5 Distributed Rational Decision Making

Tuomas W. Sandholm

5.1 Introduction

Automated negotiation systems with self-interested agents are becoming increasingly important. One reason for this is the *technology push* of a growing standardized communication infrastructure—Internet, WWW, NII, EDI, KQML, FIPA, Concordia, Voyager, Odyssey, Telescript, Java, *etc.*—over which separately designed agents belonging to different organizations can interact in an open environment in real-time and safely carry out transactions. The second reason is strong *application pull* for computer support for negotiation at the operative decision making level. For example, we are witnessing the advent of small transaction electronic commerce on the Internet for purchasing goods, information, and communication bandwidth [31]. There is also an industrial trend toward virtual enterprises: dynamic alliances of small, agile enterprises which together can take advantage of economies of scale when available (e.g., respond to more diverse orders than individual agents can), but do not suffer from diseconomies of scale.

Multiagent technology facilitates such negotiation at the operative decision making level. This automation can save labor time of human negotiators, but in addition, other savings are possible because computational agents can be more effective at finding beneficial short-term contracts than humans are in strategically and combinatorially complex settings.

This chapter discusses multiagent negotiation in situations where agents may have different goals, and each agent is trying to maximize its own good without concern for the global good. Such self-interest naturally prevails in negotiations among independent businesses or individuals. In building computer support for negotiation in such settings, the issue of self-interest has to be dealt with. In *cooperative distributed problem solving* [12, 9], the system designer imposes an interaction *protocol*[1] and a *strategy* (a mapping from state history to action; a

1. Here a protocol does not mean a low level communication protocol, but a negotiation protocol which determines the possible actions that agents can take at different points of the interaction. The *sealed-bid first-price auction* is an example protocol where each bidder is free to submit one bid for the item, which is awarded to the highest bidder at the price of his bid.

way to use the protocol) for each agent. The main question is what social outcomes follow given the protocol and *assuming that the agents use the imposed strategies.* On the other hand, in *multiagent systems* [67, 63, 61, 56, 34], the agents are provided with an interaction protocol, but each agent will choose its own strategy. A self-interested agent will choose the best strategy for itself, which cannot be explicitly imposed from outside. Therefore, the protocols need to be designed using a *noncooperative, strategic* perspective: the main question is what social outcomes follow given a protocol which *guarantees that each agent's desired local strategy is best for that agent—and thus the agent will use it.* This approach is required in designing robust non-manipulable multiagent systems where the agents may be constructed by separate designers and/or may represent different real world parties.

The rest of this chapter discusses protocols for voting, auctions, bargaining, markets, contracting, and coalition formation. However, first some central evaluation criteria for protocols are presented.

5.2 Evaluation Criteria

Negotiation protocols—i.e. mechanisms—can be evaluated according to many types of criteria, as listed below. The choice of protocol will then depend on what properties the protocol designer wants the overall system to have.

5.2.1 Social Welfare

Social welfare is the sum of all agents' payoffs or utilities in a given solution. It measures the global good of the agents. It can be used as a criterion for comparing alternative mechanisms by comparing the solutions that the mechanisms lead to. When measured in terms of utilities, the criterion is somewhat arbitrary, because it requires interagent utility comparisons, and really each agent's utility function can only be specified up to positive affine transformations [39].

5.2.2 Pareto Efficiency

Pareto efficiency is another solution evaluation criterion that takes a global perspective. Again, alternative mechanisms can be evaluated according to Pareto efficiency by comparing the solutions that the mechanisms lead to. A solution x is Pareto efficient—i.e. Pareto optimal—if there is no other solution x' such that at least one agent is better off in x' than in x and no agent is worse off in x' than in x. So, Pareto efficiency measures global good, and it does not require questionable interagent utility comparisons.

Social welfare maximizing solutions are a subset of Pareto efficient ones. Once the sum of the payoffs is maximized, an agent's payoff can increase only if another agent's payoff decreases.

5.2.3 Individual Rationality

Participation in a negotiation is individually rational to an agent if the agent's payoff in the negotiated solution is no less than the payoff that the agent would get by not participating in the negotiation. A mechanism is individually rational if participation is individually rational for all agents. Only individually rational mechanisms are viable: if the negotiated solution is not individually rational for some agent, that self-interested agent would not participate in that negotiation.

5.2.4 Stability

Among self-interested agents, mechanism should be designed to be stable (non-manipulable), i.e. they should motivate each agent to behave in the desired manner. This is because if a self-interested agent is better off behaving in some other manner than desired, it will do so.

Sometimes it is possible to design mechanisms with *dominant strategies*. This means that an agent is best off by using a specific strategy no matter what strategies the other agents use.

However, often an agent's best strategy depends on what strategies other agents choose. In such settings, dominant strategies do not exist, and other stability criteria are needed. The most basic one is the *Nash equilibrium* [48, 39, 17, 35]. The strategy profile $S_A^* = \langle S_1^*, S_2^*, ..., S_{|A|}^* \rangle$ among agents A is in Nash equilibrium if for each agent i, S_i^* is the agent's best strategy—i.e. best response—given that the other agents choose strategies $\langle S_1^*, S_2^*, ..., S_{i-1}^*, S_{i+1}^*, ..., S_{|A|}^* \rangle$. In other words, in Nash equilibrium, each agent chooses a strategy that is a best response to the other agents' strategies.

There are two main problems in applying Nash equilibrium. First, in some games no Nash equilibrium exists [39, 17, 35]. Second, some games have multiple Nash equilibria, and it is not obvious which one the agents should actually play [35].

There are also limitations regarding what the Nash equilibrium guarantees even when it exists and is unique.

First, in sequential games it only guarantees stability in the beginning of the game. At a later stage the strategies need not be in equilibrium anymore. A refined solution concept called the *subgame perfect Nash equilibrium* is defined to be a Nash equilibrium that remains a Nash equilibrium in every subgame (even subgames that are not along the actual path of play and will thus never be reached) [71, 39, 17, 35]. This solution concept also suffers from existence and uniqueness problems.

Second, the Nash equilibrium is often too weak because subgroups of agents can deviate in a coordinated manner. Some refinements of the Nash equilibrium solution concept guarantee stability against such collusive deviations as well. This will be discussed in Section 5.8.

Sometimes efficiency goals and stability goals conflict. A simple example of this is the Prisoner's Dilemma game where the unique welfare maximizing and Pareto efficient strategy profile is the one where both agents cooperate, Table 5.1. On the other hand, the only dominant strategy equilibrium and Nash equilibrium is the one where both agents defect.

		column player	
		cooperate	defect
row	cooperate	3, 3	0, 5
player	defect	5, 0	1, 1

Table 5.1 Prisoner's Dilemma game. The row player's payoff is listed first.

5.2.5 Computational Efficiency

Clearly, mechanisms should be designed so that when agents use them, as little computation is needed as possible. Classically, mechanisms have been designed so that they lead to domain solutions that satisfy some of the above evaluation criteria. Of these mechanisms, the ones with the lowest computational overhead have been preferred. However, a more advanced approach would be to explicitly trade off the cost of the process against the solution quality [62].

5.2.6 Distribution and Communication Efficiency

All else being equal, distributed protocols should be preferred in order to avoid a single point of failure and a performance bottleneck—among other reasons. Simultaneously one would like to minimize the amount of communication that is required to converge on a desirable global solution. In some cases these two goals conflict.

The rest of this chapter discusses different interaction protocols using the evaluation criteria presented so far. These mechanisms include voting, auctions, bargaining, markets, contracting, and coalition formation.

5.3 Voting

In a *voting* (social choice) setting, all agents give input to a mechanism, and the outcome that the mechanism chooses based on these inputs is a solution for all of the agents. In most settings, this outcome is enforced so that all agents have to abide to the solution prescribed by the mechanisms.

5.3.1 Truthful Voters

The classic goal has been to derive a social choice rule that ranks feasible social outcomes based on individuals' rankings of those outcomes. Let the set of agents be A, and let O be the set of feasible outcomes for the society. Furthermore, let each agent $i \in A$ have an asymmetric and transitive strict preference relation $\succ_i$ on O. A social choice rule takes as input the agents' preference relations $(\succ_1, ..., \succ_{|A|})$ and produces as output the social preferences denoted by a relation $\succ^*$. Intuitively, the following properties of a social choice rule seem desirable:

- A social preference ordering $\succ^*$ should exist for all possible inputs (individual preferences).
- $\succ^*$ should be defined for every pair $o, o' \in O$.
- $\succ^*$ should be asymmetric and transitive over O.
- The outcome should be Pareto efficient: if $\forall i \in A, o \succ_i o'$, then $o \succ^* o'$.
- The scheme should be independent of irrelevant alternatives. Specifically, if $\succ$ and $\succ'$ are arrays of consumer rankings that satisfy $o \succ_i o'$ iff $o \succ'_i o'$ for all i, then the social ranking of o and o' is the same in these two situations.
- No agent should be a dictator in the sense that $o \succ_i o'$ implies $o \succ^* o'$ for all preferences of the other agents.

Unfortunately it is not possible to satisfy these desiderata:

Theorem 5.1 Arrow's impossibility theorem
No social choice rule satisfies all of these six conditions [4, 35].

So, to design social choice rules, the desiderata have to be relaxed. Commonly the first property is relaxed in the sense that the domain (combinations of individual preferences) on which the rule works is restricted. This will be discussed later in conjunction with insincere voting.

The third desideratum can also be relaxed. This is done e.g. in the *plurality protocol* which is a majority voting protocol where all alternatives are compared simultaneously, and the one with the highest number of votes wins. Introducing an irrelevant alternative can split the majority; some in favor of the old most favored alternative, and some in favor of the newly introduced alternative. This may cause both the old favorite, and the newly introduced irrelevant alternative to drop below one of the originally less preferred alternatives, which then would become the social choice.

In a *binary protocol*, the alternatives are voted on pairwise, and the winner stays to challenge further alternatives while the loser is eliminated. As in plurality protocols, also in binary protocols the introduction of irrelevant alternatives often changes the outcome. Furthermore, in binary protocols, the *agenda*—i.e. order of the pairings—can totally change the socially chosen outcome. For example, Figure 5.1 shows four different agendas which all lead to a different outcome under the given preferences of the agents. Interestingly, in the last agenda, alternative

Figure 5.1 Four different agendas for a binary protocol with four alternatives: a, b, c, and d.

d wins although every agent prefers c over d. In other words, the social choice is Pareto dominated in this case.

If the number of alternative outcomes is large, pairwise voting may be slow, and an alternative called the *Borda protocol* is often used. The Borda count assigns an alternative $|O|$ points whenever it is highest in some agent's preference list, $|O| - 1$ whenever it is second and so on. These counts are then summed across voters. The alternative with the highest count becomes the social choice. The Borda protocol can also lead to paradoxical results, for example via irrelevant alternatives. Table 5.2 shows an example (from [49]) where removing the lowest ranked (irrelevant) alternative d from the set of possible outcomes leads to the worst of the remaining alternatives turning best and the best turning worst.

Agent	**Preferences**
1	$a \succ b \succ c \succ d$
2	$b \succ c \succ d \succ a$
3	$c \succ d \succ a \succ b$
4	$a \succ b \succ c \succ d$
5	$b \succ c \succ d \succ a$
6	$c \succ d \succ a \succ b$
7	$a \succ b \succ c \succ d$
Borda count	c wins with 20, b has 19, a has 18, d loses with 13
Borda count with d removed	a wins with 15, b has 14, c loses with 13

Table 5.2 Winner turns loser and loser turns winner paradox in the Borda protocol.

5.3.2 Strategic (Insincere) Voters

So far it was assumed that in executing the social choice method, all agents' preferences are known. In reality this is seldom the case. Instead, agents usually have to reveal, i.e. declare, their preferences. Assuming knowledge of the preferences is equivalent to assuming that the agents reveal their preferences truthfully. But if an agent can benefit from insincerely declaring his preferences, he will do so. This further complicates the design of social choice mechanisms.

An area of game theory called *mechanism design* explores such interaction mechanisms among rational agents. The goal is to generate protocols such that when agents use them according to some stability solution concept—e.g. dominant strategy equilibrium [45, 17, 35], Nash equilibrium or its refinements [40, 52, 17, 35], or some other type of equilibrium [46, 17, 35]—then desirable social outcomes follow. The strategies are not externally imposed on the agents, but instead each agent uses the strategy that is best for itself.

Let each agent $i \in A$ have some *type* $\theta_i \in \Theta_i$ which totally characterizes his preferences (which are affected by his possible private information). Now, a social choice function $f : \theta \rightarrow O$ chooses a social outcome given the agents' types. With insincere agents this is hard to implement because one needs to somehow motivate the agents to reveal the types. A protocol (i.e. rules of the game) is said to implement a particular social choice function if the protocol has an equilibrium—which may involve insincere play by the agents—whose outcome is the same as the outcome of the social choice function would be if the agents revealed their types truthfully. The following positive result conceptually allows one to restrict the search for desirable protocols to ones where revelation occurs truthfully in a single step.

Theorem 5.2 Revelation principle
Suppose some protocol (which may include multiple steps) implements social choice function $f(\cdot)$ in Nash (or dominant strategy) equilibrium (where the agents' strategies are not necessarily truthful). Then $f(\cdot)$ is implementable in Nash (or dominant strategy, respectively) equilibrium via a single-step protocol where the agents reveal their entire types truthfully [39, 35, 45, 40, 52].

The proof is based on changing the protocol so that it will construct the best (i.e. according to the original equilibrium) strategic revelation on behalf of each agent (this revelation depends on the agent's truthful preferences), and then simulate the old protocol as if the agents had constructed these insincere revelations themselves. Under this new protocol, each agent is motivated to reveal his type truthfully in a single step because the protocol will take care of optimally lying on the agent's behalf.

The idea of incorporating the strategy generation into the protocol is problematic among computationally limited agents. In the original protocol it may have been infeasible or prohibitively costly for an agent to compute its best strategy. Therefore, in a complex protocol the agents might not play the equilibrium. This may be undesirable because agents may play uncoordinated strategies leading to undesirable outcomes. On the other hand, if most equilibrium outcomes are undesirable, the protocol designer can construct a complex protocol—where agents cannot find the equilibrium—in the hope that a more desirable outcome will emerge. In the revised protocol of the proof, it is assumed that the protocol can solve for the equilibrium. However, if computation is costly, who pays for the computation that is required to solve for equilibrium? For some protocols, solving for equilibrium might be hard or even noncomputable.

The Nash equilibrium version of Theorem 5.2 has additional weaknesses. First, it assumes that the agents and the protocol designer have common knowledge about the joint probabilities of the agents' types. Second, the revised protocol may have other Nash equilibria in addition to the truthful one: Theorem 5.2 only says that a truthful one exists. This problem can be partially alleviated by what are called *augmented revelation mechanisms* [35].

While Theorem 5.2 is positive in nature, the following negative result establishes that in the general case, non-manipulable protocols are dictatorial:

Theorem 5.3 Gibbard-Satterthwaite impossibility theorem
Let each agent's type θ_i consist of a preference order $\succ_i$ on O. Let there be no restrictions on $\succ_i$, i.e. each agent may rank the outcomes O in any order.[2] Let $|O| \geq 3$. Now, if the social choice function $f(\cdot)$ is truthfully implementable in a dominant strategy equilibrium, then $f(\cdot)$ is dictatorial, i.e. there is some agent i who gets (one of) his most preferred outcomes chosen no matter what types the others reveal [18, 70].

Circumventing the Gibbard-Satterthwaite Impossibility Theorem: Restricted Preferences and the Groves-Clarke Tax Mechanism

The design of nonmanipulable protocols is not as impossible as it may seem in light of Theorem 5.3. The individual preferences may happen to belong to some restricted domain—thus invalidating the conditions of the impossibility theorem—and it is known that there are islands in the space of agents' preferences for which nonmanipulable nondictatorial protocols can be constructed.

Let us go through an example. Let the outcomes be of the form $o = (g, \pi_1, \ldots, \pi_{|A|})$, where π_i is the amount of some divisible numeraire (e.g. money) that agent i receives in the outcome, and g encodes the other features of the outcome. The agents' preferences are called *quasilinear* if they can be represented by utility functions of the form $u_i(o) = v_i(g) + \pi_i$.

For example, in voting whether to build a joint pool, say $g = 1$ if the pool is built and $g = 0$ if not. Call each agent's gross benefit from the pool $v_i^{gross}(g)$, and say that the cost P of the pool would be divided equally among the agents, i.e. $\pi_i = -P/|A|$. So, an agent's (net) benefit is $v_i(g) = v_i^{gross}(g) - P/|A|$.

---

2. Theorem 5.3 applies even if each agent's preferences are restricted to being complete, transitive, and strict.

Every agent $i \in A$ reveals his valuation $\hat{v}_i(g)$ for every possible g
The social choice is $g^* = \arg\max_g \sum_i \hat{v}_i(g)$
Every agent is levied a tax: $tax_i = \sum_{j \neq i} \hat{v}_j(g^*) - \sum_{j \neq i} \hat{v}_j(\arg\max_g \sum_{k \neq i} \hat{v}_k(g))$

Algorithm 5.1 The Clarke tax algorithm.

