AI Magazine Volume 23 Number 3 (2002) (© AAAI)

Strategic Design of Mobile Agents

Nir Vulkan

■ Much of the economic value of electronic commerce comes from the automation of interactions between businesses and individuals. Game theory is a useful set of tools that can be used by designers of electronic-commerce applications in analyzing and engineering of automated agents and communication protocols. The central theoretical concept used in game theory is the Nash equilibrium. In this article, I show how the outcomes supported by a Nash equilibrium can positively be enlarged using automated negotiations.

The last decade witnessed a phenomenal growth in the number of individuals and firms connected by the internet, which has already fundamentally changed the way many individuals and organizations think about and perform their work. *Electronic commerce*—the conduct of business activities electronically by digital media—is now part of everyday business. In addition, despite the sharp falls in the share prices of many "dot-coms" since early 2000, electronic commerce is still likely to have a major and lasting effect on most forms of economic activities.

Advances in web-based technologies further support the growth of electronic commerce. In particular, automation and delegation technologies—known variously as *intelligent,* or *smart, agents*—are likely to have a considerable effect on the future of electronic commerce. Automated agents are increasingly being used for scheduling tasks, resource allocation, automated negotiations, and online auctions. This trend is supported by growing standardization of communication infrastructures over which different organizations can interact and safely carry out transactions.

Much of the economic value of electronic commerce arises from this kind of automation. Automated electronic commerce creates new economic value not only by making business processes easier but also by opening up new possibilities for market interactions.

Automated agents that carry out one-to-one negotiations are of particular importance. One-to-one negotiations lie at the heart of many electronic-commerce applications. Designing efficient protocols that facilitate such negotiations can thus have a major impact on the development of electronic commerce. In particular, such protocols can be used for automated negotiations, opening the floodgate for a huge number of applications that benefit consumers and providers of services, such as electricity and telecoms.

Automated negotiations can be used in many electronic-commerce applications. For example, software programs can be used to obtain cheaper prices for utilities such as basic telephone services. A simple program can be installed to monitor and direct long distance calls. The user dials the country code, and as he/she continues to dial the telephone number, the program contacts various long distance providers and negotiates the best deal for its user. The program can be set up to inform the user about the price before the call is connected; for example, the best rate for this call is nine cents a minute with no minimum charge. If you are happy with this price and would like to continue with your call, please press 1.

Similarly, programs can be installed to monitor the consumption of household electricity. Because the software knows the pattern of electricity demand in the household, it is able to negotiate meaningfully with the various electricity providers at regular intervals. A number of experiments with this type of technology have been taking place in Scandinavia and the United States.

Automated negotiation technology can also be used to reduce negotiation overheads within large organizations. The interactions of the

various departments within large organizations are often best described in terms of self-interest. Of course, they all work for the same organizations, but they might be pursuing conflicting goals. The legal department might have different requirements than the sales department, and the technical people might impose their own constraints. Software can be used to replace much of the small-print repeated-type negotiation within the organization. These negotiations are dominantly one to one (the legal department must agree with the technical department; for example, there are no outside options here). The advance decision environment for process task (ADEPT) system is a good example of such an application (see Vulkan and Jennings [2000] and the references therein for more details).

Along the same lines, software can be used to facilitate negotiations between individuals and organizations. The KASBAH system, which was tested at the media lab at the Massachusetts Institute of Technology in October 1996, is a good example. Around 200 people (largely from technological industries) were given books, souvenirs, and play money to participate in a full-day marketplace where trading took place with automated agents. Although users retained some control over the negotiation strategies used by their agents, the actual negotiations were fully automated (see Chavez and Maes [1996] for further details). More generally, programs that know the preferences of their users and that are equipped with some degree of negotiation skills, can buy, sell, or schedule meetings on behalf of their users.

