Deliberation in Equilibrium: Bargaining in Computationally Complex Problems

Kate Larson and Tuomas Sandholm
Department of Computer Science
Washington University
St. Louis, MO 63130-4899
{ksl2, sandholm}@cs.wustl.edu

Abstract

We develop a normative theory of interaction—negotiation in particular—among self-interested computationally limited agents where computational actions are game-theoretically treated as part of an agent's strategy. We focus on a 2-agent setting where each agent has an intractable individual problem, and there is a potential gain from pooling the problems, giving rise to an intractable joint problem. At any time, an agent can compute to improve its solution to its problem, its opponent's problem, or the joint problem. At a deadline the agents then decide whether to implement the joint solution, and if so, how to divide its value (or cost). We present a fully normative model for controlling anytime algorithms where each agent has statistical performance profiles which are optimally conditioned on the problem instance as well as on the path of results of the algorithm run so far. Using this model, we analyze the perfect Bayesian equilibria of the games which differ based on whether the performance profiles are deterministic or stochastic, whether the deadline is known or not, and whether the proposer is known in advance. Finally, we present algorithms for finding the equilibria.

Introduction

Systems, especially on the Internet, are increasingly being used by multiple parties—or software agents that represent them—with their own preferences. This invalidates the traditional assumption that a central designer controls the behavior of all system components. The system designer can only control the *mechanism* (rules of the game), while each agent chooses its own *strategy*. The economic efficiency that a system yields depends on the agents' strategies. So, to develop a system that leads to desirable outcomes, the designer has to make sure that each agent is incented to behave in the desired way. This can be achieved by analyzing the game using the *Nash equilibrium* solution concept from game theory (or its refinements): no agent is motivated to deviate from its strategy given that the others do not deviate (Kreps 1990).

However, the equilibrium for rational agents does not generally remain an equilibrium for computationally limited agents.[1] This leaves a potentially hazardous gap in game theory as well as automated negotiation because computationally limited agents are not incented to behave in the desired way. This paper presents a framework and first steps toward filling that gap.

In this paper we begin to develop a theory of interaction—negotiation in particular—where computation actions are treated as part of an agent's strategy. We study a 2-agent bargaining setting where at any time, the agent can compute to improve its solution to its own problem, its solution to the opponent's problem, or its solution to the joint problem where the tasks and resources of the two agents are pooled. The bargaining occurs over whether or not to use a solution to the joint problem, and how to divide the associated value (or cost). This is the first piece of research that seriously treats computational actions game-theoretically.

Early on, it was recognized that humans have bounded rationality, e.g., due to cognitive limitations, so they do not act rationally as economic theory would predict (Simon 1955). He noted that there was a difference in the ways firms *do* behave as opposed to how they *should* rationally behave.

Since then, considerable work has focused on developing *normative* models that prescribe how a computationally limited agent *should* behave. Most of those methods resort to simplifying assumptions such as myopic deliberation control (Russell & Wefald 1991; Baum & Smith 1997), conditioning the deliberation control on hand-picked features, assuming that an anytime algorithm's future performance does not depend on the run on that instance so far (Horvitz 1987; Boddy & Dean 1994; Zilberstein & Russell 1996; Zilberstein, Charpillet, & Chassaing 1999; Horvitz 1997) or that performance is conditioned on quality so far but not the path (Hansen & Zilberstein 1996), resorting to asymptotic notions of bounded optimality (Russell & Subramanian 1995), or focusing on decision problems only (Sandholm & Lesser 1994).

While such simplifications can be acceptable in single-agent settings as long as the agent performs reasonably well, any deviation from full normativity can be catastrophic in games. If the designer cannot guarantee that the strategy (including deliberation actions) is the best strategy that an agent can use, there is a risk that an agent is incented to use some other strategy. Even if that strategy happens to be "close" to the desired one, the social outcome may be far from desirable. Therefore, this paper introduces a fully normative deliberation control method. Each agent uses all the information it has available to control its computation,

[1]In the relatively rare settings where the incentives can be designed so that each agent is motivated to use the desired strategy *independent of what others do* (dominant strategy equilibrium), a rational agent is best off maintaining its strategy even if some other agents are unable to act rationally, e.g. due to computational limitations.

including conditioning on the problem instance and the path of solutions found on the run so far.

Game theorists have also realized the significance of computational limitations, but the models that address this issue have mostly analyzed how complex it is to compute the rational strategies (rather than the computation impacting the strategies) (Koller, Megiddo, & Stengel 1996), or memory limitations in keeping track of history in repeated games (Rubinstein 1998), or limited uniform-depth lookahead capability in repeated games (Jehiel 1995), or showing that allowing the choice between taking one computation action or not undoes the dominant strategy property in a Vickrey auction (Sandholm 1996). On the other hand, in this paper, the limited rationality stems from the complexity of each agent's optimization problem (each agent has a computer of finite speed, some anytime algorithm which might not be perfect, and finite time), a setting which is ubiquitous in practice.[2]

In the next section we present a quantitative model for controlling computation where each agent has statistical performance profiles of its anytime algorithm, which are optimally conditioned on the problem instance as well as on the path of results of the algorithm run so far. We also present the bargaining settings we are studying. We then proceed to analyze noncooperative equilibria and present algorithms that agents can use to determine how to direct their computation in equilibrium and how to bargain after the deliberation. In other words, these algorithms determine each agent's best-response deliberation strategy and bargaining strategy. The paper ends with a discussion of future research directions.