Quasilinearity of the environment would require several things. First, no agent should care how others divide payoffs among themselves. This might be violated e.g. if an agent wants his enemies to pay more than his friends. Second, an agent's valuation $v_i^{gross}(g)$ of the pool should not depend on the amount of money that the agent will have. This might be violated for example if rich agents have more time to enjoy the pool because they do not have to work.

When voting whether to build the pool or not, the agents that vote for the pool impose an externality on the others because the others have to pay as well. On the other hand, if only the pro-pool voters would have to pay, there would be an incentive for them to vote for no pool, and free ride the pool that might be built anyway due to the others' votes. The solution is to make the agents precisely internalize this externality by imposing a tax on those agents whose vote changes the outcome. The size of an agent's tax is exactly how much his vote lowers the others' utility. Agents that do not end up changing the outcome do not pay any tax.

Theorem 5.4
If each agent has quasilinear preferences, then, under Algorithm 5.1, each agent's dominant strategy is to reveal his true preferences, i.e. $\hat{v}_i(g) = v_i(g)$ for all g. [11, 21]

So, in the example, if the pool is built, the utility for each agent i becomes $u_i(o) = v_i(1) - P/|A| - tax_i$, and if not, $u_i(o) = v_i(0)$.

The mechanism leads to the socially most preferred g to be chosen. Also, because truthtelling is every agents dominant strategy, the agents need not waste effort in counterspeculating each others' preference declarations. Furthermore, participation in the mechanism can only increase an agent's utility, which makes participation individually rational.

Unfortunately the mechanism does not maintain budget balance: too much tax is collected. There are other truth-dominant algorithms for this problem where too little tax is collected (negative taxes are paybacks), but none that guarantee that the sum of the taxes is zero. The schemes where too little is collected require an external benefactor to operate. The schemes that collect too much are not Pareto efficient because the extra tax revenue has to be burnt. It cannot be given back to the agents or donated to any cause that any of the agents care about. Such redistribution would affect the agents' utilities, and truthtelling would no longer be dominant.

Another problem with Algorithm 5.1 is that it is not coalition proof. Some coalition of voters might coordinate their insincere preference revelations and achieve higher utilities. Table 5.3 presents a 3-agent example of this where the cost of building the pool is $P = 9,000$, and $v_i(0) = 0$ for every agent. We study the case where agents 1 and 2 collude.

			No collusion				Agents 1 and 2 collude			
i	$v_i^{gross}(1)$	$v_i(1)$	$\hat{v}_i(1)$	g^*	tax_i	u_i	$\hat{v}_i(1)$	g^*	tax_i	u_i
1	$5,000$	$2,000$	$2,000$		$1,500$	500	$2,500$		1000	$1,000$
2	$4,000$	$1,000$	$1,000$	$1(build)$	500	500	$1,500$	$1(build)$	0	$1,000$
3	500	$-2,500$	$-2,500$		0	$-2,500$	$-2,500$		0	$-2,500$

Table 5.3 Example of collusion in the Clarke tax algorithm.

Traditionally, the Clarke tax mechanism has been used to solve a single isolated social choice problem. In multiagent planning—e.g. in AI—this would mean voting over all possible multiagent plans. This is is often intractable. To reduce this complexity, Ephrati has used a variant of the method where the agents repeatedly use the Clarke tax mechanism to do planning over one timestep of the plan at a time [14, 15, 13]. In such multistep approaches one has to be careful that truthtelling is still a dominant strategy. If the outcomes of the different votings are not independent in value to every agent, there is a risk e.g. that an agent will speculatively reveal higher than truthful valuations for some outcomes because he anticipates future outcomes that will be synergic with those particular ones of the currently available outcomes.

Other Ways to Circumvent the Gibbard-Satterthwaite Impossibility Theorem

Even if the agents do not happen to have preferences that are restricted in some particular way that allows one to avoid the negative conclusion of Theorem. 5.3, there are ways to circumvent the seemingly unavoidable tradeoff between manipulability and dictatoriality.

For example, *ex ante* fairness can be achieved by choosing the dictator randomly in the protocol. This can be done via a protocol where every agent submits a vote into a hat, and the decisive vote is pulled out of the hat at random. Clearly, each agent's dominant strategy is to vote truthfully: if his vote gets chosen, he would have been best off voting for his most preferred alternative, and if his vote is not chosen, it does not matter what the agent voted for.

Another possible way of getting around Theorem. 5.3 is to use a protocol for which computing an untruthful revelation—that is better than the truthful one—is prohibitively costly computationally. One difficulty with this approach is that to guarantee that an agent can never manipulate, manipulation would have to be provably hard for every instance (combination of agents' preferences), not just in the worst case. Another difficulty is that even if it were possible to prove that deterministically finding a beneficial manipulation is hard, the agent can (e.g. randomly) generate insincere revelations, and simulate the protocol (given that the others' strategies cannot matter in a dominant strategy equilibrium) to check whether his guessed manipulations are beneficial.

5.4 Auctions

Within mechanism design, auctions provide a special setting which is important and often relatively easily analyzable. Auctions also have many practical computer science applications [60, 81, 37, 25], and several successful web sites exist for buying and selling items using auction protocols. Unlike voting where the outcome binds all agents, in auctions the outcome is usually a deal between two agents: the auctioneer and one bidder. Also, in voting the protocol designer is assumed to want to enhance the social good, while in auctions, the auctioneer wants to maximize his own profit.

Auction theory analyzes protocols and agents' strategies in auctions. An auction consists of an auctioneer and potential bidders. Auctions are usually discussed in situations where the auctioneer wants to sell an item and get the highest possible payment for it while the bidders want to acquire the item at the lowest possible price. The discussion of this section will pertain to the classical setting, although in a contracting setting, the auctioneer wants to subcontract out tasks at the lowest possible price while the bidders who handle the tasks want to receive the highest possible payment for doing so. The mechanisms for the latter setting are totally analogous to mechanisms for the former.

5.4.1 Auction Settings

There are three qualitatively different auction settings depending on how an agent's value (monetary equivalent of utility) of the item is formed.

In *private value* auctions, the value of the good depends only on the agent's own preferences. An example is auctioning off a cake that the winning bidder will eat. The key is that the winning bidder will not resell the item or get utility from showing it off to others, because in such cases the value would depend on other agents' valuations (a valuation is the monetary equivalent of expected utility). The agent is often assumed to know its value for the good exactly.

On the other hand, in *common value* auctions, an agent's value of an item depends entirely on other agents' values of it, which are identical to the agent's by symmetry of this criterion. For example, auctioning treasury bills fulfills this criterion. Nobody inherently prefers having the bills, and the value of the bill comes entirely from reselling possibilities.

In *correlated value* auctions, an agent's value depends partly on its own preferences and partly on others' values. For example, a negotiation within a contracting setting fulfills this criterion. An agent may handle a task itself in which case the agent's local concerns define the cost of handling the task. On the other hand, the agent can recontract out the task in which case the cost depends solely on other agents' valuations.

The next section discusses different auction protocols. Those protocols have different properties under the three different auction settings presented above.

5.4.2 Auction Protocols

In the *English (first-price open-cry) auction*, each bidder is free to raise his bid. When no bidder is willing to raise anymore, the auction ends, and the highest bidder wins the item at the price of his bid. An agent's strategy is a series of bids as a function of his private value, his prior estimates of other bidder's valuations, and the past bids of others. In private value English auctions, an agent's dominant strategy is to always bid a small amount more than the current highest bid, and stop when his private value price is reached. In correlated value auctions the rules are often varied to make the auctioneer increase the price at a constant rate or at a rate he thinks appropriate. Also, sometimes *open-exit* is used where a bidder has to openly declare exiting without a re-entering possibility. This provides the other bidders more information regarding the agent's valuation.

In the *first-price sealed-bid auction*, each bidder submits one bid without knowing the others' bids. The highest bidder wins the item and pays the amount of his bid. An agent's strategy is his bid as a function of his private value and prior beliefs of others' valuations. In general there is no dominant strategy for bidding in this auction. An agent's best strategy is to bid less than his true valuation, but how much less depends on what the others bid. The agent would want to bid the lowest amount that still wins the auction—given that this amount does not exceed his valuation. With common knowledge assumptions regarding the probability distributions of the agents' values, it is possible to determine Nash equilibrium strategies for the agents. For example, in a private value auction where the valuation v_i for each agent i is drawn independently from a uniform distribution between 0 and $\bar{v}$, there is a Nash equilibrium where every agent i bids $\frac{|A|-1}{|A|}v_i$, see [54].

In the *Dutch (descending) auction*, the seller continuously lowers the price until one of the bidders takes the item at the current price. The Dutch auction is strategically equivalent to the first-price sealed-bid auction, because in both games, an agent's bid matters only if it is the highest, and no relevant information is revealed during the auction process. Dutch auctions are efficient in terms of real time because the auctioneer can decrease the price at a brisk pace. You can observe this e.g. by participating in a Dutch auction simulation at http://www.mcsr.olemiss.edu/ ccjimmy/auction.

In the *Vickrey (second-price sealed-bid) auction*, each bidder submits one bid without knowing the others' bids. The highest bidder wins, but at the price of the second highest bid [80, 42]. An agent's strategy is his bid as a function of his private value and prior beliefs of others' valuations.

Theorem 5.5
A bidder's dominant strategy in a private value Vickrey auction is to bid his true valuation [80].[3]

If he bids more than his valuation, and the increment made the difference between winning or not, he will end up with a loss if he wins. If he bids less, there is a smaller chance of winning, but the winning price is unaffected[4]. Theorem 5.5 means that an agent is best off bidding truthfully no matter what the other bidders are like: what are their capabilities, operating environments, bidding plans, *etc.* This has two desirable sides. First, the agents reveal their preferences truthfully which allows globally efficient decisions to be made. Second, the agents need not waste effort in counterspeculating other agents because they do not matter in making the bidding decision.

Vickrey auctions have been widely advocated and adopted for use in computational multiagent systems. For example, versions of the Vickrey auction have been used to allocate computation resources in operating systems [81], to allocate bandwidth in computer networks [37], and to computationally control building heating [25]. On the other hand, Vickrey auctions have not been widely adopted in auctions among humans [57, 58] even though the protocol was invented over 25 years ago [80]. Limitations of the Vickrey auction protocol—especially in computational multiagent systems—are discussed in [61].

All-pay auctions are another family of auction protocols. In such mechanisms, each participating bidder has to pay the amount of his bid (or some other amount) to the auctioneer. The schemes have been used in computational multiagent systems for tool reallocation [36]. These methods are often susceptible to infinite escalations of bids [53], and will not be discussed further here.

5.4.3 Efficiency of the Resulting Allocation

In isolated private value or common value auctions, each one of the four auction protocols (English, Dutch, first-price sealed-bid, and Vickrey) allocates the auctioned item Pareto efficiently to the bidder who values it the most.[5] Although all

3. If the bidders know their own values, this result does not depend on the bidders' risk neutrality. On the other hand, if a bidder has some uncertainty about his own valuation, this result only holds for a risk-neutral bidder: e.g. a risk averse bidder can be better off by bidding less than his expected valuation [61].
4. In private value auctions, the Vickrey auction is strategically equivalent to the English auction. They will produce the same allocation at the same prices. On the other hand, in correlated value auctions, the other agents' bids in the English auction provide information to the agent about his own valuation. Therefore English and Vickrey auctions are not strategically equivalent in general, and may lead to different results.
5. This holds at least as long as the auctioneer always sells the item. On the other hand, if the auctioneer has a reservation price, he may inefficiently end up with the item even though the highest bidder really values the item more than the auctioneer.

four are Pareto efficient in the allocation, the ones with dominant strategies (Vickrey auction and English auction) are more efficient in the sense that no effort is wasted in counterspeculating the other bidders.

5.4.4 Revenue Equivalence and Non-Equivalence

One could imagine that the first-price auctions give higher expected revenue to the auctioneer because in second-price auctions the auctioneer only gets the second price. On the other hand, in first-price auctions the bidders underbid while in the second-price auctions they bid truthfully. Now, which of these effects is stronger, i.e. which protocol should the auctioneer choose to maximize his expected revenue? It turns out that the two effects are exactly equally strong:

Theorem 5.6 Revenue equivalence
All of the four auction protocols produce the same expected revenue to the auctioneer in private value auctions where the values are independently distributed, and bidders are risk-neutral [80, 42, 54].

Among risk averse bidders, the Dutch and the first-price sealed-bid protocols give higher expected revenue to the auctioneer than the Vickrey or English auction protocols. This is because in the former two protocols, a risk averse agent can insure himself by bidding more than what is optimal for a risk-neutral agent. On the other hand, a risk averse auctioneer achieves higher expected utility via the Vickrey or English auction protocols than via the Dutch or the first-price sealed-bid protocol.

The fact that revenue equivalence holds in private value auctions does not mean that it usually holds in practice: most auctions are not pure private value auctions. In non-private value auctions with at least three bidders, the English auction (especially the open-exit variant) leads to higher revenue than the Vickrey auction. The reason is that other bidders willing to go high up in price causes a bidder to increase his own valuation of the auctioned item. In this type of auctions, both the English and the Vickrey protocols produce greater expected revenue to the auctioneer than the first-price sealed-bid auction—or its equivalent, the Dutch auction.

5.4.5 Bidder Collusion

One problem with all four of the auction protocols (English, Dutch, first-price sealed-bid, and Vickrey) is that they are not collusion proof. The bidders could coordinate their bid prices so that the bids stay artificially low. In this manner, the bidders get the item at a lower price than they normally would.

The English auction and the Vickrey auction actually self-enforce some of the most likely collusion agreements. Therefore, from the perspective of deterring collusion, the first-price sealed-bid and the Dutch auctions are preferable. The following example from [54] shows this. Let bidder Smith have value 20, and every other bidder have value 18 for the auctioned item. Say that the bidders collude by deciding that Smith will bid 6, and everyone else will bid 5. In an English auction this is self-enforcing, because if one of the other agents exceeds 5, Smith will observe this, and will be willing to go all the way up to 20, and the cheater will not gain anything from breaking the coalition agreement. In the Vickrey auction, the collusion agreement can just as well be that Smith bids 20, because Smith will get the item for 5 anyway. Bidding 20 removes the incentive from any bidder to break the coalition agreement by bidding between 5 and 18, because no such bid would win the auction. On the other hand, in a first-price sealed-bid auction, if Smith bids anything below 18, the other agents have an incentive to bid higher than Smith's bid because that would cause them to win the auction. The same holds for the Dutch auction.

However, for collusion to occur under the Vickrey auction, the first-price sealed-bid auction, or the Dutch auction, the bidders need to identify each other before the submission of bids—otherwise a non-member of the coalition could win the auction. On the other hand, in the English auction this is not necessary, because the bidders identify themselves by shouting bids. To prevent this, the auctioneer can organize a computerized English auction where the bidding process does not reveal the identities of the bidders.

5.4.6 Lying Auctioneer

Insincerity of the auctioneer may be a problem in the Vickrey auction. The auctioneer may overstate the second highest bid to the highest bidder unless that bidder can verify it. An overstated second offer would give the highest bidder a higher bill than he would receive if the auctioneer were truthful. Cheating by the auctioneer has been suggested to be one of the main reasons why the Vickrey auction protocol has not been widely adopted in auctions among humans [58]. To solve the problem, cryptographic electronic signatures could be used by the bidders so that the auctioneer could actually present the second best bid to the winning bidder—and would not be able to alter it. The other three auction protocols (English, Dutch, and first-price sealed-bid) do not suffer from lying by the auctioneer because the highest bidder gets the item at the price of his bid.

In non-private value auctions with the English (or all-pay) auction protocol, the auctioneer can use *shills* that bid in the auction in order to make the real bidders increase their valuations of the item. This is not possible in the sealed-bid protocols or the Dutch protocol, because the bidders do not observe the others' bids.

The auctioneer may also have other tools at his disposal. For example, he may place a bid himself to guarantee that the item will not be sold below a certain price (this can also be achieved by having a reservation price which may or may not be public to the bidders). However, for example in the Vickrey auction, the auctioneer is motivated to bid more than his true reservation price. This is because there is a chance that his bid will be second highest in which case it determines the item's price. Such overbidding leads to the possibility that the auctioneer ends up inefficiently keeping the item even though some bidders' valuations exceed his true reservation price.

5.4.7 Bidders Lying in Non-Private-Value Auctions

Most auctions are not pure private value auctions: an agent's valuation of a good depends at least in part on the other agents' valuations of that good. For example in contracting settings, a bidder's evaluation of a task is affected by the prices at which the agent can subcontract the task or parts of it out to other agents. This type of recontracting is commonly allowed in automated versions of the contract net protocol also [60, 77].

Common value (and correlated value) auctions suffer from the *winner's curse.* If an agent bids its valuation and wins the auction, it will know that its valuation was too high because the other agents bid less. Therefore winning the auction amounts to a monetary loss. Knowing this in advance, agents should bid less than their valuations [42, 54]. This is the best strategy in Vickrey auctions also. So, even though the Vickrey auction promotes truthful bidding in private-value auctions, it fails to induce truthful bidding in most auction settings.