Finally, the internet itself is being used as a huge marketplace for computations. The Popcorn Project at the Computer Science Department of the Hebrew University in Jerusalem provides an infrastructure for globally distributed computation over the whole internet.[1] It provides any programmer connected to the internet with a single huge virtual parallel computer composed of all processors on the internet that care to participate at any given moment. Negotiations can take place either manually or automatically (that is, between software programs), so that owners of underused computers can actually make money by subletting their central processing units to those who need them.

Game Theory and Automated Negotiations

Because the outcomes of interactions between self-interested agents depend on the behavior of all agents, the situation can be described as a game and the agents themselves as players. There are two significant points about game theory as a way of analyzing interactions between self-interested agents:

First, it provides a uniform language with which to describe the interactions of self-interest agents.

Second, it defines the "solutions" or "equilibria" of games.

In analyzing games, the actions and decisions available to players are referred to as their strategies. For example, a bidder in a sealed-bid auction submits a single bid that must be a nonnegative number; so, the set of positive real numbers are his/her possible strategies. Once all players have selected their strategies, the game is played, and an outcome is realized.

The main solution concept used in game theory is the Nash equilibrium. A *Nash equilibrium* is a combination of strategies, one for each player, which has the property that given the choice of strategies of the other players, none of the players wants unilaterally to change his/her strategy. In other words, each of the players chooses his/her optimal strategy given the choices of the other players, which is essentially a notion of stability. It is not always optimal, in the sense that there could be other combinations of strategies, which leads to a higher outcome to some or even all players. The prisoner's dilemma is a good example.

The notion of equilibrium is central to game theory. The rationale for using it in economic situations can roughly be described as follows: Because economic actors are assumed to be rational, they will continue to change their behavior as long as it is beneficial. Only in equilibrium can no one have an incentive to change; so, this equilibrium is likely to resemble actual behavior, at least in the long run.

The argument for looking at Nash equilibria is even more appealing in automated negotiations between software agents: Technically speaking, an agent is a preprogrammed algorithm acting on real-time data. As such, it corresponds exactly to the game-theoretic notion of what a strategy is. More precisely, a strategy in game theory is a mapping from all possible histories and information sets to actions. Once a player picks a strategy, he/she knows what to do at any given set of circumstances.

This suggests that game-theory strategies might not describe very well how people behave in many cases. Casual observation makes it clear that people tend to make decisions on the go or as events unfold (we'll cross this bridge when we come to it). This type of behavior can be suboptimal in many cases, but nevertheless it is what people do. Designers of

automated electronic commerce, in contrast, instruct their programs in advance how to act to whatever happens. A software agent (or any real-time algorithm for that matter) is a "time-consistent" description of decision making. Hence, game theory is probably more suitable for the study of automated interactions than it is for the study of the interactions between humans (a point recognized by Rosenschein and Zlotkin [1994], who pioneered the use of game theory for automated interactions).

What does this mean for actual behavior in automated negotiations? Once the equilibrium is identified, there is no need in principle for agents to actually carry out the negotiations process! Optimal strategies can be used directly by agents, thus bypassing the actual trial-and-error process of getting to equilibrium. As long as the outcome constitutes an equilibrium of the underlying game (as specified by the rules or the communication protocol), then this will be optimal for all agents. However, if the game has more than one equilibrium, then agents will need to somehow coordinate their actions to a single equilibrium. Once again, this is easier to do in an automated system because the protocol itself can choose the equilibrium to be played: Because it is an equilibrium, it will be in the best interest of agents to follow the protocol's recommendation.

Still, software agents differ from their human counterparts in several important respects, and the current challenge for researchers in this field is to find out which of the insights offered by game theory can be useful for the design of automated agents and the environments where they interact. To do this, existing models have to be adapted to the specifics of such problems.