An Example Application

To make the presentation more concrete, we now discuss an example domain where our methods are needed. Consider a distributed vehicle routing problem (Sandholm & Lesser 1997) with two geographically dispersed dispatch centers that are self-interested companies. Each center is responsible for certain tasks (deliveries) and has a certain set of resources (vehicles) to take care of them. So each agent—representing a dispatch center—has its own vehicles and delivery tasks. Each agent's *individual problem* is to minimize transportation costs (driven mileage) while still making all of its deliveries. This problem is $\mathcal{NP}$-complete.

There is a potential for savings in driven mileage by pooling the agents' tasks and resources—e.g., because one agent may be able to handle some of the other's tasks with less driving due to adjacency. The objective in this *joint problem* is to again minimize driven mileage. This problem is again $\mathcal{NP}$-complete.

[2]The same *source* of complexity has been addressed (Sandholm & Lesser 1997), but that paper only studied outcomes, not the process or the agents' strategies. It was also assumed that the algorithm's performance is deterministically known in advance. Finally, the agents had costly but unlimited computation, while in this paper the agents have free but limited computation.

The General Setting

The distributed vehicle routing problem is only one example problem where the methods of this paper are needed. In general, they are needed in any 2-agent setting where each agent has an intractable *individual problem*, and there is a potential savings from pooling the problems, giving rise to an intractable *joint problem*. We also assume that the value of any solution to an agent's individual problem is not affected by what solution the other agent uses to its individual problem.

Applications with these characteristics are ubiquitous, including transportation as discussed above, manufacturing (where two potentially interacting companies need to construct their manufacturing plans and schedules), electric power negotiation between a custom provider and an industrial consumer (where the participants need to construct their production and consumption schedules), to name just a few.

In order to determine the gain generated by pooling instead of each agent operating individually, agents need to compute solutions to both agent's individual problems as well as to the joint problem. Say that the agents have anytime algorithms that can be used to solve (vehicle routing) problems so that some feasible solution is available whenever the algorithm is terminated, and the solution improves as more computation time is allocated to the algorithm.

By computing on the joint problem, an agent reduces the amount of time it has for computing on its individual problem. This may increase the joint value to the agents (reduce the sum of the agents' costs), but makes this agent's fallback position worse when it comes to bargaining over how the joint value should be divided between the two agents. Also, if one agent is computing on the joint problem, would it not be better for the other agent to compute on something different so as not to waste computation? In this paper we present a model where each agent strategically decides on how to use its limited computation in order to maximize its own expected payoff in such settings.

The Model

Let there be two agents, 1 and 2, each with its own *individual problem*. They also have the possibility to pool, giving rise to a *joint problem*. We assume that time is discretized into T units and each computational step takes one time unit.

Normative Control of Deliberation

Each agent has an anytime algorithm that has a feasible solution available whenever it is terminated, and improves the solution as more computation time is allocated to the problem. Let $v^1(t)$ be the value of the solution to agent 1's individual problem after computing on it for t time steps. Similarly, $v^2(t)$ is the value of the solution to agent 2's individual problem after computing on it for t time steps. Finally, $v^{1\cup 2}(t)$ is the value of the solution to the joint problem after computing on it for t time steps.

The agents have statistical performance profiles that describe how their anytime algorithms increase v^1, v^2, and $v^{1\cup 2}$ as a function of the allocated computation time. As will be discussed later, each agent uses this information to decide how to allocate its computation at every step of the game.

We index the problem (agent 1's, agent 2's, and the joint) by z, $z \in \{1, 2, 1 \cup 2\}$. For each z there is a performance profile tree $\mathcal{T}^z$, representing the fact that an agent can condition its algorithm's performance profile on the problem instance. Figure 1 exemplifies one such tree. Each depth of the tree corresponds to an amount of time t spent on running the algorithm on that problem. Each node at depth t of the tree represents a possible solution quality (value), v^z, that is obtained by running the algorithm for t time steps on that problem. There may be several nodes at a depth since the algorithm may reach different solution qualities for a given amount of computation depending on the problem instance (and if it is a stochastic algorithm, also on random numbers). We assume that the solution quality in the performance profile tree, $\mathcal{T}^1$, of agent 1's individual problem is discretized into a finite number of levels. Similarly, the solution quality in $\mathcal{T}^2$ is discretized into a finite number of levels, as is the solution quality in $\mathcal{T}^{1\cup 2}$.

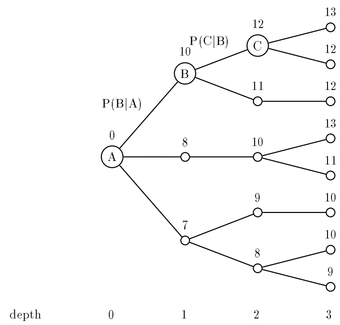

Figure 1: *A performance profile tree.*

Each edge in the tree is associated with the probability that the child is reached in the next computation step given that the parent has been reached. This allows one to compute the probability of reaching any particular future node in the tree given any node, by multiplying the probabilities on the path between the nodes. If there is no path, the probability is 0.

The tree is constructed by collecting statistical data from previous runs of the algorithm on different problem instances.[3] Each run is represented as a path in the tree. As a run proceeds along a path in the tree, the frequency of each edge of that path is incremented, and the frequencies at the nodes on the path are normalized to obtain probabilities. If the run leads to a value for which there is no node in the tree, the node is generated and an edge is inserted from the previous node to it.