5.4.8 Undesirable Private Information Revelation

Because the Vickrey auction has truthful bidding as the dominant strategy in private value auctions, agents often bid truthfully. This leads to the bidders revealing their true valuations. Sometimes this information is sensitive, and the bidders would prefer not to reveal it. For example, after winning a contract with a low bid, a company's subcontractors figure out that the company's production cost is low, and therefore the company is making larger profits than the subcontractors thought. It has been observed that when such auction results are revealed, the subcontractors will want to renegotiate their deals to get higher payoff [58]. This has been suggested—along with the problem of a lying auctioneer—as one of the main reasons why the Vickrey auction protocol is not widely used in auctions among humans [58]. First-price auction protocols do not expose a bidder's valuation as clearly because the bid is based on the agent's model of other bidders, and this (possibly inaccurate) model is not known by the subcontractors. Therefore, these auction types may be more desirable than the Vickrey auction when valuations are sensitive.

5.4.9 Roles of Computation in Auctions

Auction theory does not usually study the computational aspects of auctions. However, from a DAI perspective they are crucial. Two issues that arise from computation in auctions will be discussed: the computationally complex lookahead that arises when auctioning interrelated items one at a time, and the implications of costly local marginal cost (valuation) computation or information gathering in a single-shot auction.

Inefficient Allocation and Lying in Interrelated Auctions

In addition to single-item auctions, Vickrey auctions have been widely studied in the allocation of multiple items of a homogeneous good [42], and the dominance of truth-teling can be maintained. However, the case of auctioning heterogeneous interrelated goods has received less attention. This is the setting of many real world problems, including several where computational agents are used [66, 68, 67, 60, 62, 56].

This section discusses cases where heterogeneous items are auctioned one at a time, and an agent's valuations of these items are interdependent (not additive). This occurs for example in task allocation in transportation problems. Figure 5.2 presents a simple example of such a problem with two delivery tasks: t_1 and t_2. Task t_1 is auctioned before t_2. The auctioneer wants to get the tasks handled while paying agents 1 and 2 as little as possible for handling them. The initial locations of the two agents are presented in the figure. To handle a task, an agent needs to move to the beginning of the delivery task (arrow), and take a parcel from there to the end of the arrow. An agent's movement incurs the same cost irrespective of whether it is carrying a parcel. The agents need not return to their initial locations. The costs for handling tasks (subscripted by the name of the agent) can be measured from the figure: $c_1(\{t_1\}) = 2$, $c_1(\{t_2\}) = 1$, $c_1(\{t_1, t_2\}) = 2$, $c_2(\{t_1\}) = 1.5$, $c_2(\{t_2\}) = 1.5$, and $c_2(\{t_1, t_2\}) = 2.5$. Say that these costs are common knowledge to the agents. Clearly the globally optimal allocation is the one where agent 1 handles both tasks.

This allocation is not reached if agents treat the auctions independently and bid truthfully [61]. In the first auction of the example, task t_1 is allocated. Agent 1 bids $c_1(\{t_1\}) = 2$, and agent 2 bids $c_2(\{t_1\}) = 1.5$. The task is allocated to agent 2. In the second auction, task t_2 is allocated. Agent 1 bids $c_1(\{t_2\}) = 1$, and agent 2 bids $c_2(\{t_2\}) = 1.5$, so t_2 is allocated to agent 1. The resulting allocation of the two tasks is suboptimal. If agent 2 takes the ownership of t_1 into account when bidding for t_2, then it will bid $c_2(\{t_1, t_2\}) - c_2(\{t_1\}) = 2.5 - 1.5 = 1$. In this case t_2 may be allocated to either agent. In both cases the resulting allocation of the two tasks is still suboptimal.

Alternatively, the agents could incorporate full lookahead into their auction strategies. This way the optimal allocation is reached, but agents do not bid their true per-item costs [61]. In the last auction of the example, an agent is best off bidding its own costs that takes into account the tasks that the agent

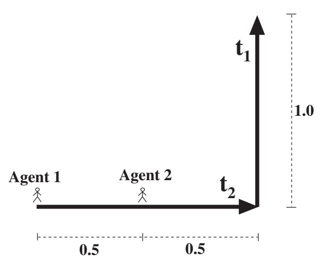

Figure 5.2 Small example problem with two agents and two delivery tasks.

already has. Let us look at the auction of t_2. If agent 1 has t_1, it will bid $c_1(\{t_1, t_2\}) - c_1(\{t_1\}) = 2 - 2 = 0$, and $c_1(\{t_2\}) = 1$ otherwise. If agent 2 has t_1, it will bid $c_2(\{t_1, t_2\}) - c_2(\{t_1\}) = 2.5 - 1.5 = 1$, and $c_2(\{t_2\}) = 1.5$ otherwise. So, if agent 1 has t_1, it will win t_2 at the price 1.5, and get a payoff of $1.5 - 0 = 1.5$ in the second auction, while agent 2 gets zero. On the other hand, if agent 2 has t_1, the bids for t_2 are equal, and both agents get a zero payoff in the second auction irrespective of who t_2 gets allocated to. Therefore it is known that getting t_1 in the first auction is worth an extra 1.5 to agent 1 while nothing extra to agent 2. So, in the auction for t_1, agent 1's dominant strategy is to bid $c_1(\{t_1\}) - 1.5 = 2 - 1.5 = 0.5$. This is lower than agent 2's bid $c_2(\{t_1\}) - 0 = 1.5 - 0 = 1.5$, so agent one gets t_1. In the second auction agent 1 gets t_2 as discussed above. So the globally optimal allocation is reached. However, agent 1 bids 0.5 for t_1 instead of 2, which would be the truthful bid if the auctions were treated independently without lookahead.

Put together, lookahead is a key feature in auctions of multiple interrelated items. To date it has not been adequately addressed in computational multiagent systems that use Vickrey auctions, and it is a common misunderstanding that Vickrey auctions promote single-shot truth-telling even in interrelated auctions. In auctions by humans, such interrelationships are sometimes addressed by allowing a bidder to pool all of the interrelated items under one *entirety bid* [42]. Another method for enhancing the efficiency of interrelated auctions is to allow agents to backtrack from commitments by paying penalties. This allows a winning agent to beneficially decommit from an auctioned item in case that agent does not get synergic items from other related auctions [41, 67, 62]. This question will be revisited in Section 5.7.

While avoidance of counterspeculation was one of the original reasons suggested for adopting the Vickrey auction, lookahead requires speculation in the sense of trying to guess which items are going to be auctioned in the future, and which agents are going to win those auctions. Other speculative issues in sequential Vickrey auctions have been discussed for example in [28].

Even under complete information, the computational cost of full lookahead (searching the game tree which is deep if there are many items to be auctioned sequentially) may be prohibitively great. Further work is required to devise methods for controlling the search: the advantages of (partial) lookahead should be traded off against the cost.

Counterspeculation When Computing One's Valuation

Sometimes even the Vickrey auction protocol fails to avoid counterspeculation—even in a single-shot auction. Let us look at a situation where an agent has uncertainty regarding its own valuation of the auction item, but can pay to remove this uncertainty. This situation often occurs among computational agents, where the value of a good (or task contract [66, 68, 67, 60, 62, 56]) can only be determined via carrying out a costly computation—e.g. a solution of a combinatorial problem [60, 66]. Alternatively the payment can be viewed as the cost of solving a prediction problem, or as the cost of performing an information gathering action, or as the cost paid to an expert oracle.

Theorem 5.7 Incentive to counterspeculate
In a single-shot private value Vickrey auction with uncertainty about an agent's own valuation, a risk neutral agent's best (deliberation or information gathering) action can depend on the other agents. It follows that (if counterspeculation is cheap enough) it is worth counterspeculating [61].

Proof by example. Let there be two bidders: 1 and 2. Let 1's valuation v_1 for the auctioned item be uniformly distributed between 0 and 1, i.e. agent 1 does not know its own valuation exactly. Let 2's exact valuation v_2 be common knowledge. Say $0 \leq v_2 < \frac{1}{2}$, which implies $E[v_1] > v_2$.

Let agent 1 have the choice of finding out its exact valuation v_1 before the auction by paying a cost c. Now, should agent 1 take this informative but costly action?

No matter what agent 1 chooses here, agent 2 will bid v_2 because bidding ones valuation is a dominant strategy in a single-shot private value Vickrey auction.

If agent 1 chooses not to pay c, agent 1 should bid $E[v_1] = \frac{1}{2}$, because bidding ones expected valuation is a risk neutral agent's dominant strategy in a single-shot private value Vickrey auction. Now agent 1 gets the item at price v_2. If agent 1's valuation v_1 turns out to be less than v_2, agent 1 will suffer a loss. Agent 1's expected payoff is

$$E[\pi_{noinfo}] = \int_0^1 v_1 - v_2 dv_1 = \frac{1}{2} - v_2$$

If agent 1 chooses to pay c for the exact information, it should bid v_1 because bidding ones valuation is a dominant strategy in a single-shot private value Vickrey auction. Agent 1 gets the item if *and only if* $v_1 \geq v_2$. Note that now the agent has no chance of suffering a loss, but on the other hand it has invested c in the information. Agent 1's expected payoff is

$$E[\pi_{info}] = \int_0^{v_2} -c dv_1 + \int_{v_2}^1 v_1 - v_2 - c dv_1 = \frac{1}{2}v_2^2 - v_2 + \frac{1}{2} - c$$

Agent 1 should choose to buy the information iff

$$\begin{aligned} & E[\pi_{info}] \geq E[\pi_{noinfo}] \\ \Leftrightarrow\ & \frac{1}{2}v_2^2 - v_2 + \frac{1}{2} - c \geq \frac{1}{2} - v_2 \\ \Leftrightarrow\ & \frac{1}{2}v_2^2 \geq c \\ \Leftrightarrow\ & v_2 \geq \sqrt{2c} \qquad \text{(because } v_2 \geq 0\text{)} \end{aligned}$$

So, agent 1's best choice of action depends on agent 2's valuation v_2. Therefore, agent 1 can benefit from counterspeculating agent 2. ■

5.5 Bargaining

In a *bargaining* setting, agents can make a mutually beneficial agreement, but have a conflict of interest about which agreement to make. In classical microeconomics, assumptions of monopoly (or monopsony) or perfect competition are often made. A monopolist gets all of the gains from interaction while an agent facing perfect competition can make no profit. Real world settings usually consist of a finite number of competing agents, so neither monopoly nor perfect competition assumptions strictly apply. Bargaining theory fits in this gap [50]. There are two major subfields of bargaining theory: axiomatic and strategic.

5.5.1 Axiomatic Bargaining Theory

Unlike noncooperative (strategic) game theory, *axiomatic bargaining theory* does not use the idea of a solution concept where the agents' strategies form some type of equilibrium. Instead, desirable properties for a solution, called axioms of the bargaining solution, are postulated, and then the solution concept that satisfies these axioms is sought [50, 30, 54, 51].

The *Nash bargaining solution* is a historically early solution concept that uses this approach. Nash analyzed a 2-agent setting where the agents have to decide on an outcome $o \in O$, and the fallback outcome $o_{fallback}$ occurs if no agreement is reached. There is a utility function $u_i : O \rightarrow \Re$ for each agent $i \in [1, 2]$. It is assumed that that the set of feasible utility vectors $\{(u_1(o), u_2(o)) | o \in O\}$ is convex. This occurs, for example, if outcomes include all possible lotteries over actual alternatives.

When many deals are individually rational—i.e. have higher utility than the fallback—to both agents, multiple Nash equilibria often exist. For example, if the agents are bargaining over how to split a dollar, all splits that give each agent more than zero are in equilibrium. If agent one's strategy is to offer ρ and no more, agent two's best response is to take the offer as opposed to the fallback which is zero. Now, one's best response to this is to offer ρ and no more. Thus, a Nash equilibrium exists for any ρ that defines a contract that is individually rational for both agents, and feasible ($0 < \rho < 1$). Due to the nonuniqueness of the equilibrium, a stronger (axiomatic) solution concept such as the Nash bargaining solution is needed to prescribe a unique solution.

The axioms of the Nash bargaining solution $u^* = (u_1(o^*), u_2(o^*))$ are:

- Invariance: The agents' numeric utility functions really only represent ordinal preferences among outcomes—the actual cardinalities of the utilities do not matter. Therefore, it should be possible to transform the utility functions in the following way: for any strictly increasing linear function f, $u^*(f(o), f(o_{fallback})) = f(u^*(o, o_{fallback}))$.
- Anonymity (symmetry): switching labels on the players does not affect the outcome.

- Independence of irrelevant alternatives: if some outcomes o are removed, but o^* is not, then o^* still remains the solution.
- Pareto efficiency: it is not feasible to give both players higher utility than under $u^* = (u_1(o^*), u_2(o^*))$.

Theorem 5.8 Nash bargaining solution
The unique solution that satisfies these four axioms is [47]:

$$o^* = \arg\max_o [u_1(o) - u_1(o_{fallback})][u_2(o) - u_2(o_{fallback})]$$

The Nash bargaining solution can be directly extended to more than two agents, as long as the fallback occurs if at least one agent disagrees. The 2-agent Nash bargaining solution is also the 2-agent special case of the Shapley value—a particular solution concept for payoff division in coalition formation, discussed later in Section 5.8.3—where coalitions of agents can cooperate even if all agents do not agree.

Other bargaining solutions also exist. They postulate different desiderata as axioms and arrive at a different utility combination as the outcome [30].

5.5.2 Strategic Bargaining Theory

Unlike axiomatic bargaining theory, *strategic bargaining theory* does not postulate desiderata as axioms on the solution concept. Instead, the bargaining situation is modeled as a game, and the solution concept is based on an analysis of which of the players' strategies are in equilibrium. It follows that for some games, the solution is not unique. On the other hand, strategic bargaining theory explains the behavior of rational utility maximizing agents better than axiomatic approaches. The latter are not based on what the agents can choose for strategies, but instead rely on the agents pertaining to axiomatic, imposed notions of fairness.

Strategic bargaining theory usually analyses sequential bargaining where agents alternate in making offers to each other in a prespecified order [50, 54, 51, 35]. Agent 1 gets to make the first offer. As an example, one can again think of deciding how to split a dollar. In a protocol with a finite number of offers and no time discount, the unique payoffs of the subgame perfect Nash equilibria are such that the last offerer will get the whole dollar (minus ϵ), because the other agent is better off accepting ϵ than by rejecting and receiving nothing. For simplicity in the rest of this section, say that in similar situations, ϵ can be zero, and the other agent will still accept.

A time discount factor δ can be incorporated in the model. In round 1 the dollar is worth 1, in round two it is worth δ, in round three it is worth δ^2, and so on. With time discount, a subgame perfect Nash equilibrium of a finite game of length T can be solved starting from the end. For example, if $\delta = 0.9$, then Table 5.4 represents the offerer's maximal claims that are acceptable to the other agent. In the last round, 2 would again accept zero. However, in the next to last round, 2 could keep 0.1, because it knows that this is how much 1 would loose by waiting to the next round. The same reasoning works for the previous rounds.

Round	1's share	2's share	Total value	Offerer
⋮	⋮	⋮	⋮	⋮
$T-3$	0.819	0.181	0.9^{T-4}	2
$T-2$	0.91	0.09	0.9^{T-3}	1
$T-1$	0.9	0.1	0.9^{T-2}	2
T	1	0	0.9^{T-1}	1

Table 5.4 Offerer's maximal acceptable claims in a finite game.

Round	1's share	2's share	Offerer
⋮	⋮	⋮	⋮
$t-2$	$1-\delta_2(1-\delta_1\bar{\pi}_1)$		1
$t-1$		$1-\delta_1\bar{\pi}_1$	2
t	$\bar{\pi}_1$		1
⋮	⋮	⋮	⋮

Table 5.5 Offerer's maximal acceptable claims in an infinite game with different discount factors.

When the protocol in a non-discounted setting allows an infinite number of bargaining rounds, the solution concept is powerless because any split of the dollar can be supported in subgame perfect Nash equilibrium—just as in the single-shot case. On the other hand, with discounting, even the infinite game can be solved:

Theorem 5.9 Rubinstein bargaining solution
In a discounted infinite round setting, the subgame perfect Nash equilibrium outcome is unique. Agent 1 gets $(1-\delta_2)/(1-\delta_1\delta_2)$, where δ_1 is 1's discount factor, and δ_2 is 2's. Agent 2 gets one minus this. Agreement is reached in the first round [59].

Proof Let us denote by $\bar{\pi}_1$ the maximum undiscounted share that 1 can get in any subgame perfect Nash equilibrium on his turn to offer. Following the same logic as in the example above, Table 5.5 can be filled. Now we have two ways to represent the maximum undiscounted share that 1 can get in any subgame perfect Nash equilibrium on his turn to offer. Setting them equal gives

$$\bar{\pi}_1 = 1-\delta_2(1-\delta_1\bar{\pi}_1) \Leftrightarrow \bar{\pi}_1 = \frac{1-\delta_2}{1-\delta_1\delta_2},$$

which is an upper bound for the undiscounted share that 1 can get in any subgame perfect Nash equilibrium on his turn to offer. But now we can go through the same argument by replacing $\bar{\pi}_1$ by $\underline{\pi}_1$, the minimum undiscounted share that 1 can get in any subgame perfect Nash equilibrium on his turn to offer. The minimum will equal the maximum, which completes the proof. ■

This proof technique allows one to solve for subgame perfect Nash equilibrium payoffs even though it is impossible to carry out complete lookahead in the game tree because it is infinitely long.