There is already some literature on extending game-theoretic models to deal with specific requirements arising from electronic-commerce applications (see, for example, Vulkan [1999] for a general discussion). New technologies are emerging based on game-theoretic analysis of such automated interactions. One such example is the one-to-one negotiations between automated agents facing strict deadlines (which are private information), by which they must reach an agreement. The large literature on bargaining does not offer much insight into this problem. This problem was addressed by Sandholm and Vulkan (1999), who found a simple and efficient mechanism that resolves these negotiations in such a way that it becomes optimal for self-interested agents to truthfully report their deadline. In a somewhat different context, Sandholm and Lesser (1996) devised a technology known as *level-commitment contracts*. This technology allows agents to specify penalties for unilateral "decommitting" from agreements, expanding the set of possible agreements and, hence, increasing the efficiency of such negotiations.

Even in these situations already studied by economists, automated negotiations can lead to different outcomes. One reason is the fact that agents, and especially *mobile agents* (that is, agents that migrate between hosts), will need to be checked by the host to ensure that the agent's code is harmless (that is, it is not a virus) and is compatible with the communication protocol. For example, an agent bidding in an English auction can safely reveal its code to the host, encrypting only its user's reservation price:

```
Begin strategy:
If ((current bid) < (encrypted reservation price)
   and (auction continues))
then (bid = current bid + constant)
End strategy
```

If agents are designed to always interact under the same conditions and always on the same host, then no such checks are required. In fact, in many existing agent-based applications, users receive their agents directly from the host (normally in the form of a JAVA script) and key in the relevant information. However, the current trend is toward applications that allow for participation of any agents compatible with the communication protocol. In this framework, users choose or design agents to interact on their behalf on a number of potential hosts and in varying circumstances. In other words, users will not know in advance exactly when and where agents are matched. A computerized agent in such a setting is likely to use a case base memory that will allow them to pursue (possibly) different strategies in different cases. Specifically, agents can credibly demonstrate to one another how their behavior depends (or not) on the identity of the host:

```
Begin strategy:
If (host 1) then (encrypted)
else if (host 2) then (encrypted)
…
else if (host n) then (encrypted)
End strategy
```

or

```
Begin strategy:
If (hosts 1, 2, .., n – 1) then (encrypted)
else if (host n) then (encrypted)
End strategy
```

and so on.

In the next two sections, which are based on Vulkan (2001), I consider the implications on

	Coop	Greedy
Coop	4, 4	0, 2
Greedy	2, 0	–2, –2

Table 1. ω1: Penalizing Host.

	Coop	Greedy
Coop	4, 4	0, 5
Greedy	2, 5	1.5, 1.5

Table 2. ω_2: Nonpenalizing Host.

the set of possible equilibrium contracts that agents might be able to credibly reveal parts of their behavioral strategy to each other. I show that the set of outcomes supported by these type of interactions is typically larger than when codes cannot be revealed. Moreover, I show that if agents can **choose** whether to reveal their code, then this will occur in equilibrium only if the outcome is welfare improving compared to the equilibrium outcomes of the game where code revelation is not possible. I provide a simple two-agent example that demonstrates how agents can coordinate on actions that otherwise would have strictly been dominated.

To demonstrate this point, consider the following stylized example: There are several providers of bandwidth. Agents representing bandwidth consumers (for example, internet service providers) are randomly matched by the market-clearing protocol used by the double-auction server (as in Band-X or RateXchange) to a provider. I reduce their choices of consumption pattern to only two strategies: (1) greedy and (2) cooperative. Unless hosts have specific mechanisms in place to induce cooperation, agents are locked in a version of the prisoners' dilemma, where the greedy consumption pattern is dominant. That is, the agents would be better off if they all cooperate, but self-interest prevents them from doing so: Given that the others cooperate, the agent is better off being greedy. However, some hosts penalize greedy bandwidth-consuming agents to resolve this dilemma. Sufficiently high penalties are imposed on agents that are greedy, so that if the others cooperate, the agent is also better off cooperating (in game theory terminology, penalties are sufficiently high to ensure that the cooperative strategy is its own best response).