Definition 1 *The* **state of deliberation** *of agent 1 at time step t is*

$$\theta_1^t = \langle n_1^1, n_1^2, n_1^{1\cup 2} \rangle$$

where n_1^1, n_1^2, and $n_1^{1\cup 2}$ are the nodes where agent 1 is currently in each of the three performance profile trees. The state of deliberation for agent 2 is defined analogously.

We denote by $\mathrm{time}(n)$ the depth of node n in the performance profile tree. In other words, $\mathrm{time}(n)$ is the number of computation steps used to reach node n. So, $\mathrm{time}(n_1^1) + \mathrm{time}(n_1^2) + \mathrm{time}(n_1^{1\cup 2}) = t$. We denote by $V(n)$ the value of node n.

In practice it is unlikely that an agent knows the solution quality for every time allocation without actually doing the computation. Rather, there is uncertainty about how the solution value improves over time. Our performance profile tree allows us to capture this uncertainty. The tree can be used to determine $P(v^z|t)$ denoting the probability that running the algorithm for t time steps produces a solution of value v^z.

Unlike previous methods for performance profile based deliberation control, our performance profile tree directly supports conditioning on the *path* of solution quality so far.[4] The performance profile tree that applies given a path of computation so far is simply the subtree rooted at the current node n. We denote this subtree by $\mathcal{T}^z(n)$. If an agent is at a node n with value v, then when estimating how much added deliberation would increase the solution value, the agent need only consider paths that emanate from node n. The probability, $P_n(n')$, of reaching a particular future node n' in $\mathcal{T}^z(n)$ given that the current node is n is simply the product of the probabilities on the path from n to n'. Similarly, given that the current node is n, the expected solution quality after allocating t more time steps to this problem is

$$\sum_{\{n' | n' \text{ is a node in } \mathcal{T}^z(n) \text{ with depth } t\}} P_n(n') \cdot V(n')$$

This can be easily computed using depth-first-search with a depth limit t in $\mathcal{T}^z(n)$.

Computation plays several strategic roles in the game. First, it improves the solution that is available—for any one of the three problems. Second, it resolves some of the uncertainty about what future computation steps will yield. Third, it gives information about what solution qualities the opponent has encountered and can expect. This helps in estimating what solution quality the other agent has available on any of the three problems. It also helps in estimating what computations the other agent might have done and might do. Therefore, in equilibrium, an agent may want to allocate computation on its individual problem, the joint problem, and even on the opponent's problem. We will show how agents use the performance profile trees to handle these considerations.

Special Case: Deterministic Performance Profiles In a deterministic performance profile, $v^z(t) \in \Re$ is known for all t. In this setting, the tree that represents the performance

[3]The more finely solution quality and time are discretized, the more accurate deliberation control is possible. However, with more refined discretization, the number of possible runs increases (it is $O(m^T)$ where m is the number of levels of solution quality), so more runs need to be seen to populate the space. Furthermore, the space should be populated densely to get good probability estimates on the edges of the performance profile trees.

[4]Our results apply directly to the case where the conditioning on the path is based on other solution features in addition to solution quality. For example, in a scheduling problem, the distribution of slack can significantly predict how well an iterative refinement algorithm can further improve the solution.

profile has only one path. Before using any computation, the agents can determine what the value will be after any number of computation steps devoted to any one of the three problems. So, computation does not provide any information about the expected results of future computations. Also, computation does not provide any added information about the performance profiles, which could be used to estimate the other agent's computational actions.

In settings where the performance profiles are not deterministic, we assume that the agents have the same performance profile trees $\mathcal{T}^1$, $\mathcal{T}^2$, and $\mathcal{T}^{1\cup2}$ which are common knowledge. One scenario where the agents have the same performance profile trees is where the agents use the same algorithm and have seen the same training instances. This is arguably roughly the case in practice if the parties have been solving the same type of instances over time, and the algorithms have evolved through experimentation and publication. In settings where the performance profiles are deterministic, all of our results go through even if the agents have different performance profile trees $\mathcal{T}_1^1$, $\mathcal{T}_1^2$, $\mathcal{T}_1^{1\cup2}$, $\mathcal{T}_2^1$, $\mathcal{T}_2^2$, and $\mathcal{T}_2^{1\cup2}$—assuming that these are common knowledge.

Bargaining

At some point in time, T, there is a deadline at which time both agents must stop deliberating and enter the bargaining round. The agents perform their computational actions in parallel with no communication between them until the deadline is reached. Call the value of the solution computed by that time by agent i to agent 1's problem v_i^1, to agent 2's problem v_i^2, and to the joint problem $v_i^{1\cup2}$. At that time, the agents decide whether to pool or not, and in the former case they also have to decide how to divide the value of the solution to the joint problem. These decisions are made via bargaining. One agent, α, $\alpha \in \{1,2\}$, makes a take-it-or-leave-it offer, x_α^o, to the other agent, β, about how much agent β's payoff will be if they pool. Agent β can then accept or reject. If agent β accepts, the agents pool and use agent α's solution to the joint problem. Agent β's payoff is x_α^o as proposed and agent α gets the rest of the value of the solution: $v_\alpha^{1\cup2} - x_\alpha^o$. If agent β rejects, both agents implement their own computed solutions to their own individual problems, in which case agent 1's payoff is v_1^1 and agent 2's payoff is v_2^2.

