newtonian iteration and generates a revised constraint, and goes back to Step 1.

Figure 9.2 illustrates the working of this model. The authors showed that, under certain assumptions on the players' utility functions, this method converges to a Pareto-optimal agreement.

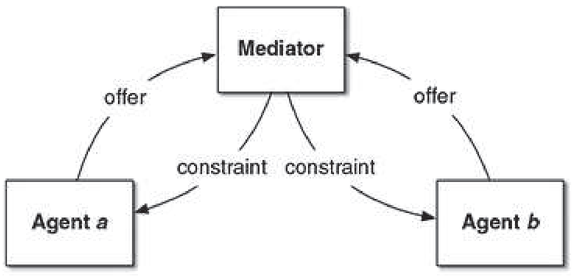

Figure 9.2 Information exchange between the mediator and the agents.

Robu *et al*. (2005) also presented a strategy for negotiations over interdependent issues. They developed heuristics that use prior information about the underlying *graphical structure* of utility functions to learn the opponent's preferences and generate counter offers. The heuristics were designed for an alternating offers protocol. There is no mediator in this approach. Experimental evaluations show that, within a few time periods, the proposed strategies lead to a Pareto-optimal agreement. Since only a few time periods are needed to reach an agreement, this approach is useful for negotiations with time constraints. This work was later extended by developing a method for automatic construction of utility graphs (Robu, 2009; Robu and La Poutre, 2009).

For non-mediated multi-issue negotiations in the context of nonlinear utility functions, Zheng *et al*. (2013) consider an imperfect-information setting. For an alternating offers protocol, they propose heuristic offer-generation strategies that converge to an agreement in finite time. For randomly generated problem instances, their simulation results demonstrate that the resulting agreements are close to the Nash bargaining solution.

Klein *et al*. (2003) developed a mediator-based model using *simulated annealing* as heuristic. Here, the aim is to generate social welfare-optimising agreements using the following protocol. The mediator proposes offers to the negotiating parties who then either accept or reject each offer. Each agent also annotates her accept or reject message as being *strong* or *weak*. Based on these messages, the mediator uses simulated annealing to decide upon a welfaremaximising solution.

A similar heuristic approach involving a mediator was later proposed by Ito *et al*. (2007). While the method of Klein *et al*. (2003) is based on the assumption that the dependencies between issues are binary, that of Ito *et al*. (2007) is more general in dealing with dependencies between more than two issues. In the latter, there are multiple negotiating parties and each uses simulated annealing to search a nonlinear utility space for optimal offers. These offers are then sent as proposals to the mediator, who then does an exhaustive search of these proposals to find a mutually consistent contract. It is shown how this method can lead to increased social welfare.

The simulated annealing approach with a mediator (Ito *et al*., 2007) was extended further in order to deal with highly nonlinear utility functions and overcome scalability issues (Marsa-Maestre *et al*., 2009, 2014). A key difference between the approach of Ito *et al*. (2007) and that of Marsa-Maestre *et al*. (2009) is that, for the former, an exhaustive search is used by the mediator while for the latter, a *probabilistic search* is used. It is shown how the probabilistic search can lead to better outcomes and also scale well with the number of issues and the number of parties.

While Klein *et al*. (2003), Ito *et al*. (2007) and Marsa-Maestre *et al*. (2009) employed a mediator to deal with the complexities of nonlinear utilities, Robu *et al*. (2005) introduced a heuristic method based on *utility graphs* (a utility graph is a graphical structure used to represent synergies between issues), and showed how it can be used to learn an opponent's preferences and reach Paretooptimal agreements through direct negotiation between the parties. Thus, a key advantage of this method is that it does not require a trusted mediator.

For those situations where an agent has several strategies available and must decide which one of these to play in a given negotiation, a learning-based approach was presented by Ilany and Gal (2012). They use a supervised machinelearning algorithm to predict the performance of the available strategies based on certain features of the negotiation and select a strategy that has the best predicted performance. Here, performance is measured in terms of individual utility; the higher the utility, the better the performance.

In some settings, such as the ANAC competition (see Baarslag *et al*. (2013a) for an overview of this competition), a negotiator must play against a range of opponents. Here, an agent must decide when to accept an offer. To this end, Baarslag *et al*. (2013b) studied generic strategies for acceptance conditions of offers in bilateral negotiations. They performed extensive experiments to compare the relative performance of these strategies in a broad range of negotiation scenarios. They also analysed correlations between the properties of a negotiation scenario and the efficacy of acceptance conditions.