Another model of sequential bargaining does not use discounts, but assumes a fixed bargaining cost per negotiation round.

- If the agents have symmetric bargaining costs, the solution concept is again powerless because any split of the dollar can be supported in subgame perfect Nash equilibrium.
- If 1's bargaining cost c_1 is even slightly smaller than 2's cost c_2, then 1 gets the entire dollar. If 2 offered π in round t, then in period $t-1$, 1 could offer $1-\pi-c_2$, and keep $\pi + c_2$ to himself. In round $t-2$, 2 would offer $\pi + c_2 - c_1$, and keep $1 - \pi - c_2 + c_1$. Following this reasoning, in round $t - 2k$, agent 2 would get to keep $1 - \pi - k(c_2 - c_1)$ which approaches $-\infty$ as k increases. Realizing this, 2 would not bargain, but accept zero up front.
- If 1's bargaining cost is greater than 2's, then 1 receives a payoff that equals the second agent's bargaining cost, and agent 2 receives the rest. Agreement is again reached on the first round. This case is equivalent to the previous case except that the agent with the smaller bargaining cost is willing to give the other agent c_2 in order to avoid going through the first period of bargaining.

Kraus et al. have extended the work on sequential bargaining to the case with outside options [34]. They also analyze the case where one agent gains and one loses over time. Finally, they discuss negotiation over time when agents do not know each others' types.

5.5.3 Computation in Bargaining

All of the bargaining models discussed above assume perfect rationality from the agents. No computation is required in finding a mutually desirable contract. The space of deals is assumed to be fully comprehended by the agents, and the value of each potential contract known. On the other hand, future work should focus on developing methods where the cost of search (deliberation) for solutions is explicit, and it is decision-theoretically traded off against the bargaining gains that the search provides. This becomes particularly important as the bargaining techniques are scaled up to combinatorial problems with a multidimensional negotiation space as opposed to combinatorially simple ones like splitting the dollar.

There are actually two searches occurring in bargaining. In the intra-agent *deliberative search*, an agent locally generates alternatives, evaluates them, counterspeculates, does lookahead in the negotiation process *etc.* In the inter-agent *committal search*, the agents make (binding) agreements with each other regarding the solution. The agreements may occur over one part of the solution at a time. The agreed issues provide context for more focused intra-agent deliberative search—thus reducing the amount of costly computation required. The committal search may also involve iteratively renegotiating some parts of the solution that have already been agreed on, but have become less desirable in light of the newer agreements regarding other parts of the solution [60]. The two-search model proposed here is similar to the Real-Time A* search where an agent has to trade off thorough deliberation against more real-world actions [33]. Similarly, in modeling bargaining settings that require nontrivial computations, each agent's strategy should incorporate both negotiation actions and deliberation actions. The bargaining setting is more complex than the single agent setting of Real-Time A* in that there are multiple self-interested agents: the agents' strategies should be in equilibrium.

5.6 General Equilibrium Market Mechanisms

This section presents *general equilibrium theory*, a microeconomic market framework that has recently been successfully adapted for and used in computational multiagent systems in many application domains [82, 83, 44, 85, 10]. General equilibrium theory provides a distributed method for efficiently allocating goods and resources among agents—i.e. striking the best tradeoffs in a moderately complex multidimensional search space—based on market prices.

Such a market has $n > 0$ *commodity goods* g. The commodities can be physicals, e.g. coffee and meat, or they can be more abstract, e.g. parameters of an airplane design [83], flows in a traffic network [82], electricity in a power network [85], or mirror sites on the Internet [44]. The amount of each commodity is unrestricted, and each commodity is assumed arbitrarily divisible (continuous as opposed to discrete). Different elements within a commodity are not distinguishable, but different commodities are distinguishable from each other. The market also has prices $\mathbf{p} = [p_1, p_2, \ldots, p_n]$, where $p_g \in \Re$ is the price for good g.

The market can have two types of agents, *consumers* and *producers*. Each consumer i has a *utility function* $u_i(\mathbf{x_i})$ which encodes its preferences over different *consumption bundles* $\mathbf{x_i} = [x_{i1}, x_{i2}, \ldots, x_{in}]^T$, where $x_{ig} \in \Re_+$ is consumer i's allocation of good g. Each consumer i also has an initial *endowment* $\mathbf{e_i} = [e_{i1}, e_{i2}, \ldots, e_{in}]^T$, where $e_{ig} \in \Re$ is his endowment of commodity g.

The producers—if there are any—can use some commodities to produce others. Let $\mathbf{y_j} = [y_{j1}, y_{j2}, \ldots, y_{jn}]^T$ be the *production vector*, where y_{jg} is the amount of good g that producer j produces. Net usage of a commodity is denoted by a negative number. A producer's capability of turning inputs into outputs is characterized by its *production possibilities set* Y_j, which is the set of feasible production vectors. The *profit* of producer j is $\mathbf{p} \cdot \mathbf{y_j}$, where $\mathbf{y_j} \in Y_j$. The producer's profits are divided among the consumers according to predetermined proportions which need not be equal (one can think of the consumers owning stocks of the producers). Let θ_{ij} be the fraction of producer j that consumer i owns. The producers' profits are divided among consumers according to these shares. However, the consumers are assumed to have no say-so in the producers' production decisions.

Prices may change, and the agents may change their consumption and production plans, but actual production and consumption only occur once the market has reached a *general equilibrium*. We say that $(\mathbf{p}^*, \mathbf{x}^*, \mathbf{y}^*)$ is a general (Walrasian) equilibrium if

I markets clear:

$$\sum_i \mathbf{x_i^*} = \sum_i \mathbf{e_i} + \sum_j \mathbf{y_j^*}, \text{ and}$$

II each consumer i maximizes its preferences given the prices:

$$\mathbf{x_i^*} = \arg \max_{\mathbf{x_i} \in \Re_+^n \ | \ \mathbf{p}^* \cdot \mathbf{x_i} \leq \mathbf{p}^* \cdot \mathbf{e_i} + \sum_j \theta_{ij} \mathbf{p}^* \cdot \mathbf{y_j}} u_i(\mathbf{x_i}), \text{ and}$$

III each producer j maximizes its profits given the prices:

$$\mathbf{y_j^*} = \arg \max_{\mathbf{y_j} \in Y_j} \mathbf{p}^* \cdot \mathbf{y_j}$$

5.6.1 Properties of General Equilibrium

General equilibrium solutions have some very desirable properties:

Theorem 5.10 Pareto efficiency
Each general equilibrium is Pareto efficient, i.e. no agent can be made better off without making some other agent worse off [39].

This means that there is no possible methodology for finding solutions to the agents' problem such that every agent is better off than in the general equilibrium. The solution is also stable against collusion:

Theorem 5.11 Coalitional stability
Each general equilibrium with no producers is stable in the sense of the *core* solution concept of coalition formation games: no subgroup of consumers can increase their utilities by pulling out of the equilibrium and forming their own market [39].

The situation is more complex when producers are present: for example, if a set of consumers colludes, and they own part of a producer via the shares, what can the coalition produce?

Unfortunately, in some domains no general equilibrium exists. For example, it may be best for some producer to produce an infinite amount of some good. However, sufficient conditions for existence are known:

Theorem 5.12 Existence
Let the production possibilities sets be closed (i.e. include their boundaries), convex (i.e. if bundles y and y' are producible, then so is $\alpha y + (1-\alpha)y' \quad \forall \alpha \in [0,1]$),

and bounded above (i.e. an infinite amount of no good can be produced). Let the consumers' preferences be continuous (i.e. the preferences have no "jumps"), strictly convex (i.e. if the consumer prefers y to y'' and y' to y'', then he prefers $\alpha y+(1-\alpha)y'$ to y'' $\forall \alpha \in [0,1]$), and strongly monotone (i.e. each consumer strictly prefers more to less of each commodity). Now, if a society-wide bundle is producible where the amount of each commodity is positive (positive endowments trivially imply this), a general equilibrium exists [39].

For example, economies of scale in production violate convexity of production possibilities. Continuity of the consumer's preferences is violated e.g. in bandwidth allocation if an agent's welfare jumps as the threshold for being able to participate in a video conference is reached. Similarly, the consumer's preferences are not convex if the consumer starts to prefer a good (relative to other goods) more as he gets more of that good. Drugs and Web surfing are examples of this.

Even if a general equilibrium exists, it might not be unique. However, there is an easily understood sufficient condition for uniqueness:

Theorem 5.13 Uniqueness under gross substitutes
A general equilibrium is unique if the society-wide demand for each good is nondecreasing in the prices of the other goods [39].

For example, as the price of meat increases, consumers have to convert to satisfying their hunger with less expensive foods. It follows that the demand of potatoes increases. On the other hand, the conditions of this theorem are not always met. For example, as the price of bread increases, the demand of butter decreases. Complementarities are also very common in production, where the producers often need all of the inputs to create the outputs.

The basic general equilibrium framework does not account for *externalities*. In *consumption externalities*, one agent's consumption affects another agent's utility. In *production externalities*, one agent's production possibilities set is directly affected by another agent's actions. Glance and Hogg have presented examples of computational ecologies (not based on general equilibrium theory) where externalities are so dominant that, counterintuitively, adding resources to the system makes it operate less efficiently [19]. Hogg has also shown that externality problems are likely to be common in computational ecosystems [23]. Evolutionary aspects of such systems have also been discussed [43], and the behaviors under incomplete and delayed information analyzed [26]. Some mechanisms to attack externality problems include taxes and viewing some of the externality issues as commodities themselves [79].

5.6.2 Distributed Search for a General Equilibrium

The operational motivation behind market mechanisms is that the agents can find an efficient joint solution—which takes into account tradeoffs between agents and the fact that the values of different goods to a single agent may be interdependent—

```
Algorithm for the price adjustor:
    p_g = 1 for all g ∈ [1..n]
    Set λ_g to a positive number for all g ∈ [1..n − 1]
    Repeat
        Broadcast p to consumers and producers
        Receive a production plan y_j from each producer j
        Broadcast the plans y_j to consumers
        Receive a consumption plan x_i from each consumer i
        For g = 1 to n − 1
            p_g = p_g + λ_g(Σ_i(x_ig − e_ig) − Σ_j y_jg)
    Until |Σ_i(x_ig − e_ig) − Σ_j y_jg| < ε for all g ∈ [1..n]
    Inform consumers and producers that an equilibrium has been reached
Algorithm for consumer i:
    Repeat
        Receive p from the adjustor
        Receive a production plan y_j for each j from the adjustor
        Announce to the adjustor a consumption plan x_i ∈ ℜ^n_+ that
          maximizes u_i(x_i) given the budget constraint p·x_i ≤ p·e_i + Σ_j θ_ij p·y_j
    Until informed that an equilibrium has been reached
    Exchange and consume
Algorithm for producer j:
    Repeat
        Receive p from the adjustor
        Announce to the adjustor a production plan y_j ∈ Y_j that maximizes p·y_j
    Until informed that an equilibrium has been reached
    Exchange and produce
```

Algorithm 5.2 The distributed price tâtonnement algorithm.

while never centralizing all the information or control. There are many algorithms that can be used to search for a general equilibrium, some centralized, and some decentralized. The most common decentralized algorithm for this purpose is the *price tâtonnement process*, (Algorithm 5.2) which is a steepest descent search method.

Clearly, if no general equilibrium exists, no algorithm can find it. Furthermore, sometimes the price tâtonnement algorithm fails to find an equilibrium even if equilibria exist. However, there are sufficient conditions that guarantee that an equilibrium is found if it exists. One such sufficient condition is the gross substitutes property which was used in Theorem. 5.13. More generally,

Theorem 5.14 Convergence

The price tâtonnement algorithm convergences to a general equilibrium if $\mathbf{p}^* \cdot (\sum_i(\mathbf{x_i}(\mathbf{p}) - \mathbf{e_i}) - \sum_j \mathbf{y_j}(\mathbf{p})) > 0$ for all $\mathbf{p}$ not proportional to an equilibrium price vector $\mathbf{p}^*$ [39].

Strictly speaking, these convergence guarantees only apply to the continuous variant

$$\frac{dp_g}{dt} = \lambda_g(\sum_i(x_{ig}(\mathbf{p}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p})),$$

not to the more realistic discrete step version (Algorithm 5.2). However, these results suggest that even the discrete variant often converges—e.g. under gross substitutes—as long as the λ-multipliers in the algorithm are sufficiently small. If the λ-multipliers are too large, the search may keep "overshooting" the equilibrium. On the other hand, too small λ-multipliers will make the convergence slow. One potential solution to this problem is to dynamically adjust the step size, e.g. via the Newton method

$$\frac{dp_g}{dt} = -\lambda_g[J(\mathbf{p})]^{-1}(\sum_i(x_{ig}(\mathbf{p}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p})), \text{ where}$$

$$J(\mathbf{p}) = \begin{bmatrix} \frac{\partial(\sum_i(x_{i1}(\mathbf{p})-e_{i1})-\sum_j y_{j1}(\mathbf{p}))}{\partial p_1} & \cdots & \frac{\partial(\sum_i(x_{i1}(\mathbf{p})-e_{i1})-\sum_j y_{j1}(\mathbf{p}))}{\partial p_n} \\ \vdots & & \vdots \\ \frac{\partial(\sum_i(x_{in}(\mathbf{p})-e_{in})-\sum_j y_{jn}(\mathbf{p}))}{\partial p_1} & \cdots & \frac{\partial(\sum_i(x_{in}(\mathbf{p})-e_{in})-\sum_j y_{jn}(\mathbf{p}))}{\partial p_n} \end{bmatrix}$$

The Newton method often requires fewer iterations than steepest descent, but each iteration is computationally more intensive for the adjustor, and requires the computation and communication of the derivative information by the consumers and producers. One could conceptually take this information communication to the limit by having the producers and consumers submit their entire production and consumption functions (plans as a function of the possible price vectors), and the price adjustor could run a centralized search—with known efficient algorithms—for an equilibrium. However, this conflicts with one of the original motivations of market mechanisms: decentralization.

The tâtonnement process used in the WALRAS simulation [82] differs from Algorithm 5.2. WALRAS uses asynchronous declarations by the agents, i.e. an agent might only change its plan regarding a subset of goods at a time. Similarly, agents might take arbitrary turns in making new declarations. Under certain conditions, this process still converges to a general equilibrium [10]. As in tâtonnement, trades in WALRAS only occur after the market process has arrived (close) to a general equilibrium.

In addition to price-based market mechanisms, quantity-based (commodity-based, resource-based) mechanisms exist for reaching the general equilibrium [39]. In those mechanisms, the adjustor announces production and consumption plans, and the producers and consumers announce willingness to pay in terms of prices or marginal utilities. Unlike price-based algorithms, quantity-based algorithms maintain a feasible solution (once—e.g. up front—a feasible solution has been found) where markets clear at every iteration. This constitutes an interruptible anytime algorithm. Also, quantity-based algorithms offer the choice of carrying out the actual exchanges at every iteration or only at the end as in price-based algorithms. These advantages come at the cost of increased information centralization (communication). For example, the adjustor needs to know the production possibilities sets.

Most treatments of market-based search only discuss the complexity of finding an equilibrium once the agent's supply and demand functions are known. However, it may be computationally complex for each agent to generate its optimal supply/demand decision given the current prices. For example, if the agent is a manufacturer, it may need to solve several planning and scheduling problems just to construct its production possibilities set from which it has to choose the profit maximizing production plan. Furthermore, each agent has to go through this local deliberation at every iteration of the market protocol because prices change, and that affects what the optimal plan for each agent is.

5.6.3 Speculative Strategies in Equilibrium Markets

In general equilibrium markets, the agents are assumed to act *competitively*: they treat prices as exogenous. This means that each agent makes and reveals its demand (supply) decisions truthfully so as to maximize its utility (profit) given the market prices—assuming that it has no impact on those prices. The idea behind this *price-taking assumption* is that the market is so large that no single agent's actions affect the prices. However, this is paradoxical since the agents' declarations completely determine the prices. The price-taking assumption becomes valid as the number of agents approaches infinity: with infinitely many agents (of comparable size), each agent is best off acting competitively since it will not affect the prices.

However, in markets with a finite number of agents, an agent can act strategically, and potentially achieve higher utility by over/under representing [69], [38, pp. 220-223], [27]. In doing so, the agent has to speculate how its misrepresentation affects the market prices, which are simultaneously affected by how other agents respond to the prices which changed due to the first agent's strategic actions. In other words, general equilibria do not in general correspond to strictly rational, strategic equilibria of game theory.

This section is based on [69]. We analyze how much an agent can gain by speculation. Standard lies are also presented via which an agent can drive the market to a solution that maximizes the agent's gains from speculation, and looks like a general equilibrium to the other agents and the adjustor. These results are independent of the market algorithm as long as actual exchanges take place only after the market has reached (close to) an equilibrium.