Before the deadline, the agents may or may not know who is to make the offer. The probability that agent 1 will be the proposer is P_{prop}, and this is common knowledge. When agents reach the bargaining stage, each agent's strategy is captured by an *offer-accept vector*. An offer-accept vector for agent 1 is $OA_1 = (x_1^o, x_1^a) \in \mathbf{R}^2$, where x_1^o is the amount that agent 1 would offer if it had to make the proposal, and x_1^a is the minimum value it would accept if agent 2 made the proposal. The offer-accept vector for agent 2 is defined similarly.

The agents strategies incorporate actions from both parts of the game. For the deliberation part of the game, an agent's strategy is a mapping from the state of deliberation to the next deliberation action (i.e., selecting which solution z, $z \in \{1, 2, 1\cup2\}$ to compute another time step on—in words, whether to compute on the agent's own problem, the other agent's problem, or the joint problem). At the deadline, T, each agent has to decide its offer-accept vector. Therefore, the strategy at time T is a mapping from the state of deliberation at time T to an offer-accept vector.

Definition 2 *A* **strategy**, S_1 *for agent 1 with deadline* T *is*

$$S_1 = ((S_1^{D,t})_{t=0}^T, S_1^B)$$

where the deliberation strategy

$$S_1^{D,t} : \theta_1^{t-1} \rightarrow \{a^1, a^2, a^{1\cup2}\}$$

is a mapping from the deliberation state θ_1^{t-1} *at time* $t-1$ *to a deliberation action* a^z *where* a^z *is the action of computing one time step on the solution for problem* $z \in \{1, 2, 1\cup2\}$.

The bargaining strategy $S_1^B : \theta_1^T \rightarrow \Re^2$ *is a mapping from the final deliberation state to an offer-accept vector* (x_1^o, x_1^a). *A strategy,* S_2, *for agent 2 is defined analogously.*

Our analysis will also allow *mixed strategies*. A mixed strategy for agent 1 is $S_1 = ((\tilde{S}_1{}^{D,t})_{t=0}^T, S_1^B)$ where $\tilde{S}_1{}^{D,t}$ is a mapping from a deliberation state θ_1^t to a probability distribution over the set of deliberation actions $\{a^1, a^2, a^{1\cup2}\}$. We let p^1 be the probability that an agent takes action a^1, p^2 be the probability that an agent takes action a^2, and therefore, $1-p^1-p^2$ is the probability that an agent takes action $a^{1\cup2}$. It is easy to show that in equilibrium, each agent will use a pure strategy for picking its offer-accept vector [5] (i.e., the agent plays one vector with probability 1), so in the interest of simplifying the notation, we define S_1^B as a pure strategy as before.

Proposer's Expected Payoff

Say that at time T the proposing agent, α, is in deliberation state $\theta_\alpha^T = \langle n_\alpha^1, n_\alpha^2, n_\alpha^{1\cup2}\rangle$ and the other agent, β, is in deliberation state $\theta_\beta^T = \langle n_\beta^1, n_\beta^2, n_\beta^{1\cup2}\rangle$. Each agent has a set of beliefs (a probability distribution) over the set of deliberation states in which the other agent may be. If agent α offers agent β value x_α^o, then the expected payoff to agent α is

$$E[\pi_\alpha(\theta_\alpha^T, x_\alpha^o, S_\beta)] = P_a(x_\alpha^o)[V(n_\alpha^{1\cup2}) - x_\alpha^o] + (1 - P_a(x_\alpha^o))V(n_\alpha^\alpha)$$

where $P_a(x_\alpha^o)$ is the probability that agent β will accept an offer of x_α^o. These probabilities are determined by agent α's beliefs.

We can determine the proposer's expected payoff of following a particular strategy as follows. Assume agent α is following strategy $S_\alpha = ((p^{1,i}, p^{2,i})_{i=1}^T, (x_\alpha^o, x_\alpha^a))$ and agent β is following strategy S_β. At time t, if agent α is in deliberation state θ_α^t, the expected payoff is

$$\begin{aligned}
&E[\pi_\alpha(\theta_\alpha^t, ((p^{1,i}, p^{2,i})_{i=t}^T, x_\alpha^o), S_\beta)] = \\
&p^{1,t} \sum_{\theta_\alpha^{t+1} \in \Theta(\theta_\alpha^t, a^1)} P(\theta_\alpha^{t+1}) E[\pi_\alpha(\theta^{t+1}, ((p^{1,i}, p^{2,i})_{i=t+1}^T, x_\alpha^o), S_\beta)] \\
&+ p^{2,t} \sum_{\theta_\alpha^{t+1} \in \Theta(\theta_\alpha^t, a^2)} P(\theta_\alpha^{t+1}) E[\pi_\alpha(\theta^{t+1}, ((p^{1,i}, p^{2,i})_{i=t+1}^T, x_\alpha^o), S_\beta)] \\
&+ (1 - p^{1,t} - p^{2,t}) \sum_{\theta_\alpha^{t+1} \in \Theta(\theta_\alpha^t, a^{1\cup2})} P(\theta_\alpha^{t+1}) E[\pi_\alpha(\theta^{t+1}, ((p^{1,i}, p^{2,i})_{i=t+1}^T, x_\alpha^o), S_\beta)]
\end{aligned}$$

[5] This holds whether or not the proposer is known in advance.