Somefun and La Poutre (2007) consider the problem of a shop agent negotiating bilaterally with many customers about a bundle of goods or services together with a price. To facilitate the shop agent's search for mutually beneficial alternative bundles, they developed a method for learning customers' preferences. They deal with nonlinear preferences by converting them to a linear form.

9.3 Generating optimal agendas

Heuristics can not only be used to predict an opponent's preferences and generate counter offers, but also to find *optimal negotiation agendas*. The approaches described in the previous sections assume that the issues to be negotiated are given, and then show how to generate offers for those issues. However in many cases, it is not just the negotiation offers that determine the outcome of a negotiation: the actual issues included for negotiation also play a crucial part. In Chapter 7, we saw how the agenda can affect an agent's utility. Finding optimal agendas is therefore crucial to the outcome of those negotiations where an agenda can be chosen by an agent.

Recall from Chapter 7 that, depending on the number of available issues and the size of the agenda, the space of possible agendas can be huge. This means that an exhaustive search of this space to find an agent's optimal agenda may not be computationally feasible. The problem of finding an optimal agenda can be complicated further if the players' utility functions are nonlinear, in which case a player's trade-off problem for the PDP will be a nonlinear optimisation problem. Thus, finding an optimal agenda requires solving two key problems:

P_1 For a given agenda, find a player's equilibrium utility by solving a nonlinear optimisation problem.

P_2 Search the space of possible agendas to find one with the highest equilibrium utility.

We need heuristics to solve these problems. To this end, Fatima and Kattan (2011) used *genetic algorithms* (GAs) to evolve a player's optimal agenda for the PDP. The approach employs two GA search engines: one called an *inner search engine* to solve the problem P_1 and another called an *outer search engine* to solve P_2 . The inner search engine is a GA that takes an agenda as input and searches the space of possible solutions to the player's trade-off problem (i.e., a nonlinear optimisation problem) to find an optimal solution. The output of this search engine is the agreement that will result for the agenda and the player's utility from it.

The outer search engine explores the space of possible agendas to find an optimal one. Instead of a standard GA, this search engine uses a *surrogateassisted* GA. The reason for not using a standard GA is as follows. For a standard GA, the individuals that comprise the population would be the possible agendas. The fitness of each individual (i.e., an agenda) would have to be computed by running the inner search engine. However, running the inner GA for each individual that is part of the population of the outer GA will greatly slow down the evolutionary process. Thus, in order to speed up the process of evolution, a surrogate-assisted GA is used for the outer engine. A surrogate is a model used for finding the approximate fitness of an individual (i.e., agenda) in the population for the outer GA. Running the surrogate to find an approximate fitness (i.e., utility) for an agenda is computationally less expensive than running the inner GA (i.e., a standard GA).

The surrogate model, based on radial basis function networks (RBFNs) (Bishop, 2006) (an RBFN is a non-parametric regression model used for learning functions – it is a variant of artificial neural networks and uses radial basis functions (Buhmann, 2003) as activation functions), is created through a process of training as follows. Initially, a random set of agendas is chosen to train the surrogate. Then, each agenda in the training set is run through the inner search engine to find an equilibrium solution (i.e., the fitness of an agenda). Next, the agendas that comprise the training set and their corresponding equilibrium solutions are used to train the surrogate. The training process is demonstrated in Figure 9.3.

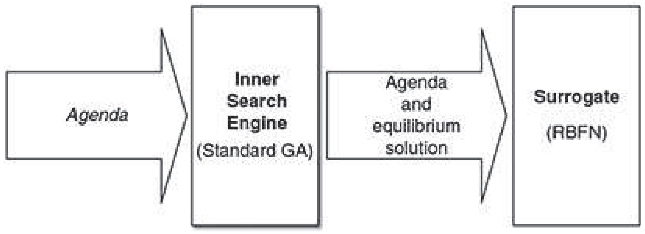

Figure 9.3 Training the surrogate.

Once the surrogate is trained, it becomes ready to take an agenda (that is not an element of the training set) as input, and compute an approximate equilibrium solution (i.e., an approximate fitness) for it. This surrogate is then used by the outer search engine to speed up the evolutionary process. This happens as follows. Instead of running the inner GA to find an equilibrium solution for every agenda that is part of the population of the outer GA, the surrogate is run for every agenda. Then, those agendas that are found to be promising (i.e., have a high approximate equilibrium utility) are run through the inner GA. In other words, the surrogate is used to direct the search towards those agendas that are likely to yield a high utility and therefore might be optimal. This process is demonstrated in Figure 9.4. For a range of utility functions, it is shown that this approach evolves better agendas than those evolved using a standard GA.