Case A: Speculating Consumer

The goal of a self-interested consumer is to find the consumption bundle that maximizes its utility. To find the optimal bundle when acting in an equilibrium market, the consumer must speculate how other agents respond to prices. This is because its demand decisions affect the prices, which affect the demand and supply decisions of others, which again affect the prices that the consumer faces. Using the model of other agents, the consumer computes its optimal demand decisions. Note that other agents might also be speculating (in the same way or some other, suboptimal way). That is included in the agent's model of the other agents. A solution to the following maximization problem gives the highest utility that a speculating consumer s can possibly obtain.

$$
\begin{aligned}
&\max_{\mathbf{p}} \; u_s(\mathbf{x_s}(\mathbf{p})) \;\; \text{s.t.} \qquad (5.1)\\
&x_{sg}(\mathbf{p}) \geq 0 \;\; \text{(consumer does not produce)}\\
&x_{sg}(\mathbf{p}) = e_{sg} - \Big(\sum_{i \in \text{Consumers}-\{s\}} (x_{ig} - e_{ig}) - \sum_j y_{jg}\Big) \;\; \text{(supply meets demand)}\\
&\mathbf{p} \cdot (\mathbf{x_s} - \mathbf{e_s}) \leq \sum_j \theta_{sj} \; \mathbf{p} \cdot \mathbf{y_j}(\mathbf{p}) \;\; \text{(budget constraint)}
\end{aligned}
$$

Case B: Speculating Producer

The goal of a self-interested producer is to find the production vector that maximizes its profits. Again, this requires a model of how others react to prices because the producer's production decisions affect the prices, which affect the demand and supply decisions of others, which again affect the prices that the producer faces. A solution to the following maximization problem gives the highest profit that a speculating producer s can possibly obtain.

$$
\begin{aligned}
&\max_{\mathbf{p}} \; \mathbf{p} \cdot \mathbf{y}_s(\mathbf{p}) \;\; \text{s.t.} \qquad (5.2)\\
&\mathbf{y}_s(\mathbf{p}) \in Y_s \;\; \text{(feasible production plan)}\\
&y_{sg} = \sum_i (x_{ig} - e_{ig}) - \sum_{j \in \text{Producers}-\{s\}} y_{jg} \;\; \text{(supply meets demand)}
\end{aligned}
$$

The last equality turns into $\geq$ if free disposal for both inputs and outputs is possible.

The solution to the applicable optimization problem above (depending on whether the speculator is a producer or a consumer) is denoted $\mathbf{p}^{**}$. The equilibrium at p^{**} is not Pareto efficient in general. This does not violate Theorem. 5.10 because that result only applies to true general equilibria where agents act competitively.

Reaching Equilibrium under Speculation: Driving the Market

The discussion above focused on the prices that a speculating agent would like to drive the market to. However, there is a risk for the speculator that even though such an equilibrium exists, the market algorithm would not find it. A speculating agent's best strategy is to declare demand plans $\mathbf{x_s}(\mathbf{p})$ (or production plans $\mathbf{y_s}(\mathbf{p})$) such that the market clears at the desired prices $\mathbf{p}^{**}$ (an equilibrium exists), and the market process will find it. Formally, the market clears at $\mathbf{p}^{**}$ if for each good g,

$$x_{sg}(\mathbf{p}^{**}) = e_{sg} - (\sum_{i \in \text{Consumers}-\{s\}} (x_{ig}(\mathbf{p}^{**}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p}^{**})) \text{ if } s \text{ is a consumer, and}$$

$$y_{sg}(\mathbf{p}^{**}) = \sum_i (x_{ig}(\mathbf{p}^{**}) - e_{ig}) - \sum_{j \in \text{Producers}-\{s\}} y_{jg}(\mathbf{p}^{**}) \text{ if } s \text{ is a producer.}$$

What remains to be analyzed is whether the particular market algorithms finds the equilibrium even if the speculator acts strategically. Many standard market algorithms, e.g. price tâtonnement, Newtonian price tâtonnement, and WALRAS, are guaranteed to find the equilibrium if $\partial(\sum_i (x_{ig}(\mathbf{p}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p}))/\partial p_g < 0$ (society-wide demand decreases as price increases), and $\partial(\sum_i (x_{ig}(\mathbf{p}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p}))/\partial p_h \geq 0$, for goods $g \neq h$ (goods are gross substitutes). Let us assume that these two conditions would hold in the market if the speculator were not present. Now, if the speculating agent uses a strategy that satisfies

- Eq. 5.6.3, and
- $\partial(x_{sg}(\mathbf{p}) - e_{sg})/\partial p_g \leq 0$ if s is a consumer, and $\partial y_{sg}(\mathbf{p})/\partial p_g \geq 0$ if s is a producer, and
- for goods $g \neq h$, $\partial(x_{sg}(\mathbf{p}) - e_{sg})/\partial p_h \geq 0$ if s is a consumer, and $\partial y_{sg}(\mathbf{p})/\partial p_h \leq 0$ if s is a producer,

the market is guaranteed to converge to the unique equilibrium prices $\mathbf{p}^{**}$ that maximize the speculator's gain.

It turns out that simple generic strategies exist for the speculator that guarantee that these three conditions are met, i.e. that the speculator will be able to drive the market to an equilibrium where his maximal gain from speculation materializes [69]. For example, the following linear strategy is viable for a consumer:

$$\mathbf{x_s}(\mathbf{p}) = -\mathbf{e_s} + \mathbf{p}^{**} - \mathbf{p} - (\sum_{i \in \text{Consumers}-\{s\}} (x_{ig}(\mathbf{p}^{**}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p}^{**}))$$

and so is the constant strategy

$$\mathbf{x_s}(\mathbf{p}) = -\mathbf{e_s} - (\sum_{i \in \text{Consumers}-\{s\}} (x_{ig}(\mathbf{p}^{**}) - e_{ig}) - \sum_j y_{jg}(\mathbf{p}^{**})).$$

The corresponding strategies for a speculating producer are

$$\mathbf{y_s}(\mathbf{p}) = \mathbf{p} - \mathbf{p}^{**} + \sum_i (x_{ig}(\mathbf{p}^{**}) - e_{ig}) - \sum_{j \in \text{Producers}-\{s\}} y_{jg}(\mathbf{p}^{**})$$

and

$$\mathbf{y_s}(\mathbf{p}) = \sum_i (x_{ig}(\mathbf{p}^{**}) - e_{ig}) - \sum_{j \in \text{Producers}-\{s\}} y_{jg}(\mathbf{p}^{**}).$$

The last consideration is the speed of convergence to equilibrium. In any particular market setting, it may be that the market converges slower or faster when an agent acts strategically than when he acts competitively.

Strategic Behavior by Multiple Agents

In the analysis so far, one agent designed its speculative strategy while the others' strategies were fixed. However, the others would like to tailor their strategies to the specific strategy that the agent chooses. For this reason, we argue that strategic solution concepts from game theory should be used to design market protocols. The strategies are in Nash equilibrium if each agent's strategy is its best response to the others' strategies. This can be viewed as a necessary condition for system stability in settings where all agents act strategically.

A stronger condition is to require dominant strategy equilibrium, i.e. that each agent's strategy is optimal for that agent no matter what strategies others choose. Market protocols have been studied using dominant strategy equilibrium in [6]. The results are negative in the sense that the agents need to be given price ratios for trading in advance by the protocol designer, and the designer does not know the agents' preferences and capabilities. Therefore, not all beneficial trades can occur, and thus the solution is usually not Pareto efficient.

In sequential protocols, one can also strengthen the Nash equilibrium solution concept in multiple ways by requiring that the strategies stay in equilibrium at every step of the game [39, 35]. Unlike the market speculation analysis presented in this section so far, the Nash equilibrium outcome is specific to the market protocol. Important factors impacting the outcome are the order in which bids are submitted (see e.g. Stackleberg vs. Cournot models [39]), whether the bids are sealed or open [61], whether the protocol is iterative (the agents can change their excess demand between iterations) or not, whether the agents can decommit from their agreements by paying a penalty [67, 62], *etc.*

In some games, no Nash equilibrium exists for the market in pure (non-randomized) strategies. The following simple example illustrates this. Let there be two consumer agents, A and B, that engage in a market where they reveal their excess demand functions simultaneously and in a single round. Agent A can choose between two strategies (A1 and A2), and B can choose between B1 and B2. Provided that A knows that B will choose B1, A will choose A2, and A1 if B chooses B2. Provided that B knows that A will choose A2, B will choose B2, and B1 if A chooses A1. Now, from every possible pair of strategies, one agent would be motivated to deviate to another strategy, i.e. no Nash equilibrium exists. In general, existence and uniqueness of a general equilibrium (where agents act competitively) for a market does not imply existence and uniqueness of a Nash equilibrium.

Some market protocols may be difficult to analyze game theoretically. For example, in WALRAS, the agents might change their demand functions during the computation of the equilibrium. Then some agents may deliberately send false bids to generate more iterations of the market process in order to learn more about other agents' excess demand/supply functions. If many agents are involved in such probing, it seems that time becomes an important factor. Some agents might reveal progressively more of their competitive demands in order to speed up the convergence (as it might be urgent for them to get the resources traded), while others might extend the probing in order to maximize their benefit from the trade.[6]

While the game theoretic approach is clearly to be preferred (when it is viable) over the general equilibrium approach for designing interaction mechanisms for self-interested agents, the general equilibrium approach may still allow one to build reasonably nonmanipulable multiagent systems. For example, as the number of—comparably sized—agents increases, the gains from strategic acting decrease, approaching zero as the number of agents approaches infinity [55, 69]. Secondly, lack of knowledge about the others may make speculation unprofitable. If there is even minor uncertainty in the speculator's estimates about the others' strategies, the speculator's expected payoff may be significantly higher by acting competitively than by acting speculatively [69]. Finally, although beneficial lies are easy to compute once the others' strategies are known, it may be computationally complex to deduce the others' strategies even if the speculator knows the others' physical characteristics completely. For example, the speculator would need to solve a manufacturer's planning and scheduling problems in order to be able to deduce what the production possibilities sets of the manufacturer are, and what the manufacturer's (competitive) strategy will be. Sometimes the potential gains from speculation are not great enough to warrant such costly computation that may be required for speculation.

5.7 Contract Nets

General equilibrium market mechanisms use global prices, and—at least in the implementations up to now—use a single centralized mediator. The mediator might become a communication and computation bottleneck or a potential point of failure for the whole system. Also, in some settings the agents want to have direct control of who receives their sensitive information instead of posting the information to a mediator who controls its dissemination. Furthermore, sometimes it is unrealistic to assume that prices are global because there may be market frictions, costs to propagate information to all agents, *etc.* In such settings, a more distributed negotiation may be warranted.

The contract net protocol (see Chapter 2) was an early variant of such distributed

6. Some work has addressed non-competitive behavior in WALRAS [24], although there was only one speculating agent in the experiments, and this agent was limited to simple linear price prediction about how its actions affect the prices. Further analysis is required to determine whether its optimal strategy can be captured in this model. This need not be the case because the optimal strategy may involve some more "aggressive" behavior, e.g. the probing described above.

negotiation in a task allocation domain [77]. This section discusses some of the recent improvements to the contract net protocol. The new methods lead to better results, and they are viable among self-interested agents as well—unlike the original contract net protocol which was for cooperative agents only.

5.7.1 Task Allocation Negotiation

The capability of (re)allocating tasks among agents is a key feature in automated negotiation systems. In many domains, significant savings can be achieved by reallocating tasks among agents. Some tasks are inherently synergic, and should therefore be handled by the same agent. On the other hand, some tasks have negative interactions, in which case it is better to allocate them to different agents. Furthermore, different agents may have different resources which leads to different capabilities and costs for handling tasks. This section discusses task allocation among self-interested agents in the following model which captures the above considerations.

Definition 5.1 *[64]*
A *task allocation problem* is defined by a set of tasks T, a set of agents A, a cost function $c_i : 2^T \rightarrow \Re \cup \{\infty\}$ (which states the cost that agent i incurs by handling a particular subset of tasks), and the initial allocation of tasks among agents $\langle T_1^{init}, ..., T_{|A|}^{init} \rangle$, where $\bigcup_{i \in A} T_i^{init} = T$, and $T_i^{init} \cap T_j^{init} = \emptyset$ for all $i \neq j$. [7]

The original contract net and many of its later variants lacked a formal model for making bidding and awarding decisions. More recently, such a formal model was introduced which gives rise to a negotiation protocol that provably leads to desirable task allocations among agents [60, 62, 64]. In that model, contracting decisions are based on marginal cost calculations, i.e. that model invokes the concept of *individual rationality* on a per contract basis (which implies individual rationality of sequences of contracts). A contract is individually rational (IR) to an agent if that agent is better off with the contract than without it.

Specifically, a contractee q accepts a contract if it gets paid more than its marginal cost

$$MC^{add}(T^{contract} | T_q) = c_q(T^{contract} \cup T_q) - c_q(T_q)$$

of handling the tasks $T^{contract}$ of the contract. The marginal cost is dynamic in the sense that it depends on the other tasks T_q that the contractee already has.

Similarly, a contractor r is willing to allocate the tasks $T^{contract}$ from its current task set T_r to the contractee if it has to pay the contractee less than it saves by not

7. Although a static version of the problem is discussed, the contracting scheme works even if tasks and resources (resources affect the cost functions) are added and removed dynamically.

handling the tasks $T^{contract}$ itself:

$$MC^{remove}(T^{contract}|T_r) = c_r(T_r) - c_r(T_r - T^{contract}).$$

In the protocol, agents then suggest contracts to each other, and make their accepting/rejecting decisions based on these marginal cost calculations. An agent can take on both contractor and contractee roles. It can also recontract out tasks that it received earlier via another contract. The scheme does not assume that agents know the tasks or cost functions of others.

With this domain independent contracting scheme, the task allocation can only improve at each step. This corresponds to hill-climbing in the space of task allocations where the height-metric of the hill is social welfare $(-\sum_{i \in A} c_i(T_i))$. The fact that the contractor pays the contractee some amount between their marginal costs (e.g. half way between) causes the benefit from the improved task allocation to be divided so that no agent is worse off with a contract than without it.

The scheme is an *anytime algorithm*: contracting can be terminated at any time, and the worth (payments received from others minus cost of handling tasks) of each agent's solution increases monotonically. It follows that social welfare increases monotonically. Details on an asynchronous distributed implementation based on marginal costs can be found in [60, 62, 66].

Convergence to the Globally Optimal Task Allocation

In most contract net implementations, each contract regards only one task , i.e. one task is moved from one agent to another against a payment [77, 72, 22]. Such an *original (O) contract* can be understood as a particular search operator in the global hill-climbing contracting algorithm that is used for task reallocation. When the contracting protocol is equipped with O-contracts only, it may get stuck in a local optimum where no contract is individually rational but the task allocation is not globally optimal.

To solve this problem, several new contract types have recently been introduced: *cluster (C) contracts* where a set of tasks is atomically contracted from one agent to another, *swap (S) contracts* where a pair of agents swaps a pair of tasks, and *multiagent (M) contracts* where more than two agents are involved in an atomic exchange of tasks [64, 62, 60]. Each of the four contract types avoids some of the local optima that the other three do not:

Theorem 5.15
For each of the four contract types (O, C, S, and M), there exist task allocations where no IR contract with the other three contract types is possible, but an IR contract with the fourth type is [64].

Unfortunately, even if the contracting protocol is equipped with all four of the contract types, the globally optimal task allocation may not be reached via IR contracts—even if there were an oracle for choosing the sequence of contracts:

Theorem 5.16
There are instances of the task allocation problem where no IR sequence from the initial task allocation to the optimal one exists using O-, C-, S- and M-contracts [64].

Clearly, no subset of the contract types suffices either. Another problem is that without an oracle, contracting may get stuck in a local optimum even if some IR sequence exists because the agents may choose some other IR sequence.

To address this shortcoming, a new contract type, *OCSM-contract*, has been defined, which combines the characteristics of O-, C-, S-, and M-contracts into one contract type—where the ideas of the four earlier contract types can be applied simultaneously (atomically):

Definition 5.2 [64, 62]
An *OCSM-contract* is defined by a pair $\langle \mathbf{T}, \boldsymbol{\rho} \rangle$ of $|A| \times |A|$ matrices. An element $T_{i,j}$ is the set of tasks that agent i gives to agent j, and an element $\rho_{i,j}$ is the amount that i pays to j.

So OCSM contracts allow moving from a task allocation to any other task allocation with a single contract.

It could be shown that an IR sequence always exists from any task allocation to the optimal one if the contracting protocol incorporates OCSM-contracts. However, a stronger claim is now made. The following theorem states that OCSM-contracts are sufficient for reaching the global task allocation optimum in a finite number of contracts. The result holds for any sequence of IR OCSM-contracts, i.e. for any hill-climbing algorithm that uses OCSM-contracts: an oracle is not needed for choosing the sequence. This means that from the perspectives of social welfare maximization and of individual rationality, agents can accept IR contracts as they are offered. They need not wait for more profitable ones, and they need not worry that a current contract may make a more profitable future contract unprofitable. Neither do they need to accept contracts that are not IR in anticipation of future contracts that make the combination beneficial. Furthermore, these hill-climbing algorithms do not need to backtrack.