where

$$\Theta(\theta_\alpha^t, a^z) = \{\theta_\alpha^{t+1} | \theta_\alpha^{t+1} \text{is reachable from } \theta_\alpha^t \text{via action } a^i\}.$$

Overloading the notation, we denote the expected payoff to agent α from following strategy S_α, given that agent β follows strategy S_β by

$$E[\pi_1(S_\alpha, S_\beta)] \stackrel{\text{def}}{=} E[\pi_\alpha(\theta_\alpha^0, ((p^{1,i}, p^{2,i})_{i=1}^T, S_\beta))]$$

Equilibria and Algorithms

We want to make sure that the strategy that we propose for each agent—and according to which we study the outcome—is indeed the best strategy that the agent has from its self-interested perspective. This makes the system behave in the desired way even though every agent is designed by and represents a different self-interested real-world party. One approach would be to just require that the analysis shows that no agent is motivated to deviate to another strategy given that the other agent does not deviate (i.e., *Nash equilibrium*). We actually place a stronger requirement on our method. We require that at any point in the game, an agent's strategy prescribes optimal actions from that point on, given the other agent's strategy and the agent's beliefs about what has happened so far in the game. We also require that the agent's beliefs are consistent with the strategies. This type of equilibrium is called a *perfect Bayesian equilibrium* (PBE) (Kreps 1990).

An agent's offer-accept vector is affected by the solutions that it computes and also what it believes the other agent has computed for solutions. The *fallback* value of an agent is the value it obtained for the solution to its own problem. An agent will not accept any offer less than its fallback.

In making a proposal, agent α must try to determine agent β's fallback value and then decide whether, by making an acceptable proposal to agent β, agent α's payoff would be greater than or less than its own fallback.[6]

The games differ significantly based on whether the proposer is known in advance or not, as will be discussed in the next sections.

Known Proposer

For an agent that is never going to make an offer, we can prescribe a dominant strategy independent of the statistical performance profiles:

Proposition 1 *If an agent, β, knows that it cannot make a proposal at the deadline T, then it has a dominant strategy of computing only on its own problem, and accepting any offer x_α^o such that $x_\alpha^o \geq V(n)$ where n is the node in the performance profile $\mathcal{T}^\beta$ that agent β has reached at time T. If the performance profile does not flatten before the deadline ($V(n') < V(n)$ for every node n' on the path to n), then this is the unique dominant strategy.*

Proof: In the event that an agreement is not reached, agent β could not have achieved higher payoff than by computing on its individual problem (even if it knows that further computation will not improve its solution). In the event that an agreement is reached, agent β would have been best off by computing so as to maximize the minimal offer it will accept, $V(n_\beta^\beta)$. Since solution quality is nondecreasing in computation time, if agent β deviates and computes t steps on a different problem, then the value of its fallback is $V({n'}_\beta^\beta) \leq V(n_\beta^\beta)$ where $\mathrm{time}({n'}_\beta^\beta) = \mathrm{time}(n_\beta^\beta) + t$. If $V(n') < V(n)$ for every node n' on the path to n, then this inequality is strict.

Corollary 1 *In the games where the proposer is known, there exists a pure strategy PBE.*

Proof: By Proposition 1, the receiver of the offer has a dominant strategy. Say the proposer were to use a mixed strategy. In general, every pure strategy that has nonzero probability in a best-response mixed strategy has equal expected payoff (Kreps 1990). Since mixing by the proposer will not affect the receiver's strategy, the proposer might as well use one of the pure strategies in its mix.

The equilibrium differs based on whether or not the deadline is known, as discussed in the next subsections.

Known Proposer, Known Deadline In the simplest setting, both the deadline and proposer are common knowledge. Without loss of generality we assume that agent 1 is the proposer. The game differs based on whether the performance profiles are deterministic or stochastic.

`Deterministic Performance Profiles` In an environment where the performance profiles are deterministic, the equilibria can be analytically determined.

Proposition 2 *There exists a PBE where agent 2 will only compute on its own problem, and agent 1 will never split its computation. It will either compute solely on its own problem or solely on the joint problem. The PBE payoffs to the agents are unique, and the PBE is unique unless the performance profile that an agent is computing on flattens, after which time it does not matter where the agent computes since that does not change its payoff or bargaining strategy. The PBEs are also the only Nash equilibria.*

Proof: Let $\eta_1^{1\cup 2}$ be the node in $\mathcal{T}^{1\cup 2}$ that agent 1 reaches after allocating all of its computation on the joint problem. Let η_1^1 be the node in $\mathcal{T}^1$ that agent 1 reaches after allocating all of its computation on its own problem. Let η_2^2 be the node in $\mathcal{T}^2$ that agent 2 reaches after allocating all of its computation on its own problem.

By Proposition 1, agent 2 has a dominant strategy to compute on its own solution (unless its performance profile flattens after which time it does not matter where the agent computes since that does not change its payoff). Agent 1's strategies are more complex since they depend on agent 2's final fallback value, $V(\eta_2^2)$, and also on what potential values the joint solution and 1's individual solution may have.

1. **Case 1:** $V(\eta_1^{1\cup 2}) - V(\eta_2^2) > V(\eta_1^1)$. Agent 2 will accept any offer greater than or equal to $V(n_2^2)$ since that is its fallback. If agent 1 makes an offer that is acceptable to agent 2, then the highest payoff that agent 1 can receive is $V(\eta_1^{1\cup 2}) - V(\eta_2^2)$. If this value is greater than

[6] Since solution values are discretized, the offer-accept vectors are also from a discrete space.