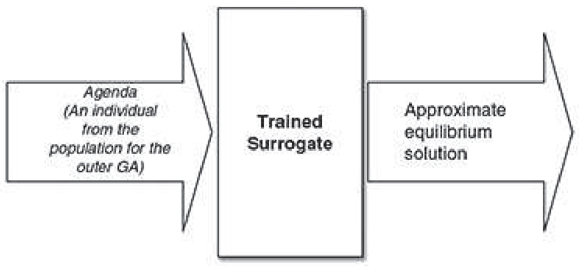
Agenda (An individual from the population for the outer GA)
Trained Surrogate
Approximate equilibrium solution

Figure 9.4 Using the surrogate with the outer GA.

More recent work by Kattan and Fatima (2012) uses a similar approach (i.e., genetic algorithms) to find optimal agendas for sequential negotiations with independent implementation. In Chapter 7, we saw how to find optimal agendas for negotiations with linear utility functions. When a utility function is nonlinear, finding an optimal agenda may not be easy. To solve this problem, Kattan and Fatima (2012) provide an evolutionary model comprising two GA systems: one for evolving an optimal set of issues and the other for evolving an optimal ordering for that set. Here, running two GA systems can be expensive in terms of execution time and computational resources. In order to overcome this problem, swarm optimisation (Clerc and Kennedy, 2002b) is used to tune the parameters (such as population size and number of generations) of the two GAs. For a range of utility functions, it is shown that this approach evolves better agendas than those evolved using a standard GA or a 1+1 evolutionary strategy (Clerc and Kennedy, 2002a).

9.4 Design and evaluation of heuristic strategies

From the discussion so far it is evident that there are a number of approaches for designing heuristic strategies. Choosing one of these is not easy, since there is no single approach that will work well for all negotiations. One must take the following factors into consideration when designing a strategy:

1. What is the objective of the strategy? Is it to maximise individual gain, to reach a Pareto-optimal agreement, and/or to maximise social welfare?
2. What is the negotiation protocol?
3. Are there constraints on time and/or resources?

4. What information is available on the opponent? Is she under time pressure to reach an agreement? If so, what is her deadline and what is her discount factor?
5. Is there an opponent model? If so, is it possible to learn (i.e., is the interaction going to be long enough for learning to be meaningful) new information about the opponent and use it to update the model?

The design of a strategy depends on the answers to these questions. Rahwan *et al*. (2007) provide a methodology for guiding the design of heuristic strategies. Their methodology is comprised of a number of stages described in Figure 9.5. Strategies may be developed by iterative refinement across stages. In Stage 1, an *objectives model* is specified. The objective may, for example, be “maximise individual utility and never concede on quality”, or “maximise a social welfare function but never concede on price”, and so on. In Stage 2, a *capabilities model* is distilled. Capabilities are specified in terms of the rules of a game, that is, a protocol. Stage 3 involves the construction of an *environment model*. The environment model is a representation of information about the parameters of negotiation and information about the opponent. With the objectives identified, capabilities clarified, and environment understood, the next stage is to *construct strategies*. Strategies are constructed by decomposing objectives into simpler sub-objectives using a top-down approach and combining the strategies for sub-objectives using a bottom-up approach. The fifth and last stage involves *testing* the strategies to ensure they meet the objectives laid out in the first stage. The strategies can then be improved through an iterative process of refinement.

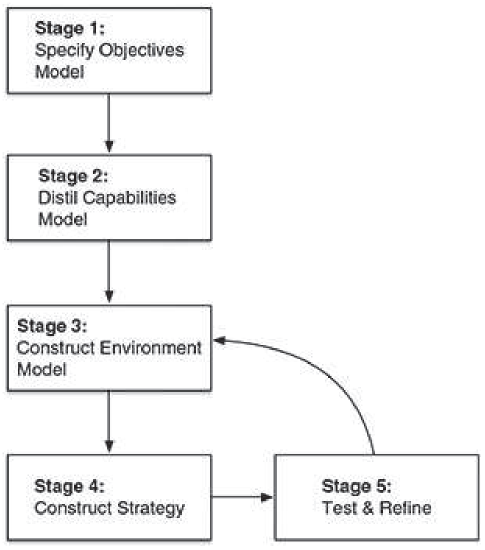

Figure 9.5 Designing a heuristic strategy.