If codes cannot be revealed, agents will always be greedy when matched on a host that does not reward cooperation. However, if codes are revealed, and if on average, cooperation with cooperators is beneficial, then agents can in equilibrium choose not to distinguish between hosts and to always cooperate with each other. The intuition for this result is that the choice of host-based strategy is used as a coordination device by signaling the willingness to play a particular continuation equilibrium.

A Detailed Example

Tables 1 and 2 show the payoffs for the two agents, as a function of the state (the host), ω:

That is, in ω_1, if both agents cooperate, then they each receive a payoff of 4. If one of the agents cooperates and the other is being greedy, then the first agent receives a payoff of 0, and the greedy agent receives a payoff of 2. The numeric value is, of course, not important—what is important is that each agent prefers a payoff of 4 to a payoff of, say, 2. Still, by picking specific values, it becomes possible to illustrate the strategic value of code revelation.

Let *g* denote the greedy strategy and *c* the cooperative strategy. In the game specified by ω_2, the strategy *g* strictly dominates *c* (that is, an agent is better off being greedy whatever the strategy used by its opponent). The game has, therefore, a unique Nash equilibrium, where both players play *g*. In ω_1, the situation is reversed, and (*c, c*) is the only Nash equilibrium. We assume that agents are assigned to ω_1-type hosts with probability 3/8 and to ω_2-type hosts with probability 5/8. We now consider a metagame that consists of the following four stages:

Stage 1: Users choose agents.

Stage 2: Agents are assigned to a host, according to the previous probabilities.

Stage 3: Each agent observes the nonencrypted part of its opponent strategy.

Stage 4: Agents simultaneously choose bandwidth-consumption patterns.

This four-stage metagame allows us to simultaneously consider the choices of agents and strategies. Thus, users have a richer set of metagame strategies to choose from. For example, a user can choose an agent that distinguishes between the states ω_1 and ω_2 and that, if matched with a state-distinguishing agent, plays *g* in ω_1 and *c* in ω_2 and that plays *c* in ω_1

and g in ω_2 if matched with a nondistinguishing agent. By analyzing the equilibria of this four-stage game, we gain insight into what is an optimizing agent in a multihost setting.

It turns out (Vulkan 2001) that the metagame has two equilibria: First, there is an equilibrium where agents optimize in each state. Specifically, agents do distinguish between states and (both) play c in ω_1 and g in ω_2. Second, and far less obvious, there exists an additional equilibrium to the metagame where players choose agents that cannot distinguish between states and cooperate with each other on either type of host. In other words, even though agents can, at no costs, distinguish between states, they (more specifically, their designers) choose not to, so that they can always cooperate with each other. However, the mechanics of this equilibrium are such that it crucially depends on agents' ability to credibly reveal the case base part of their code (more formally, agents' behavior in equilibrium depends on their behavior off equilibrium, that is, what they were to do if their opponent deviated from its equilibrium strategy). If we remove this ability, agents will have incentives to only pretend (that is, to claim) to not be able to distinguish between states but actually become informed about the state and then play g in ω_2.

The first equilibrium is essentially a combination of the two Nash equilibria of each of the two games specified by the hosts, ω_1 and ω_2. The second equilibrium, however, does not correspond to the equilibria of the underlying games; in fact, the agents play a strategy (c in ω_1) that is clearly not rational (recall that the strategy c is dominated by the strategy g in ω_1). The following section generalizes the relationship between the set equilibria of metagames, that is, where users pick agents and strategies, and the equilibria of the underlying games.

We can already, however, use the previous example to give the intuition behind the main result. Suppose that users can choose whether their agents reveal their nonencrypted code. That is, the game has an additional stage, 2.5, where agents choose whether to reveal their type. If users pick agents that do not reveal their code, then it will not be possible, in equilibrium, to always cooperate: Because agents cannot signal that they cannot distinguish between states, they will distinguish (and, consequently, play c in ω_1 and g in ω_2). If, however, users do pick agents that reveal their code, then this in itself is a credible signal and a coordination device to always cooperate (because they are both better off cooperating). In the following section, I generalize this intuition.