Theorem 5.17
Let $|A|$ and $|T|$ be finite. If the contracting protocol allows OCSM-contracts, any hill-climbing algorithm (i.e. any sequence of IR contracts) finds the globally optimal task allocation in a finite number of steps (without backtracking) [64, 62].

Proof With OCSM-contracts there are no local optima (that are not global optima) since the global optimum can be reached from any task allocation in a single contract. This last contract will be IR because moving to the optimum from some suboptimal allocation improves welfare, and this gain can be arbitrarily divided among the contract parties. Thus the algorithm will not run out of IR contracts before the optimum has been reached. With finite $|A|$ and $|T|$, there are only a finite number of task allocations. Since the algorithm hill-climbs, no task allocation will be repeated. Therefore, the optimum is reached in a finite number of contracts. ■

OCSM-contracts are also necessary: no weaker set of contract types suffices—even if there were an oracle to choose the order in which to apply them:

Theorem 5.18
If there is some OCSM-contract that the protocol does not allow, there are instances of the task allocation problem where no IR sequence exists from the initial allocation to the optimal one [64].

While OCSM-contracts are necessary in the general case, there may well be cost functions $c_i(\cdot)$ with special structure that guarantees that the global optimum is reached even with less powerful contract types.

Theorem 5.17 gives a powerful tool for problem instances where the number of possible task allocations is relatively small. On the other hand, for large problem instances, the number of contracts made before the optimal task allocation is reached may be impractically large—albeit finite. For example on a large-scale real-world distributed vehicle routing problem instance, the TRACONET [60] (marginal cost based) contracting system never reached even a local optimum even with just O-contracts—with multiple hours of negotiation on five Unix machines. Another problem is that although any OCSM-contract can be represented in $O(|A|^2 + |T|)$ space, the identification of welfare increasing contracts may be complex—especially in a distributed setting—because there are $\frac{v^2-v}{2} = \frac{|A|^{2|T|}-|A|^{|T|}}{2}$ possible OCSM-contracts, and the evaluation of just one contract requires each contract party to compute the cost of handling its current tasks and the tasks allocated to it via the contract. With such large problem instances, one cannot expect to reach the global optimum in practice. Instead, the contracting should occur as long as there is time, and then have a solution ready: the anytime character of this contracting scheme becomes more important.

Insincere Agents in Task Allocation

So far in this section on contracting it was assumed that agents act based on individual rationality. This differs from payoff maximizing agents of game theory. Such an agent may reject an IR contract e.g. if it believes that it could be better off by waiting for a more beneficial contract that cannot be accepted if the former contract is accepted (e.g. due to limited resources). Similarly, such an agent may accept a non-IR contract in anticipation of a synergic later contract that will make the combination beneficial. Furthermore, strategic agents can also speculate on the order of accepting contracts because different sequences of (IR) contracts may have different payoff to the agent. The IR approach is sometimes more practical than the full game theoretic analysis because each contract can be made by evaluating just a single contract (each contract party evaluating one new task set) instead of doing exponential lookahead into the future. The deviation from game theory comes at the cost of not being able to normatively guarantee that a self-interested agent is best off by following the strategy of accepting any IR contracts.

In this section on contracting it was also assumed that agents truthfully bid their marginal costs. However, an agent can benefit more in the contract payment by exaggerating its marginal cost. On the other hand, too much lying may cause some IR contracts to be perceived non-IR, and to be rejected. This issue of lying about the valuation was discussed in Section 5.4 on auctions.

Agents could also lie about what tasks they have. This type of lying has been thoroughly analyzed in a 2-agent task allocation setting [56]. The "Task Oriented Domains (TODs)" in that work are a strict subset of the task allocation problems presented here. Specifically, they assume that agents have symmetric cost functions ($c_i(T') = c_j(T')$) and that every agent is capable of handling all tasks of all agents, i.e. that the cost functions are always finite. The analysis is specific to a protocol where all agents reveal their costs for all possible task sets up front, the social welfare maximizing allocation is chosen, and then payoff is divided according to the Nash bargaining solution (Section 5.5.1). So their protocol is not an anytime algorithm: all task allocations have to be evaluated before any agreement is made.

How truthfully do agents reveal tasks to each other when each agent only knows about its own tasks? The domain class of TODs includes subclasses with very different properties regarding insincere task revelation. *Subadditive TODs* are TODs where $c_i(T' \cup T'') \leq c_i(T') + c_i(T'')$. A subclass of Subadditive TODs, *Concave TODs* are TODs where $c_i(T' \cup T''') - c_i(T') \geq c_i(T'' \cup T''') - c_i(T'')$. Finally, a subclass of Concave TODs, *Modular TODs* are TODs where $c_i(T' \cup T'') = c_i(T') + c_i(T'') - c_i(T' \cap T'')$.

Three alternative types of deals are analyzed. In *pure deals*, agents are deterministically allocated exhaustive, disjoint task sets. *Mixed deals* specify a probability distribution over such partitions. *All-or-nothing* deals are mixed deals where the alternatives only include partitions where one agent handles the tasks of all agents.

Three forms of lying are analyzed. First, an agent may *hide tasks* by not revealing them. Second, it may declare *phantom tasks* which do not exist and cannot be generated if another agent wants to see them. Finally, it may announce *decoy tasks*, which really did not exist, but which can be generated on demand. The forms of lying that are possible in different domain classes and with different deal types are summarized in Table 5.6. With more general TODs, many different lying methods can be profitable.

The analysis shows that even in the restricted settings, lying is often beneficial under the three variants of this protocol. Because these restricted domains are subproblems of more complex task allocation domains, the negative results carry over directly to the more complex settings. The results leave open the possibility that other protocols would demote lying more (while leading to Pareto efficient IR outcomes).

Deal type	General TOD			SubadditiveTOD			Concave TOD			Modular TOD		
	Hid	Pha	Dec	Hid	Pha	Dec	Hid	Pha	Dec	Hid	Pha	Dec
Pure	L	L	L	L	L	L	L	L	L	L		
Mixed	L		L	L		L	L			L		
All-or-nothing	-	-	-			L						

Table 5.6 Results on lying in task revelation. An 'L' indicates that lying of the specified type is profitable in some problem instances within the given domain class using the deal type. In general TODs using all-or-nothing deals, the negotiation set (set of individually rational Pareto efficient deals) may be empty.

5.7.2 Contingency Contracts and Leveled Commitment Contracts

In traditional multiagent negotiation protocols among self-interested agents, once a contract is made, it is binding, i.e. neither party can back out [56, 60, 64, 1, 14, 34, 69, 10]. Once an agent agrees to a contract, it has to follow through with it no matter how future events unravel. Although a contract may be profitable to an agent when viewed *ex ante*, it need not be profitable when viewed after some future events have occurred, i.e. *ex post*. Similarly, a contract may have too low expected payoff *ex ante*, but in some realizations of the future events, the same contract may be desirable when viewed *ex post*. Normal full commitment contracts are unable to efficiently take advantage of the possibilities that such—probabilistically known—future events provide.

On the other hand, many multiagent systems consisting of cooperative agents incorporate some form of decommitment possibility in order to allow the agents to accommodate new events. For example, in the original contract net protocol, the agent that had contracted out a task could send a termination message to cancel the contract even when the contractee had already partially fulfilled the contract. This was possible because the agents were not self-interested: the contractee did not mind losing part of its effort without a monetary compensation. Similarly, the role of decommitment possibilities among cooperative agents has been studied in meeting scheduling using a contracting approach [73]. Again, the agents did not require a monetary compensation for their efforts: an agent agreed to cancel a contract merely based on the fact that some other agent wanted to decommit. In such multiagent systems consisting of cooperative agents, each agent can be trusted to use such an externally imposed strategy even though using that strategy might not be in the agent's self-interest.

Some research in game theory has focused on utilizing the potential provided by probabilistically known future events by *contingency contracts* among self-interested agents. The obligations of the contract are made contingent on future events. There are games in which this method provides an expected payoff increase to both parties of the contract compared to any full commitment contract [53]. Also, some deals are enabled by contingency contracts in the sense that there is no full commitment contract that both agents prefer over their fallback positions, but there is a contingency contract that each agent prefers over its fallback.

There are at least three problems regarding the use of contingency contracts in automated negotiation among self-interested agents. Though useful in anticipating a small number of key events, contingency contracts get cumbersome as the number of relevant events to monitor from the future increases. In the limit, all domain events (changes in the domain problem, e.g. new tasks arriving or resources breaking down) and all negotiation events—contracts from other negotiations—can affect the value of the obligations of the original contract, and should therefore be conditioned on. Furthermore, these future events may not only affect the value of the original contract independently: the value of the original contract may depend on combinations of the future events [66, 60, 56]. Thus there is a potential combinatorial explosion of items to be conditioned on. Second, even if it were feasible to use such cumbersome contingency contracts among the computerized agents, it is often impossible to enumerate all possible relevant future events in advance. The third problem is that of verifying the unraveling of the events. Sometimes an event is only observable by one of the agents. This agent may have an incentive to lie to the other party of the contract about the event in case the event is associated with a disadvantageous contingency to the directly observing agent. Thus, to be viable, contingency contracts would require an event verification mechanism that is not manipulable and not prohibitively complicated or costly.

Leveled commitment contracts are another method for taking advantage of the possibilities provided by probabilistically known future events [67, 62, 2, 3]. Instead of conditioning the contract on future events, a mechanism is built into the contract that allows unilateral decommitting at any point in time. This is achieved by specifying in the contract decommitment penalties, one for each agent. If an agent wants to decommit—i.e. to be freed from the obligations of the contract—it can do so simply by paying the decommitment penalty to the other party. Such contracts are called leveled commitment contracts because the decommitment penalties can be used to choose a level of commitment. The method requires no explicit conditioning on future events: each agent can do its own conditioning dynamically. Therefore no event verification mechanism is required either.

While the leveled commitment contracting protocol has intuitive appeal and several practical advantages [62], it is not obvious that it is beneficial. First, the breacher's gain may be smaller than the breach victim's loss. Second, agents might decommit insincerely. A truthful agent would decommit whenever its best outside offer plus the decommitting penalty is better than the current contract. However, a rational self-interested agent would be more reluctant in decommitting. It can take into account the chance that the other party will decommit, in which case the former agent gets freed from the contract obligations, does not have to pay a decommitting penalty, and will collect a decommitting penalty from the other party. Due to such reluctant decommitting, contracts may end up being kept even though breaking them would be best from the social welfare perspective.

This issue was recently analyzed formally [67, 62]. A Nash equilibrium analysis was carried out where both contract parties' decommitting strategies (characterized by how good an agent's outside offer has to be to induce the agent to decommit) were best responses to each other. Both agents were decommitting insincerely but neither was motivated to change the extent of his lie given that the other did not change. It was shown that even under such insincere decommitting, the leveled commitment protocol outperforms the full commitment protocol. First, it enables contracts by making them IR in settings where no full commitment contract is IR (the reverse cannot happen). Second, leveled commitment contracts increase both contract parties' expected payoffs over any full commitment contracts.

5.8 Coalition Formation

In many domains, self-interested real world parties—e.g., companies or individual people—can save costs by coordinating their activities with other parties. For example when the planning activities are automated, it can be useful to automate the coordination activities as well. This can be done via a negotiating software agent representing each party.

The most general approach would be to state the coalition formation protocol as a normal form game (see e.g. Table 5.1) or an extensive form (i.e. sequential) game, and then analyze the Nash equilibria to see how the game would be played and how efficient the outcomes would be. To rigorously incorporate computation in the analysis, one could treat computational actions as part of each agent's strategy—just like physical actions.

However, the Nash equilibrium is often too weak because subgroups of agents can deviate in a coordinated manner. The *Strong Nash equilibrium* is a solution concept that guarantees more stability [5]. It requires that there is no subgroup that can deviate by changing strategies jointly in a manner that increases the payoff of all of its members given that nonmembers do not deviate from the original solution. The Strong Nash equilibrium is often too strong a solution concept because in many games no such equilibria exist.

The *Coalition-Proof Nash equilibrium* has been suggested as a partial remedy to the nonexistence problem of the Strong Nash equilibrium [7, 8]. It requires that there is no subgroup that can make a mutually beneficial deviation (keeping the strategies of nonmembers fixed) *in a way that the deviation itself is stable according to the same criterion*. A conceptual problem with this solution concept is that the deviation may be stable within the deviating group, but the solution concept ignores the possibility that some of the agents that deviated may prefer to deviate again with agents that did not originally deviate. Furthermore, even these kinds of solutions do not exist in all games.

Instead of the strategic approach that uses equilibrium analysis, coalition formation is often studied in a more abstract setting called a *characteristic function game*

(CFG). The rest of this section will be restricted to coalition formation in CFGs. In such games, the value of each coalition S is given by a characteristic function v_S. In other words, each coalition's value is independent of nonmembers' actions. However, in general the value of a coalition may depend on nonmembers' actions due to positive and negative externalities (interactions of the agents' solutions). Negative externalities between a coalition and nonmembers are often caused by shared resources. Once nonmembers are using the resource to a certain extent, not enough of that resource is available to agents in the coalition to carry out the planned solution at the minimum cost. Negative externalities can also be caused by conflicting goals. In satisfying their goals, nonmembers may actually move the world further from the coalition's goal state(s). Positive externalities are often caused by partially overlapping goals. In satisfying their goals, nonmembers may actually move the world closer to the coalition's goal state(s). From there the coalition can reach its goals less expensively than it could have without the actions of nonmembers. Settings with externalities between coalitions and nonmembers can be modeled e.g. as normal form games. CFGs are a strict subset of them. However, many real-world multiagent problems happen to be CFGs [68].

Coalition formation in CFGs includes three activities:

1. *Coalition structure generation*: formation of coalitions by the agents such that agents within each coalition coordinate their activities, but agents do not coordinate between coalitions. Precisely this means partitioning the set of agents into exhaustive and disjoint coalitions. This partition is called a *coalition structure (CS)*. For example, in a game with three agents, there are seven possible coalitions: {1}, {2}, {3}, {1,2}, {2,3}, {3,1}, {1,2,3} and five possible coalition structures: {{1}, {2}, {3}}, {{1}, {2,3}}, {{2}, {1,3}}, {{3}, {1,2}}, {{1,2,3}}.
2. *Solving the optimization problem* of each coalition. This means pooling the tasks and resources of the agents in the coalition, and solving this joint problem. The coalition's objective is to maximize monetary value: money received from outside the system for accomplishing tasks minus the cost of using resources. (In some problems, not all tasks have to be handled. This can be incorporated by associating a cost with each omitted task.)
3. *Dividing the value* of the generated solution among agents. This value may be negative because agents incur costs for using their resources.

These activities may be interleaved, and they are not independent. For example, the coalition that an agent wants to join depends on the portion of the value that the agent would be allocated in each potential coalition.

5.8.1 Coalition Formation Activity 1: Coalition Structure Generation

Classically, coalition formation research has mostly focused on the payoff division activity. Coalition structure generation and optimization within a coalition have not previously received as much attention. Research has focused [29, 86] on superadditive games, i.e. games where $v_{S \cup T} \geq v_S + v_T$ for all disjoint coalitions $S, T \subseteq A$. In such games, coalition structure generation is trivial because the agents are best off by forming the grand coalition where all agents operate together.

Superadditivity means that any pair of coalitions is best off by merging into one. Classically it is argued that almost all games are superadditive because, at worst, the agents in a composite coalition can use solutions that they had when they were in separate coalitions.

However, many games are not superadditive because there is some cost to the coalition formation process itself. For example, there might be coordination overhead like communication costs, or possible anti-trust penalties. Similarly, solving the optimization problem of a composite coalition may be more complex than solving the optimization problems of component coalitions. Therefore, under costly computation, component coalitions may be better off by not forming the composite coalition [68]. Also, if time is limited, the agents may not have time to carry out the communications and computations required to coordinate effectively within a composite coalition, so component coalitions may be more advantageous.

In games that are not superadditive, some coalitions are best off merging while others are not. In such settings, the social welfare maximizing coalition structure varies, and coalition structure generation becomes highly nontrivial. The goal is to maximize the social welfare of the agents A by finding a coalition structure

$$CS^* = \arg \max_{CS \in \text{partitions of A}} V(CS),$$

where

$$V(CS) = \sum_{S \in CS} v_S$$

The problem is that the number of coalition structures is large ($\Omega(|A|^{|A|/2})$), so not all coalition structures can be enumerated unless the number of agents is extremely small—in practice about 15 or fewer. Instead, one would like to search through a subset ($N \subset$ partitions of A) of coalition structures, and pick the best coalition structure seen so far:

$$CS^*_N = \arg \max_{CS \in N} V(CS)$$

Taking an outsider's view, the coalition structure generation process can be viewed as search in a *coalition structure graph*, Figure 5.3. Now, how should such a graph be searched if there are too many nodes to search it completely?