Once a strategy is designed, it must be tested. The testing of a heuristic strategy is not straightforward. Testing is not easy because the performance of a strategy depends on the negotiation environment it is tested in. A strategy that performs well in one specific environment may be of little use. One would expect a heuristic strategy to perform well in a range of environments. Thus, strategies that adapt well to changing environments are desirable. But the number of possible environments may be too large to allow a thorough and comprehensive testing across all possible environments. In such cases, testing is limited to a reduced set of possible environments, and the performance of a strategy is evaluated by means of a comparison with some benchmark strategies, or else by playing them against opponents in competitions such as TAC (Collins *et al*., 2005) and ANAC (details given below).

In order to facilitate strategy design and testing, R. Lin *et al*. (2012) developed a software tool called GENIUS (*G*eneral *E*nvironment for *N*egotiation with *I*ntelligent multi-purpose *U*sage *S*imulation) for assisting in design and testing of heuristic negotiating agents. GENIUS is a best-practice repository for negotiation techniques. It provides repositories of techniques and negotiation domains, and contains information about which techniques are best suited to a given type of problem or domain. It also provides an easy-to-use environment for implementing new strategies.

Before closing this section, we provide a brief description of the Automated Negotiating Agents Competition (ANAC). The competition is an annual event, and has been running since 2010. It provides a unique benchmark for testing and evaluating bilateral negotiation strategies in multi-issue domains. The issues are discrete and independent, and the utility functions are linear and additive. Negotiation is time constrained by a deadline and a discount factor. The agents have incomplete information about each other.

In ANAC 2013, there were 18 participant agents. The participant strategies were varied; but most of them used strategies that adapted (e.g., by making a concession on a given round if the opponent conceded in the previous round) well to their opponent's strategy, although few did not adapt. Negotiations were conducted using an alternating offers protocol. The participants were played against each other in pairs in a wide range of negotiation scenarios. A scenario is defined in terms of the number of issues, the number of possible values the issues can take, the negotiation deadline, and the discount factor. For each negotiation, the participants are given a score. An agent's score is her utility from agreement. The agent that gets the highest average score across all the negotiations is the winner. It is interesting to note that, although there were adaptive strategies, the winning strategy was non-adaptive in that it did not make concessions in response to the opponent's concessions.

In order to understand the participant strategies and to determine what strategies constituted a Nash equilibria, a post-competition analysis was done. Since it is infeasible to compute Nash equilibria over the entire range of possible strategies and scenarios, a reduced strategy space was considered using empirical game-theoretic methods. See Baarslag *et al*. (2013a) for interesting details of this analysis.

So far in this book, our focus has been on the design of artificial agents that can negotiate effectively with each other without the intervention of humans. In the next chapter, we shift focus to the design of agents that can negotiate well with humans.

9.5 Historical notes and further reading

The ANAC has been running since 2010. Over the years, the rules of the competition have undergone minor changes. In 2013, for example, unlike previous years, the participants were allowed to download and save data from past negotiations in order to learn and improve performance. A similar competition is the TAC (Collins *et al*., 2005; Trading Agent Competition).

GENIUS is a Java-based negotiation platform that was developed to facilitate research in the area of bilateral multi-issue negotiations.[1] GENIUS facilitates the rapid development and integration of existing negotiating agents, and can be used to simulate individual negotiation sessions as well as tournaments between negotiating agents in various negotiation scenarios. The core functionality of the system includes:

1. specification of negotiation domains and preference profiles,
2. simulation of a bilateral negotiation between agents, and
3. analysis of the negotiation outcomes and negotiation dynamics.

It also allows for specification of negotiation domains and preference profiles by means of a graphical user interface. GENIUS has been used successfully for several years at graduate level in different universities to teach students how to create and evaluate negotiating agents.

Another negotiation platform is a configurable system called Coloured Trails.[2] It is a game played by two or more participants on a board of coloured squares. CT is an abstract, conceptually simple but highly versatile game in which players negotiate and exchange resources to enable them to achieve their individual or group goals. It provides a realistic analogue to multi-agent task domains, while not requiring extensive domain modelling.

[1] See `http://mmi.tudelft.nl/negotiation/index.php/Genius`.

[2] See `http://www.eecs.harvard.edu/ai/ct`.