The Model

Let ω_i denote the game specified by the communication protocol used by host i. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ denote the set of games (states). Let A_j be the set of actions for agent $j = 1, 2$. We assume that these sets are constant over all states in Ω (that is, all games have the same set of strategies). Nature chooses an element in Ω according to a probability measure ρ, where ρ is common knowledge. Players choose, in the first stage of the game, a partition of Ω. Formally, $P = \{P_1, P_2, \ldots, P_k\}$ is a partition of Ω if (1) $P_i \neq \varnothing$ (all i), (2) $P_i \cap P_j = \varnothing$ (all $i \neq j$), and (3) for all $\omega \in \Omega$ there exist P_i with $\omega \in P_i$. Let $\boldsymbol{P}$ denote the set of all possible partitions of Ω. P^i denotes player's i partition. Once nature chooses the state, both players receive a (perfect) signal. Player i belief is determined by ρ conditional on P^i.

Formally we define the following four-, or five-, stage game, Γ

Stage 1: Users choose agents.

Stage 2: Nature chooses ω (agents are assigned to a host), according to ρ.

(**Stage 2.5:** Agents choose whether to make their partition public.)

Stage 3: Agents observe public partitions.

Stage 4: Agents choose actions, and payoffs are distributed.

An agent in this context is identical to the game-theoretic notion of strategy in the stage game. If we do not include stage 2.5, then an agent for player i is a pair $s_i = (P^i, z_i)$, where z_i: $P^i \times \boldsymbol{P} \rightarrow A_i$. If stage 2.5 is included, then an agent is also a pair $s_i = (P^i, z_i)$ but with the difference that now z_i: $P^i \times \{Reveal, Not\text{-}reveal\} \times \{\boldsymbol{P} \cup (P^{-i}\ Not\text{-}reveal)\} \rightarrow A_i$.

Expected payoffs are defined in the following way: Fix the partitions of both players, P^1 and P^2. Denote by

$$u_i^{\omega_i}(a_1, a_2) \quad (1)$$

player i's payoff when the pair of actions (a_1, a_2) is chosen in state ω_k. Expected payoff in Γ for player i is given by

$$\pi_i(s_1, s_2) \equiv \sum_{k=1}^{n} u_i^{\omega_i}\left(z_1\left(P^1(\omega_k)\right), z_2\left(P^2\right)(\omega_k)\right) \quad (2)$$

for $I = 1, 2$.

We are now able to prove the following results for the stage game G, which includes stage 2.5:

Proposition 1

For any sequential equilibrium s of Γ, where both players choose not to reveal their code, it must follow that $s(\omega_i)$ is a Nash equilibrium of the game ω_i for $I = 1, \ldots, n$ (Vulkan 2001).

Proposition 1 formalizes the intuition that if users pick agents that do not reveal their codes, then optimally designed agents will behave optimally at each state. In other words, on every site, agents will play the Nash equilibrium of the game specified by the rules of this site. If, however, agents do reveal (part of) their codes, then as we saw in the previous section, other equilibria are also possible. To characterize these additional equilibria, we introduce the following notations.

Let S^k be the set of equilibria of the game ω_k, $k = 1, \ldots n$. Let $\boldsymbol{S}$ be the set of all combinations of these equilibria over all states in Ω (that is, $|\boldsymbol{S}|=|S^1| \times |S^2| \times \ldots \times |S^n|$). Proposition 1 states that a sequential equilibrium of Γ, where players choose agents that do not reveal their code, must correspond to an element of $\boldsymbol{S}$. We now characterize a sufficient condition for a sequential equilibrium of Γ, which does not correspond to any element of $\boldsymbol{S}$.