One desideratum is to be able to guarantee that this coalition structure is within a worst case bound from optimal, i.e. that

$$k \geq \frac{V(CS^*)}{V(CS^*_N)}$$

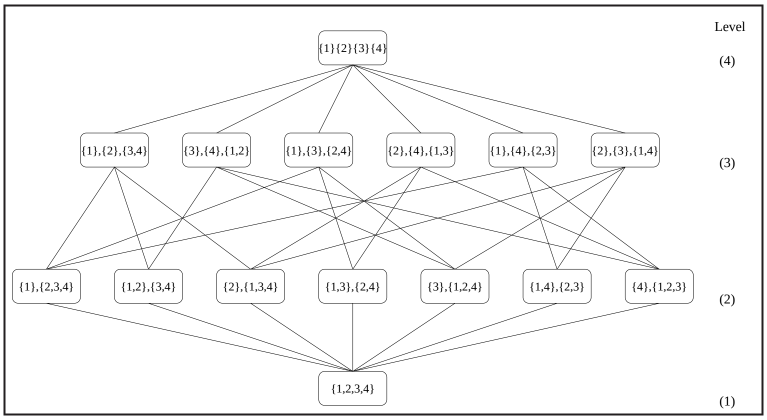

Figure 5.3 Coalition structure graph for a 4-agent game. The nodes represent coalition structures. The arcs represent mergers of two coalition when followed downward, and splits of a coalition into two coalitions when followed upward.

is finite, and as small as possible. Let us define n_{min} to be the smallest size of N that allows us to establish such a bound k.

Theorem 5.19 Minimal search to establish a bound
To bound k, it suffices to search the lowest two levels of the coalition structure graph (Figure 5.3). With this search, the bound $k = |A|$, this bound is tight, and the number of nodes searched is $n = 2^{|A|-1}$. No other search algorithm (than the one that searches the bottom two levels) can establish a bound k while searching only $n = 2^{|A|-1}$ nodes or fewer [65].

Interpreted positively, this means that—somewhat unintuitively—a worst case bound from optimum can be guaranteed without seeing all CSs. Moreover, as the number of agents grows, the fraction of coalition structures needed to be searched approaches zero, i.e. $\frac{n_{min}}{|\text{partitions of A}|} \rightarrow 0$ as $|A| \rightarrow \infty$. This is because the algorithm needs to see only $2^{|A|-1}$ coalition structures while the total number of coalition structures is $\Omega(|A|^{|A|/2})$.

Interpreted negatively, the theorem shows that exponentially many coalition structures have to be searched before a bound can be established. This may be prohibitively complex if the number of agents is large—albeit significantly better than attempting to enumerate all coalition structures. Viewed as a general impossibility result, the theorem states that no algorithm for coalition structure generation can establish a bound in general characteristic function games without trying at least $2^{|A|-1}$ coalition structures.[8] This sheds light on earlier algorithms. Specifi-

8. In restricted domains where the v_S values have special structure, it may be possible to establish a bound k with less search. Shehory and Kraus have analyzed coalition structure generation in one such setting [75]. However, the bound that they compute is not a bound

1. Search the **bottom** two levels of the coalition structure graph.
2. Continue with a breadth-first search from the **top** of the graph as long as there is time left, or until the entire graph has been searched.
3. Return the coalition structure that has the highest welfare among those seen so far.

Algorithm 5.3 COALITION-STRUCTURE-SEARCH-1 [Sandholm et al.]

cally, all prior coalition structure generation algorithms for general characteristic function games [76, 32]—which we know of—fail to establish such a bound. In other words, the coalition structure that they find may be arbitrarily far from optimal.

On the other hand, the following algorithm will establish a bound in the minimal amount of search, and then rapidly reduce the bound further if there is time for more search. If the domain happens to be superadditive, the algorithm finds the optimal coalition structure immediately.

The next theorem shows how this algorithm reduces the worst case bound, k, as more of the graph is searched. For convenience, we introduce the notation $h = \lfloor \frac{|A|-l}{2} \rfloor + 2$.

Theorem 5.20 Lowering the bound with further search
After searching level l with Algorithm 5.3, the bound k is $\lceil \frac{|A|}{h} \rceil$ if $|A| \equiv h-1 \pmod h$ and $|A| \equiv l \pmod 2$. Otherwise the bound is $\lfloor \frac{|A|}{h} \rfloor$. The bound is tight [65].

As was discussed earlier, before $2^{|A|-1}$ nodes have been searched, no bound can be established, and at $n = 2^{|A|-1}$ the bound $k = |A|$. The surprising fact is that by seeing just one additional node, i.e. the top node, the bound drops in half ($k = \frac{|A|}{2}$). Then, to drop k to about $\frac{|A|}{3}$, two more levels need to be searched. Roughly speaking, the divisor in the bound increases by one every time two more levels are searched. So, the anytime phase (step 2) of Algorithm 5.3 has the desirable feature that the bound drops rapidly early on, and there are overall diminishing returns to further search, Figure 5.4.

Comparison to Other Algorithms

All previous coalition structure generation algorithms for general CFGs [76, 32]—that we know of—fail to establish any worst case bound because they search fewer than 2^{a-1} coalition structures. Therefore, we compare Algorithm 5.3 to two other obvious candidates:

- **Merging algorithm**, i.e. breadth first search from the top of the coalition structure graph. This algorithm cannot establish any bound before it has searched the

from optimum, but from a benchmark (best that is achievable given a preset limit on the size of coalitions) which itself may be arbitrarily far from optimum.

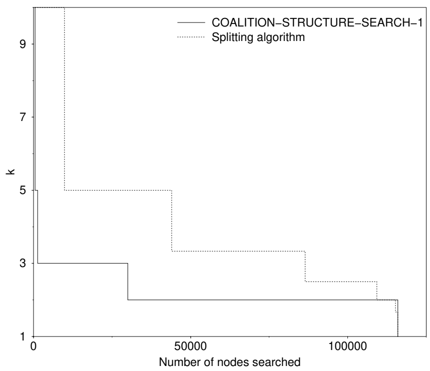

Figure 5.4 Ratio bound k as a function of search size in a 10-agent game.

entire graph. This is because, to establish a bound, the algorithm needs to see every coalition, and the grand coalition only occurs in the bottom node. Visiting the grand coalition as a special case would not help much since at least part of level 2 needs to be searched as well: coalitions of size $a-2$ only occur there.

- **Splitting algorithm**, i.e. breadth first search from the bottom of the graph. This is identical to Algorithm 5.3 up to the point where 2^{a-1} nodes have been searched, and a bound $k = a$ has been established. After that, the splitting algorithm reduces the bound much slower than Algorithm 5.3. This can be shown by constructing bad cases for the splitting algorithm: the worst case may be even worse. To construct a bad case, set $v_S = 1$ if $|S| = 1$, and $v_S = 0$ otherwise. Now, $CS^* = \{\{1\}, ..., \{a\}\}$, $V(CS^*) = a$, and $V(CS^*_N) = l - 1$, where l is the level that the algorithm has completed (because the number of unit coalitions in a CS never exceeds $l-1$). So, $\frac{V(CS^*)}{V(CS^*_N)} = \frac{a}{l-1}$,[9] Figure 5.4. In other words the divisor drops by one every time a level is searched. However, the levels that this algorithm searches first have many more nodes than the levels that Algorithm 5.3 searches first.

Variants of the Coalition Structure Generation Problem

One would like to construct an anytime algorithm that establishes a lower k for any amount of search n, compared to any other anytime algorithm. However, such an algorithm might not exist. It is conceivable that the search which establishes the minimal k while searching n' nodes ($n' > n$) does not include all nodes of the search which establishes the minimal k while searching n nodes. This hypothesis is supported by the fact that the curves in Figure 5.4 cross in the end. However, this is not conclusive because Algorithm 5.3 might not be the optimal anytime algorithm,

9. The only exception comes when the algorithm completes the last (top) level, i.e $l = a$. Then $\frac{V(CS^*)}{V(CS^*_N)} = 1$.

and because the bad cases for the splitting algorithm might not be the worst cases.

If it turns out that no anytime algorithm is best for all n, one could use information (e.g. exact, probabilistic, or bounds) about the termination time to construct a *design-to-time algorithm* which establishes the lowest possible k for the specified amount of search.

So far we have discussed algorithms that have an *off-line search control* policy, i.e. the nodes to be searched have to be selected without using information accrued from the search so far. With *on-line search control*, one could perhaps establish a lower k with less search because the search can be redirected based on the values observed in the nodes so far. With on-line search control, it might make a difference whether the search observes only values of coalition structures, $V(CS)$, or values of individual coalitions, v_S, in those structures. The latter gives more information.

None of these variants (anytime vs. design-to-time, and off-line vs. on-line search control) would affect the result that searching the bottom two levels of the coalition structure graph is the unique minimal way to establish a worst case bound, and that the bound is tight. However, the results on searching further might vary in these different settings.

Parallelizing Coalition Structure Search among Insincere Agents

This section discusses the parallelizing of coalition structure search—or any other search for that matter—across agents because the search can be done more efficiently in parallel, and the agents will share the burden of computation. Self-interested agents prefer greater personal payoffs, so they will search for coalition structures that maximize personal payoffs, ignoring k. Algorithm 5.4 can be used to motivate self-interested agents to exactly follow the socially desirable search. The randomizations in that algorithm can be done without a trusted third party by using a distributed nonmanipulable protocol for randomly permuting the agents, discussed at the end of Section 5.8.3.

5.8.2 Coalition Formation Activity 2: Optimization within a Coalition

Under unlimited and costless computation, each coalition would solve its optimization problem, which would define the value of that coalition. However, in practice, in many domains it is too complex from a combinatorial viewpoint to solve the problem exactly. Instead, only an approximate solution can be found. In such settings, self-interested agents would want to strike the optimal tradeoff between solution quality and the cost of the associated computation. This will affect the values of coalitions, which in turn will affect which coalition structure gives the highest welfare. This issue and several related questions are studied in detail in [68].

5.8.3 Coalition Formation Activity 3: Payoff Division

Payoff division strives to divide the value of the chosen coalition structure among agents in a fair and stable way so that the agents are motivated to stay with the

1. **Deciding what part of the coalition structure graph to search**. This can be decided in advance, or be dictated by a central authority or a randomly chosen agent, or be decided using some form of negotiation.

2. **Partitioning the search space among agents**. Each agent is assigned some part of the coalition structure graph to search. The enforcement mechanism in step 4 will motivate the agents to search exactly what they are assigned, no matter how unfairly the assignment is done. One way of achieving *ex ante* fairness is to randomly allocate the set search space portions to the agents. In this way, each agent searches equally on an expected value basis, although *ex post*, some may search more than others. Another option is to distribute the space equally among agents, or have some agents pay others to compensate for unequal amounts of search.

3. **Actual search**. Each agent searches its part of the search space, and tells the others which CS maximized $V(CS)$ in its search space.

4. **Enforcement**. Two agents, i and j, will be selected at random. Agent i will re-search the search space of j to verify that j has performed its search. Agent j gets caught of mis-searching (or misrepresenting) if i finds a better CS in j's space than j reported (or i sees that the CS that j reported does not belong to j's space at all). If j gets caught, it has to pay a penalty P. To motivate i to conduct this additional search, we make i the claimant of P. There is no pure strategy Nash equilibrium in this protocol. (If i searches and the penalty is high enough, then j is motivated to search sincerely. But then i is not motivated to search since it cannot receive P.) Instead, there will be a mixed strategy Nash equilibrium where i and j search truthfully with some probabilities. By increasing P, the probability that j searches can be made arbitrarily close to one. The probability that i searches approaches zero, which minimizes enforcement overhead.

5. **Additional search**. The previous steps can be repeated if more time to search remains. For example, the agents could first do step 1 of Algorithm 5.3. Then, they could repeatedly search more and more as time allows.

6. **Payoff division**. Many alternative methods for payoff division among agents could be used here. The only concern is that the division of $V(CS)$ may affect what CS an agent wants to report as a result of its search since different CSs may give the agent different payoffs—depending on the payoff division scheme. However, by making P high enough compared to $V(CS)$s, this consideration can be made negligible compared to the risk of getting caught.

Algorithm 5.4 Parallel search for self-interested agents [Sandholm et al.]

Repeat:
  Choose a coalition T
  For every agent $i \in T$, $x_i^{new} = x_i + \frac{v_T - \sum_{j \in T} x_j}{|T|}$, and $x_i^{new} = x_i$ for $i \notin T$
  Maintain feasibility: For every agent $i \in A$, $x_i = x_i^{new} - \frac{\sum_{j \in A} x_j^{new} - \sum_{S \in CS} v_S}{|A|}$

Algorithm 5.5 A transfer scheme for reaching the core [84].

coalition structure rather than move out of it. Several ways of dividing payoffs have been proposed in the literature [29]. This section discusses only two of them: the *core*, and the *Shapley value*.

Payoff Division According to the Core

The *core* of a CFG with transferable payoffs is a set of *payoff configurations* $(\vec{x}, CS)$, where each $\vec{x}$ is a vector of payoffs to the agents in such a manner that no subgroup is motivated to depart from the coalition structure CS:

Definition 5.3
Core $= \{(\vec{x}, CS) | \forall S \subset A, \sum_{i \in S} x_i \geq v_S$ and $\sum_{i \in A} x_i = \sum_{S \in CS} v_S\}$

Clearly, only coalition structures that maximize social welfare can be stable in the sense of the core because from any other coalition structure, the group of all agents would prefer to switch to a social welfare maximizing one.

The core is the strongest of the classical solution concepts in coalition formation. It is often too strong: in many cases it is empty [29, 78, 53, 86]. In such games there is no way to divide the social good so that the coalition structure becomes stable: any payoff configuration is prone to deviation by some subgroup of agents. The new solution that is acquired by the deviation is again prone to deviation and so on. There will be an infinite sequence of steps from one payoff configuration to another. To avoid this, explicit mechanisms such as limits on negotiation rounds, contract costs, or some social norms need to be in place in the negotiation setting.

Another problem is that the core may include multiple payoff vectors and the agents have to agree on one of them. An often used solution is to pick the *nucleolus* which, intuitively speaking, corresponds to a payoff vector that is in the center of the set of payoff vectors in the core [29, 78, 53].

A further problem with the core is that the constraints in the definition become numerous as the number of agents increases. This is due to the combinatorial subset operator in the definition. To reduce the associated cognitive burden of the agents that try to reach a payoff division in the core, Algorithm 5.5 can be used for payoff division. It stays within the given CS, and iteratively changes the payoff division. If the core is nonempty, Algorithm 5.5 will converge to a solution in the core starting from any initial payoff division. The choice of T can be made at random, or largest $v_T - \sum_{j \in T} x_j$ first. The latter variant tends to converge faster. There is no guarantee that a self-interested agent is motivated to follow the transfer scheme truthfully.

Payoff Division according to the Shapley Value

The *Shapley value* is another policy for dividing payoff in CFGs. It will first be characterized axiomatically. Agent i is called a *dummy* if $v_{S\cup\{i\}} - v_S = v_{\{i\}}$ for every coalition S that does not include i. Agents i and j are called *interchangeable* if $v_{(S\setminus\{i\})\cup\{j\}} = v_S$ for every coalition S that includes i but not j. The axioms of the Shapley value are:

- Symmetry: If i and j are interchangeable then $x_i = x_j$.
- Dummies: If i is a dummy then $x_i = v_{\{i\}}$.
- Additivity: For any two games v and w, x_i in $v + w$ equals x_i in v plus x_i in w, where $v + w$ is the game defined by $(v + w)_S = v_S + w_S$.

Theorem 5.21
The following is the only payoff division scheme that satisfies these three axioms [74]:

$$x_i = \sum_{S\subseteq A} \frac{(|A| - |S|)!(|S| - 1)!}{|A|!}[v_S - v_{S-\{i\}}]$$

This payoff is called the Shapley value of agent i. It can be interpreted as the marginal contribution of agent i to the coalition structure, averaged over all possible joining orders. The joining order matters since the perceived contribution of agent i varies based on which agents have joined before it.

The Shapley value always exists and is unique, while the core guarantees neither of these desirable properties. Like the core, the Shapley value is also Pareto efficient: the entire value of the coalition structure gets distributed among the agents. Like the core, the Shapley value guarantees that individual agents and the grand coalition are motivated to stay with the coalition structure. However, unlike the core, it does not guarantee that all subgroups of agents are better off in the coalition structure than by breaking off into a coalition of their own. This is not guaranteed by the Shapley value even in games where such a solution exists, i.e. the core is nonempty.

Another problem with the Shapley value is that the marginal contribution of each agent has to be computed over all joining orders, and there are $|A|!$ of them. One can guarantee each agent an *expected* payoff equal to its Shapley value by randomizing the joining order. This allows one to focus on one joining order only. A trusted third party needs to carry out the randomization since each agent has strong preferences over different joining orders because these orders lead to different payoffs for the agent.

The need for a trusted third party randomizer can be overcome via Zlotkin and Rosenschein's recent distributed nonmanipulable protocol for finding a randomized joining order [86]. First, every agent constructs a random permutation of the agents, encrypts it, and sends it to all others. Once an agent has received an encrypted permutation from every other agent, it broadcasts its key. These keys are then used to decrypt the permutations. The overall joining order is determined by sequentially permuting the results. For example, say that in a game of three agents, agent one's permutation is 3, 1, 2, agent two's permutation is 1, 3, 2, and agent three's permutation is 2, 3, 1. Applying agent one's permutation gives 3, 1, 2. Applying two's permutation to that gives 3, 2, 1. Applying three's permutation to that results in a joining order of 2, 1, 3.