$V(\eta_1^1)$—i.e., the highest fallback value agent 1 can have—then agent 1 will make an acceptable offer. To maximize the amount it will get from making the offer, agent 1 must compute only on the joint problem. Any deviation from this strategy will result in agent 1 receiving a lesser payoff (and strictly less if its performance profile has not flattened).

2. **Case 2:** $V(\eta_1^{1\cup 2}) - V(\eta_2^2) < V(\eta_1^1)$. Any acceptable offer that agent 1 makes results in agent 1 receiving a lesser payoff than if it had computed on its own solution solely, and made an unacceptable offer (and strictly less if its performance profile has not flattened). Therefore agent 1 will compute only on its own problem until that performance profile flattens, after which it does not matter where it allocates the rest of its computation.

3. **Case 3:** $V(\eta_1^{1\cup 2}) - V(\eta_2^2) = V(\eta_1^1)$. By computing only on its own problem, agent 1's payoff is $V(\eta_1^1)$. By computing only on the joint problem, the payoff is $V(\eta_1^{1\cup 2}) - V(\eta_2^2)$. These payoffs are equal. However, by dividing the computation across the problems, both payoffs decrease (unless at least one of the two performance profiles has flattened, after which it does not matter where the agent allocates the rest of its computation).

The above arguments also hold for Nash equilibrium.

`Stochastic Performance Profiles` If the performance profiles are stochastic, determining the equilibrium is more difficult. By Proposition 1, agent 2 has a dominant strategy, S_2, and only computes on its individual problem (if that performance profile has flattened and agent 2 has computed on agent 1's or the joint problem thereafter, this does not change agent 2's fallback, and this is the only aspect of agent 2 that agent 1 cares about).

However, based on the results it has obtained so far, agent 1 may decide to switch the problem on which it is computing—possibly several times. We use a dynamic programming algorithm to determine agent 1's best response to agent 2's strategy. The base case involves looping through all possible deliberation states θ_1^T for agent 1 at the deadline T. Each θ_1^T determines a probability distribution over the set of nodes agent 2 reached by computing T time steps. For any offer x_1^o that agent 1 may make, the probability that agent 2 will accept is

$$P_a(x_1^o) = \sum_{\{n^2 | n^2 \text{in subtree } \mathcal{T}^2(n_1^2) \text{ at depth } T - \text{time}(n_1^2) \text{ s.t. } V(n^2) \leq x\}} P(n^2)$$

Using this expression for $P_a(x)$, the best offer, x_1^o, that agent 1 can make to agent 2 is

$$x_1^o(\theta_1^T) = \arg\max_x [E[\pi_1(\theta_1^T, x, S_2)]]$$

For each deliberation state, θ_1^T, we can compute the expected payoff for agent 1, if at time t agent 1 is in deliberation state θ_1^t and then executes the sequence of actions $((a^{z,i})_{i=t}^T, x_1^o(\theta_1^T))$. The expected payoff is

$$E[\pi_1(\theta_1^t, ((a^{z,i})_{i=t}^T, x_1^o), S_2)] = \sum P(\theta_1^{t+1}) E[\pi_1(\theta_1^{t+1}, ((a^{z,i})_{i=t+1}^T, x_1^o), S_2)]$$

The sum is over the set $\{\theta_\alpha^{t+1} | \theta_1^{t+1}$ is reachable from θ_1^t via action $a^z\}$. The algorithm works backwards and determines the optimal sequence of actions, $(a^{*,i})_{i=1}^T$, for agent 1. For every time t it solves

$$a^{*t} = \max_a [E[\pi_1(\theta_1^t, ((a, (a^{*i})_{i=t+1}^T), x_1^o), S_2)]]$$

It returns the optimal sequence of actions, $(a^{*i})_{i=1}^T$, and the expected payoff $E[\pi_1(((a^{*i})_{i=1}^T, x_o^1), S_2)]$.

Algorithm 1 *StratFinder1(T)*
For each deliberation state θ_1^T at time T

$$x_o^1(\theta_1^T) \leftarrow \arg\max_x [E[\pi_1(\theta_1^T, x, S_2)]]$$

For time $t = T - 1$ down to 1
For each deliberation state θ_1^t

$$a^{*t} \leftarrow \max_a [E[\pi_1(\theta_1^t, ((a, (a^{*i})_{i=t+1}^T), x_1^o), S_2)]]$$

Return $(a^{*i})_{i=1}^T$ and $E[\pi_1(((a^{*i})_{i=1}^T, x_o^1), S_2)]$

Proposition 3 *Algorithm 1 correctly computes a PBE strategy for agent 1.*[7] *Assume that the degree of any node in $\mathcal{T}^1$ is at most B^1, the degree of any node in $\mathcal{T}^2$ is at most B^2 and the degree of any node in $\mathcal{T}^{1\cup 2}$ is at most $B^{1\cup 2}$. Algorithm 1 runs in $O((B^1 B^2 B^{1\cup 2})^{T^2})$ time.*

Known Proposer, Unknown Deadline There are situations where agents may not know the deadline. We represent this by a probability distribution $Q = \{q(i)\}_{i=1}^T$ over possible deadlines. Q is assumed to be common knowledge.

Whenever time t is reached but the deadline does not arrive, agents update their beliefs about Q. The new distribution is $Q' = \{q'(i)\}_{i=t}^T$ where $q'(t) = \frac{q(t)}{\sum_{j=t}^T q(j)}$.