Proposition 2

> Let s be a Nash equilibrium of the game with fixed partitions, P^1 and P^2. Then if $\pi_1(s) \geq \pi_1(s')$ for all $s' \in S$, then there exists a sequential equilibrium of Γ where, on the equilibrium path, players choose agents with partition P^1 and P^2, which then reveal their code, and continue with s (Vulkan 2001).

The important condition is $\pi_1(s) \geq \pi_1(s')$ for all $s' \in S$; that is, both agents must be better off in s compared to any equilibrium in $\boldsymbol{S}$. In economic terminology, we say that s is a *Pareto improvement* over all the equilibria in $\boldsymbol{S}$ (that is, the equilibria that consist of the Nash equilibria of the underlying games). Combining propositions 1 and 2, we obtain an interesting positive result: By revealing their codes, agents can never become worst off (proposition 1), but can, and in equilibrium will, become better off compared to a situation where codes cannot be revealed.

Conclusions

Security is one of the most important issues in the design of agent-based electronic-commerce systems. Because agents carry information about their users preferences and willingness to pay, there are potentially substantial gains from reversely engineering these agents. For this reason, anonymous and synchronizing technologies are being developed to ensure secure transactions between automated agents. In this article, I showed that parts of the agents' strategies that can credibly be revealed can also be used by agents to increase the utility of their users. Specifically, agents can, by credibly revealing how their behavior is conditioned on features such as the identity of the host, increase the set of equilibrium outcomes. The choice to reveal these behavioral strategies can be interpreted as a signal to continue with a welfare-improving continuation equilibrium.

Automated interactions between software agents provide a new framework to investigate the intuitions of game theory. In particular, it is interesting to compare the equilibrium outcomes of automated interactions with those of humans. One important difference is that an agent can credibly demonstrate that it is, in some sense, ignorant. For example, in repeated interactions, Moderer and Tenennholtz (1999) study the equilibrium behavior of software agents capable of demonstrating that they cannot remember previous rounds (that is, cannot condition on outcomes of previous rounds). Moreover, these types of code revelations are likely to be supported because hosts will need to check compatibility with the communication protocol. This article shows that this, in fact, might have a positive effect on overall payoffs to all participants.

Note

1. See www.cs.huji.ac.il/~popcorn/index.html for details.

References

Chavez A., and Maes, P. 1996. KASBAH: An Agent Marketplace for Buying and Selling Goods. In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multiagent Systems, April, London, United Kingdom.

Monderer D., and Tenennholtz, M. 1999. Distributed Games. *Games and Economic Behavior* 28:55–72.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter.* Cambridge, Mass.: MIT Press.

Sandholm, T. W., and Lesser, V. R. 1996. Advantages of a Level-Commitment Contracting Protocol. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 126–133. Menlo Park, Calif.: American Association for Artificial Intelligence.

Sandholm T. W., and Vulkan, N. 1999. Bargaining with Deadlines. In Proceedings of the Sixteenth National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Vulkan N. 2001. Equilibria in Automated Interactions. *Games and Economic Behavior* 1–2:339–348.

Vulkan N. 1999. Economic Implications of Agent Technology and E-Commerce. *The Economic Journal,* 109(453): F67-F90.

Vulkan, N., and Jennings, N. R. 2000. Efficient Mechanisms for the Supply of Services in Multi-Agent Environments. *Decision Support Systems* 28:5–19.

Nir Vulkan is an assistant professor in economics at the Said Business School, Oxford University. He is also the management fellow in Worcester College. He specializes in applying ideas from game theory to electronic commerce. Vulkan has extensive commercial experience and is involved with a number of electronic-commerce startup companies. He has developed several electronic-commerce technologies that are used in B2B and B2C electronic markets and holds two patents in this area. Vulkan also has a long-term consulting relationship with the Electronic-Services Division at Hewlett Packard. He holds a BSC in mathematics and computer science from Tel-Aviv University and a Ph.D. in game theory from University College London (UCL). His e-mail address is nir.vulkan@sbs.ox.ac.uk.