An agent can do no better than randomize its permutation—assuming that at least one other agent randomizes its permutation (if the former agent knew exactly what all other agents' permutations are, that agent could tailor its permutation and do better). This assumes that the agent cannot change the interpretation of its permutation by changing its key after receiving the keys from others and decrypting their permutations. Changing the key so as to customize the interpretation of the permutation at that point may be difficult, and it can be made more difficult by enhancing Zlotkin and Rosenschein's protocol by requiring that every agent prefixes its permutation by a common string, e.g. "Hello world". Now manipulation would require the agent to construct a key that will change the interpretation of the agent's permutation in a desirable way while not changing the prefix.

5.9 Conclusions

Multiagent systems consisting of self-interested agents are becoming ubiquitous. Such agents cannot be coordinated by externally imposing the agent's strategies. Instead the interaction protocols have to be designed so that each agent really is motivated to follow the strategies that the protocol designer wants it to follow. This chapter discussed these issues under different types of protocols and different settings. Substantial knowledge exists of impossibility results and of constructive possibility demonstrations [39, 35]. This chapter only touched on some of it.

The implications of computational limitations were given special emphasis as a topic that has not traditionally received adequate attention. It is clear that such limitations have fundamental impact on what strategies agents want to use, and therefore also on what protocols are desirable, and what is (im)possible. This is one area where microeconomics and computer science fruitfully blend. Another area of substantial current and potential future cross-fertilization is the relaxation of the common knowledge assumption that underlies the Nash equilibrium solution concept and its refinements [16, 20].

In the future, systems will increasingly be designed, built, and operated in a distributed manner. A larger number of systems will be used by multiple real-world parties. The problem of coordinating these parties and avoiding manipulation cannot be tackled by technological or economic methods alone. Instead, the successful solutions are likely to emerge from a deep understanding and careful hybridization of both.

5.10 Exercises

1. *[Level 1]* The Gibbard-Satterthwaite theorem states that it is impossible to devise a truthpromoting voting mechanism for insincere agents. On the other hand, the Clarke tax mechanism is such a voting mechanism. Explain why this is not a contradiction.
2. *[Level 2]* Let there be a salesman located at each one of the following three coordinates: $(0,0)$, $(0,5)$, and $(5,0)$. Let there be a customer at each one of the following five locations: $(1,4)$, $(1.5,0)$, $(2,2)$, $(3,2)$, $(5,2)$. Each customer has to be assigned to exactly one salesman who will visit the customer. After visiting all of the customers assigned to him, the salesman has to return to his initial location. The domain cost that the salesman incurs from his travel is the Euclidean length of the trip. The tasks (locations of customers) are known to all salesmen. Write a program which uses the Clarke tax voting mechanism to solve this problem, i.e. tax is levied in a way that each salesman is motivated to reveal his preferences (over task allocations among agents) truthfully.
   (a) How many possible task allocations are there?
   (b) List each agent's preference (numeric value) for each of these.
   (c) Which task allocation will be chosen?
   (d) List the route of each salesman.
   (e) How much domain (travel) cost does each salesman incur?
   (f) How much tax does each agent pay/receive?
   (g) What is the budget balance/deficit?
   (h) Demonstrate a way—if one exists—how some agents can beneficially collude by revealing their preferences untruthfully. How would this changes answers (c)-(g)?
3. *[Level 3]* Program an example general equilibrium market economy that satisfies the gross substitutes property. Compare the convergence of the price tâtonnement algorithm and the Newtonian price tâtonnement algorithm. Then, experiment with how much one agent can gain by acting strategically (speculatively) instead of acting competitively as a price taker.
4. *[Level 1]* Discuss how the revelation principle relates the Vickrey auction to the English auction. How does this relate to the "agent" that bids on the human's behalf at http://www.webauction.com?
5. *[Level 2]* Prove Theorem 5.5.
6. *[Level 2]* Show an example where an agent is best off bidding insincerely if the second-price auction is implemented as open-cry instead of sealed-bid.
7. *[Level 2]* Construct an example where O-contracts lead to a local optimum (when agents use per contract individual rationality as their decision criterion)

that is not globally optimal.

8. *[Level 4]* How should agents look ahead in contracting and in auctions of interrelated items? The extremes are no lookahead (IR contracts), and full (game theoretic) lookahead. In practice something in between these extremes is likely to be best since there is a tradeoff between the computational cost of looking ahead and the domain cost savings that lookahead may provide.
9. *[Level 3]* Construct a 2-agent task allocation problem instance where an agent benefits from a decoy lie. The protocol should make every agent reveal its tasks at once (cost functions over tasks may be assumed common knowledge), should use pure deals, and should divide payoffs according to the Nash bargaining solution.
10. *[Level 1]* Program the transfer scheme for the core. Run it on an example problem instance where the core is nonempty. What happens when you run it on a problem instance where the core is empty?
11. *[Level 2]* This question is based on [68]. Let there be three agents. Let the unit cost of computation be \$200 (e.g. for a day of supercomputer time). Let the algorithms' performance profiles be:

$$\begin{aligned}
c_{\{1\}}(t_{CPU}) &= \$100 \cdot e^{-t_{CPU}} + \$300 \\
c_{\{2\}}(t_{CPU}) &= \$80 \cdot e^{-t_{CPU}} + \$200 \\
c_{\{3\}}(t_{CPU}) &= \$65 \cdot e^{-t_{CPU}} + \$200 \\
c_{\{1,2\}}(t_{CPU}) &= \$240 \cdot e^{-t_{CPU}} + \$400 \\
c_{\{2,3\}}(t_{CPU}) &= \$175 \cdot e^{-t_{CPU}} + \$419 \\
c_{\{1,3\}}(t_{CPU}) &= \$190 \cdot e^{-t_{CPU}} + \$400 \\
c_{\{1,2,3\}}(t_{CPU}) &= \$500 \cdot e^{-t_{CPU}} + \$595
\end{aligned}$$

Note that different coalitions might use different amounts of computation. What is the social welfare maximizing coalition structure? Is it stable according to the core (justify your answer)? How would these answers change if computation were free?

5.11 References

1. Martin R Andersson and Tuomas W Sandholm. Contract types for satisficing task allocation: II experimental results. In *AAAI Spring Symposium Series: Satisficing Models*, pages 1–7, Stanford University, CA, March 1998.
2. Martin R Andersson and Tuomas W Sandholm. Leveled commitment contracting among myopic individually rational agents. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS)*, Paris, France, July 1998.
3. Martin R Andersson and Tuomas W Sandholm. Leveled commitment contracts

with myopic and strategic agents. In *Proceedings of the National Conference on Artificial Intelligence*, Madison, WI, July 1998.

4. Kenneth Arrow. *Social choice and individual values.* New Haven: Cowles Foundation, 2nd edition, 1963. 1st edition 1951.
5. R Aumann. Acceptable points in general cooperative n-person games. volume IV of *Contributions to the Theory of Games*. Princeton University Press, 1959.
6. Salvador Barbera and Matthew O Jackson. Strategy-proof exchange. *Econometrica*, 63(1):51–87, 1995.
7. B Douglas Bernheim, Bezalel Peleg, and Michael D Whinston. Coalition-proof Nash equilibria: I concepts. *Journal of Economic Theory*, 42(1):1–12, June 1987.
8. B Douglas Bernheim and Michael D Whinston. Coalition-proof Nash equilibria: II applications. *Journal of Economic Theory*, 42(1):13–29, June 1987.
9. Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.
10. John Q. Cheng and Michael P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 1997.
11. E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.
12. E Durfee, V Lesser, and D Corkill. Cooperative distributed problem solving. In A Barr, P Cohen, and E Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume IV, pages 83–147. Addison Wesley, 1989.
13. Eithan Ephrati. A non-manipulable meeting scheduling system. In *Proc. 13th International Distributed Artificial Intelligence Workshop*, Lake Quinalt, Washington, July 1994. AAAI Press Technical Report WS-94-02.
14. Eithan Ephrati and Jeffrey S Rosenschein. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 173–178, Anaheim, CA, 1991.
15. Eithan Ephrati and Jeffrey S Rosenschein. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 423–429, Chambery, France, 1993.
16. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge.* MIT Press, 1995.
17. Drew Fudenberg and Jean Tirole. *Game Theory.* MIT Press, 1991.
18. A Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.
19. Natalie S. Glance and Tad Hogg. Dilemmas in computational societies. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 117–124, San Francisco, CA, 1995.
20. P. J. Gmytrasiewicz and E. H. Durfee. A rigorous, operational formalization of recursive modeling. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 125–132, San Francisco, CA, June 1995.
21. Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.
22. Cheng Gu and Toru Ishida. A quantitative analysis of the contract net protocol. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, page 449, San Francisco, CA, June 1995. In the poster collection.
23. Tad Hogg. Social dilemmas in computational ecosystems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 711–716,

Montreal, Canada, 1995.

24. J. Hu and M. P. Wellman. Self-fulfilling bias in multiagent learning. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS)*, pages 118–125, Keihanna Plaza, Kyoto, Japan, December 1996.
25. Bernardo Huberman and Scott H Clearwater. A multi-agent system for controlling building environments. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 171–176, San Francisco, CA, June 1995.
26. Bernardo A. Huberman and Tad Hogg. The behavior of computational ecologies. In B. A. Huberman, editor, *The Ecology of Computation*. North-Holland, 1988.
27. L. Hurwicz. On informationally decentralized systems. In C.B McGuire and R. Radner, editors, *Decision and Organization*, chapter 14, pages 297–336. University of Minnesota Press, 1986. Second edition.
28. Matthew Jackson and James Peck. Speculation and price fluctuations with private, extrinsic signals. *Journal of Economic Theory*, 55:274–295, 1991.
29. James P Kahan and Amnon Rapoport. *Theories of Coalition Formation*. Lawrence Erlbaum Associates Publishers, 1984.
30. Ehud Kalai. Solutions to the bargaining problem. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, chapter 3, pages 77–106. Cambridge University Press, 1985.
31. Ravi Kalakota and Andrew B. Whinston. *Frontiers of Electronic Commerce*. Addison-Wesley Publishing Company, Inc, 1996.
32. Steven Ketchpel. Forming coalitions in the face of uncertain rewards. In *Proceedings of the National Conference on Artificial Intelligence*, pages 414–419, Seattle, WA, July 1994.
33. Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, March 1990.
34. Sarit Kraus, Jonathan Wilkenfeld, and Gilad Zlotkin. Multiagent negotiation under time constraints. *Artificial Intelligence*, 75:297–345, 1995.
35. David M Kreps. *A Course in Microeconomic Theory*. Princeton University Press, 1990.
36. Jacques Lenting and Peter Braspenning. An all-pay auction approach to reallocation. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 259–263, August 1994.
37. Jeffrey K. MacKie-Mason and Hal R. Varian. Pricing the Internet. In *Proceedings of the Public Access to the Internet Conference*. JFK School of Government, May 1993.
38. E. Malinvaud. *Lectures on Microeconomic Theory*. North-Holland, 1985.
39. Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
40. Eric S Maskin. The theory of implementation in Nash equilibrium: a survey. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, chapter 6, pages 173–204. Cambridge University Press, 1985.
41. R Preston McAfee and John McMillan. Analyzing the airwaves auction. *Journal of Economic Perspectives*, 10(1):159–175, 1996.
42. Paul R Milgrom. The economics of competitive bidding: a selective survey. In

Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, chapter 9, pages 261–292. Cambridge University Press, 1985.

43. Mark S. Miller and K. Eric Drexler. Comparative ecology: A computational perspective. In B. A. Huberman, editor, *The Ecology of Computation*. North-Holland, 1988.

44. Tracy Mullen and Michael P Wellman. A simple computational market for network information services. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 283–289, San Francisco, CA, June 1995.

45. Eithan Muller and Mark A Satterwaithe. Strategy-proofness: the existence of dominant strategy mechanisms. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, chapter 5, pages 131–172. Cambridge University Press, 1985.

46. Roger B Myerson. Bayesian equilibrium and incentive compatibility: an introduction. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, chapter 8, pages 229–260. Cambridge University Press, 1985.

47. John Nash. The bargaining problem. *Econometrica*, 18:155–162, 1950.

48. John Nash. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36:48–49, 1950.

49. Peter C Ordeshook. *Game theory and political theory*. Cambridge University Press, 1986.

50. Martin J Osborne and Ariel Rubinstein. *Bargaining and Markets*. Academic Press, Inc., 1990.

51. Martin J Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

52. Andrew Postlewaite. Implementation via Nash equilibria in economic environments. In Leonid Hurwicz, David Schmeidler, and Hugo Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, chapter 7, pages 205–228. Cambridge University Press, 1985.

53. H. Raiffa. *The Art and Science of Negotiation*. Harvard Univ. Press, Cambridge, Mass., 1982.

54. Eric Rasmusen. *Games and Information*. Basil Blackwell, 1989.

55. Donald John Roberts and Andrew Postlewaite. The incentives for price-taking behavior in large exchange economies. *Econometrica*, 44(1):115–127, 1976.

56. Jeffrey S Rosenschein and Gilad Zlotkin. *Rules of Encounter*. MIT Press, 1994.

57. Michael H Rothkopf and Ronald M Harstad. Two models of bid-taker cheating in Vickrey auctions. *Journal of Business*, 68(2):257–267, 1995.

58. Michael H Rothkopf, Thomas J Teisberg, and Edward P Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98(1):94–109, 1990.

59. Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50, 1982.

60. Tuomas W Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 256–262, Washington, D.C., July 1993.

61. Tuomas W Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Proceedings of the Second International Conference on*

Multi-Agent Systems (ICMAS), pages 299–306, Keihanna Plaza, Kyoto, Japan, December 1996.

62. Tuomas W Sandholm. *Negotiation among Self-Interested Computationally Limited Agents*. PhD thesis, University of Massachusetts, Amherst, 1996. Available at http://www.cs.wustl.edu/~sandholm/dissertation.ps.
63. Tuomas W Sandholm. Unenforced E-commerce transactions. *IEEE Internet Computing*, 1(6):47–54, Nov–Dec 1997. Special issue on Electronic Commerce.
64. Tuomas W. Sandholm. Contract types for satisficing task allocation: I theoretical results. In *AAAI Spring Symposium Series: Satisficing Models*, pages 68–75, Stanford University, CA, March 1998.
65. Tuomas W Sandholm, Kate S Larson, Martin R Andersson, Onn Shehory, and Fernando Tohmé. Anytime coalition structure generation with worst case guarantees. In *Proceedings of the National Conference on Artificial Intelligence*, Madison, WI, July 1998.
66. Tuomas W Sandholm and Victor R Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 328–335, San Francisco, CA, June 1995. Reprinted in *Readings in Agents*, Huhns and Singh, eds., pp. 66–73, 1997.
67. Tuomas W Sandholm and Victor R Lesser. Advantages of a leveled commitment contracting protocol. In *Proceedings of the National Conference on Artificial Intelligence*, pages 126–133, Portland, OR, August 1996.
68. Tuomas W Sandholm and Victor R Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997. Special issue on Economic Principles of Multiagent Systems.
69. Tuomas W Sandholm and Fredrik Ygge. On the gains and losses of speculation in equilibrium markets. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 632–638, Nagoya, Japan, August 1997.
70. M A Satterthwaite. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.
71. R Selten. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit. *Zeitschrift für die gesamte Staatswissenschaft*, 12:301–324, 1965.
72. Sandip Sen. *Tradeoffs in Contract-Based Distributed Scheduling*. PhD thesis, Univ. of Michigan, 1993.
73. Sandip Sen and Edmund Durfee. The role of commitment in cooperative negotiation. *International Journal on Intelligent Cooperative Information Systems*, 3(1):67–81, 1994.
74. Lloyd S Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume 2 of *Annals of Mathematics Studies, 28*, pages 307–317. Princeton University Press, 1953.
75. Onn Shehory and Sarit Kraus. Task allocation via coalition formation among autonomous agents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 655–661, Montreal, Canada, August 1995.
76. Onn Shehory and Sarit Kraus. A kernel-oriented model for coalition-formation in general environments: Implemetation and results. In *Proceedings of the National Conference on Artificial Intelligence*, pages 134–140, Portland, OR, August 1996.

77. Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

78. Wim J van der Linden and Albert Verbeek. Coalition formation: A game-theoretic approach. In Henk A M Wilke, editor, *Coalition Formation*, volume 24 of *Advances in Psychology*. North Holland, 1985.

79. Hal R Varian. *Microeconomic analysis*. New York: W. W. Norton, 1992.

80. W Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

81. Carl A Waldspurger, Tad Hogg, Bernardo Huberman, Jeffrey O Kephart, and W Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

82. Michael Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

83. Michael Wellman. A computational market model for distributed configuration design. In *Proc. 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 401–407, Seattle, WA, July 1994.

84. L. S. Wu. A dynamic theory for the class of games with nonempty cores. *SIAM Journal of Applied Mathematics*, 32:328–338, 1977.

85. F. Ygge and J. M. Akkermans. Power load management as a computational market. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS)*, pages 393–400, Keihanna Plaza, Kyoto, Japan, December 1996.

86. Gilad Zlotkin and Jeffrey S Rosenschein. Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 432–437, Seattle, WA, July 1994.