`Stochastic Performance Profiles` The algorithm differs from Algorithm 1 in that it considers the probability that the deadline might arrive at any time.

Algorithm 2 *StratFinder2(Q)*
For each deliberation state θ_1^T at time T

$$x_o^1(\theta_1^T) \leftarrow \arg\max_x [E[\pi_1(\theta_1^T, x, S_2)]]$$

For $t = T - 1$ down to 1 $q'(t) \leftarrow \frac{q(t)}{\sum_{j=t}^T q(j)}$
For each deliberation state θ_1^t

$$x_o^1(\theta_1^t) \leftarrow \arg\max_x [E[\pi_1(\theta^t, x, S_2)]]$$

$$a^{*t} \leftarrow \max_a [q' t E[\pi_1(\theta_1^t, x_1^o(\theta_1^t), S_2)]$$

$$+ (1 - q'(t)) \max_a [E[\pi_1(\theta_1^t, ((a, (a^{*i})_{i=t+1}^T), x_1^o), S_2)]$$

Return $(a^{*i})_{i=1}^T$ and $E[\pi_1(((a^{*i})_{i=1}^T, x_o^1), S_2)]$

Proposition 4 *Algorithm 2 correctly computes a PBE strategy for agent 1. Assume that the degree of any node in $\mathcal{T}^1$ is at most B^1, the degree of any node in $\mathcal{T}^2$ is at most B^2 and the degree of any node in $\mathcal{T}^{1\cup 2}$ is at most $B^{1\cup 2}$. Algorithm 2 runs in $O((B^1 B^2 B^{1\cup 2})^{T^2})$ time.*

[7] By keeping track of equally good actions at every step, Algorithms 1, 2, and 3 can return all PBE strategies for agent 1.

`Deterministic Performance Profiles` When the performance profiles are deterministic, determining an optimal strategy for agent 1 is a special case of Algorithm 2. Since there is no uncertainty as to agent 2's fallback value, agent 1 need never compute on agent 2's problem. Therefore, agent 1 will only be in deliberation states $\langle n_1^1, n_1^2, n_1^{1\cup 2}\rangle$ where $\mathrm{time}(n_1^2) = 0$. Therefore, strategies that include computation actions a^2 need not be considered. This, and the lack of uncertainty in which deliberation state action a leads to, greatly reduce the space of deliberation states to consider. Denote by Γ_1^t any deliberation state of agent 1 where $\mathrm{time}(n_1^1) + \mathrm{time}(n_1^{1\cup 2}) = t$ and $\mathrm{time}(n_1^2) = 0$.

Algorithm 3 *StratFinder3(Q)*
For each deliberation state Γ_1^T at time T

$$x_o^1(\Gamma_1^T) \leftarrow \arg\max_x [E[\pi_1(\Gamma_1^T, x, S_2)]]$$

For $t = T-1$ down to 1 $q'(t) \leftarrow \frac{q(t)}{\sum_{j=t}^{T} q(j)}$
For each deliberation state Γ_1^t

$$x_o^1(\Gamma_1^t) \leftarrow \arg\max_x [E[\pi_1(\Gamma^t, x, S_2)]]$$

$$a^{*t} \leftarrow \max_a [q' t E[\pi_1(\Gamma_1^t, x_1^o(\Gamma_1^t), S_2)]$$

$$+ (1-q'(t)) \max_a [E[\pi_1(\Gamma_1^t, ((a, (a^{*i})_{i=t+1}^T), x_1^o), S_2)]$$

Return $(a^{*i})_{i=1}^T$ and $E[\pi_1(((a^{*i})_{i=1}^T, x_o^1), S_2)]$

Proposition 5 *With deterministic performance profiles, Algorithm 3 correctly computes a PBE strategy for agent 1 in $O(T^2)$ time.*

Unknown Proposer

This section discusses the case where the proposer is unknown, but the probability of each agent being the proposer is common knowledge. The deadline may be common knowledge. Alternatively, the deadline is not known but its distribution is common knowledge.

Proposition 6 *There are instances (defined by $\mathcal{T}^1$, $\mathcal{T}^2$, and $\mathcal{T}^{1\cup 2}$) of the game that have a unique mixed strategy PBE, but no pure strategy PBE (not even a pure strategy Nash equilibrium).*

Proof: Let the deadline $T = 2$, and let p be the probability that agent 1 will be the proposer. Consider the following $\mathcal{T}^1$, $\mathcal{T}^2$, and $\mathcal{T}^{1\cup 2}$. Assume that $v^1(1) = v^1(2)$ and $v^2(1) = v^2(2)$. Furthermore, assume that the values satisfy the following constraints:

- $v^{1\cup 2}(1) \geq v^1(1)$
- $v^{1\cup 2}(1) \geq v^2(1)$
- $v^1(1) + v^2(1) \geq v^{1\cup 2}(1)$
- $pv^{1\cup 2}(2) \geq pv^{1\cup 2}(1) + (1-p)v^1(1)$
- $v^1(1) \geq p(v^{1\cup 2}(2) - v^2(1))$
- $pv^2(1) + (1-p)v^{1\cup 2}(1) \geq (1-p)v^{1\cup 2}(2)$
- $(1-p)(v^{1\cup 2}(2) - v^1(1)) \geq v^2(1)$

Agent 1 has two undominated strategies: to compute only on the joint problem, or to compute one step on the joint and one on its individual problem. Agent 2 also has two undominated strategies: to compute only on the joint problem, or to compute one step on the joint and one step on its individual problem. There is no pure strategy equilibrium in this game. However, there is a mixed strategy equilibrium where agent 1 computes on the joint problem only, with probability

$$\gamma = \frac{pv^2(1) - pv^{1\cup 2}(2) + v^1(1)}{pv^2(1) - pv^{1\cup 2}(1) + 2v^1(1) - pv^1(1)}$$

and agent 2 computes on the joint problem only with probability

$$\delta = \frac{v^{1\cup 2}(2) - v^2(1) - pv^{1\cup 2}(2) - v^1(1) + pv^1(1)}{pv^2(1) - v^2(1) + v^{1\cup 2}(1) - pv^{1\cup 2}(1) - v^1(1) + pv^1(1)}$$

One approach of solving for PBE strategies is to convert the game into its normal form. There are efficient algorithms for solving normal form games, but the conversion itself usually incurs an exponential blowup since the number of pure strategies is often exponential in the depth of the game tree. (Koller, Megiddo, & Stengel 1996) suggest representing the game in *sequence form* which is more compact than the normal form representation. They then solve the game using Lemke's algorithm to find Nash equilibria. Their algorithm can be directly used to solve our problem where the proposer is unknown. Their algorithm is guaranteed to find some Nash equilibrium strategies, albeit not all.

Conclusions and Future Research

Noncooperative game-theoretic analysis is necessary to guarantee nonmanipulability of systems that consist of self-interested agents. However, the equilibrium for rational agents does not generally remain an equilibrium for computationally limited agents. This leaves a potentially hazardous gap in theory. This paper presented a framework and the first steps toward filling that gap.

We studied a setting where each agent has an intractable optimization problem, and the agents can benefit from pooling their problems and solving the joint problem. We presented a fully normative model of deliberation control that allows agents to condition their projections on the problem instance and path of solutions seen so far. Using that model, we solved the equilibrium of the bargaining game. This is, to our knowledge, the first piece of research to treat deliberation actions strategically via noncooperative game-theoretic analysis.

In games where the agents know which one gets to make a take-it-or-leave-it offer to the other, the receiver of the offer has a dominant strategy of computing on its own problem, independent of the algorithm's statistical performance profiles. It follows that these games have pure strategy equilibria. In equilibrium, the proposer can switch multiple times between computing on its own, the other agent's, and the joint problem. The games differ based on whether or not the deadline is known and whether the performance profiles are deterministic or stochastic. We presented algorithms for computing a pure strategy equilibrium in each of these variants. For games where the proposer is not known in advance, we use a general algorithm for finding a mixed strategy equilibrium in a 2-person game. This generality comes at the cost of potentially being slower than our algorithms for the other cases.

This area is filled with promising future research possibilities. We plan to extend this work to more than two agents, to settings where the agents have algorithms with different performance profiles, to games where computation is costly instead of limited, and games where bargaining is allowed amidst computation, not just after it. In such settings, the offers and rejections along the way signal about the agents' computation strategies, the results of their computations so far, and what can be expected from further computation.

Acknowledgments

This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, and Grant IIS-9800994.

References

Baum, E. B., and Smith, W. D. 1997. A Bayesian approach to relevance in game playing. *Artificial Intelligence* 97(1–2):195–242.

Boddy, M., and Dean, T. 1994. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence* 67:245–285.

Hansen, E. A., and Zilberstein, S. 1996. Monitoring the progress of anytime problem-solving. In *AAAI*, 1229–1234.

Horvitz, E. 1987. Reasoning about beliefs and actions under computational resource constraints. In *3rd Workshop on Uncertainty in AI*, 429–444. Seattle.

Horvitz, E. J. 1997. Models of continual computation. In *AAAI*, 286–293.

Jehiel, P. 1995. Limited horizon forecast in repeated alternate games. *J. of Economic Theory* 67:497–519.

Koller, D.; Megiddo, N.; and Stengel, B. 1996. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior* 14(2):247–259.

Kreps, D. M. 1990. *A Course in Microeconomic Theory*. Princeton University Press.

Rubinstein, A. 1998. *Modeling Bounded Rationality*. MIT Press.

Russell, S., and Subramanian, D. 1995. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research* 1:1–36.

Russell, S., and Wefald, E. 1991. *Do the right thing: Studies in Limited Rationality*. The MIT Press.

Sandholm, T., and Lesser, V. R. 1994. Utility-based termination of anytime algorithms. In *ECAI Workshop on Decision Theory for DAI Applications*, 88–99. Extended version: UMass Amherst, CS TR 94-54.

Sandholm, T., and Lesser, V. R. 1997. Coalitions among computationally bounded agents. *Artificial Intelligence* 94(1):99–137. Early version in IJCAI-95.

Sandholm, T. 1996. Limitations of the Vickrey auction in computational multiagent systems.*ICMAS*,299-306.

Simon, H. A. 1955. A behavorial model of rational choice. *Quarterly Journal of Economics* 69:99–118.

Zilberstein, S., and Russell, S. 1996. Optimal composition of real-time systems. *Artificial Intelligence* 82(1–2):181–213.

Zilberstein, S.; Charpillet, F.; and Chassaing, P. 1999. Real-time problem solving with contract algorithms. In *IJCAI*, 1008–1013.