An Optimal Contracting Strategy in a Digital Library

Sunju Park and Edmund H. Durfee

Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109-2110
{boxenju, durfee}@eecs.umich.edu

Abstract

Agents can benefit from contracting some of their tasks that cannot be performed by themselves or that can be performed more efficiently by other agents. Developing an agent's contracting strategy in the University of Michigan Digital Library (UMDL), however, is not easy for the following reasons. The UMDL consists of self-interested agents who will perform a task of another agent's only when doing it is their own interests. In addition, multiple contracts take place concurrently such that other contracts currently in the system may have an impact on the success of one's own contract. Therefore, an agent who has a task (contractor) needs to model what the other self-interested agents think and will do, and it also needs to consider the influence of other contracts on its contract.

In this paper, we define the contractor's and the contractee's decision problems in the UMDL contracting situations, and present a contracting strategy by which a contractor can determine an optimal payment to offer. The contractor's problem is to find a payment that maximizes its expected utility, and it finds such a payment by modeling the contracting process stochastically using a Markov process. The Markov-process model is built based on the information the contractor has about the potential contractees and about the other contracts in the system. Our early results show that the contractor receives a higher utility when thinking about the potential contractees and the other contracts.

Introduction

As agent technology advances, we encounter many systems in which multiple agents interact with one another. As noted in (Sandholm & Lesser 1995a), the importance of automated contracts in such multiagent systems is likely to increase where the agents representing small enterprises can form short term contracts to be able to respond to larger tasks than they can individually do.

One example of such federated multiagent systems is the University of Michigan Digital Library (UMDL), a large-scale multiagent system for information services (Atkins et al. 1996). The agents in the UMDL coordinate their activities to accomplish a variety of high-valued information services, such as delivering a document to a user, processing a query, filtering information, and so on.

Instead of using central planning, we advocate an open architecture, where users and information collections will choose to be part of the system, and where the population of value-added information-service providers will evolve based on market forces. In terms of multiagent contracts, the open architecture adopted in the UMDL has the following implications.

- Self-interested agents:

Compared to cooperative systems where the goals of agents are aligned, the agents in the UMDL are purely self-interested. The only reason for an agent to be in the UMDL system is to satisfy its needs: the agent representing a user wants to receive the needed information; the agent representing an information collection wants to publish and disseminate the information, possibly with compensation; and the agent representing an information-service provider desires to provide its specialized service in exchange for compensation. In short, the agents in the UMDL do not share a common global objective; their actions are guided by their self-interests.

- Provision of rewards:

A self-interested agent who has tasks (i.e., contractor) can benefit from contracting some of its tasks that cannot be performed by itself or that can be performed more efficiently by other agents. However, since the agent who will perform the task (i.e., contractee) is also self-interested, the contractor cannot force the contractee to do its task without some incentives. Promising rewards is one way to convince a self-interested agent to perform a task that is not amongst its tasks. Accordingly, the UMDL provides a monetary system where performing a task is rewarded by some payment which is mutually decided upon by the contractor and the contractee.

- Retraction from the agreed-upon contracts:

Multiple contracts will take place concurrently in the UMDL, and one contract may influence the success of another. For example, a contractee with a certain capability constraint may retract from its old contract (while paying a retraction penalty) to accept a new, more lucrative contract. So, when making a contract, a contractor needs to consider

the other contracts in the system which may impact the success of its task.

Consequently, an agent's contracting strategy in the UMDL will be more complicated than that for a cooperative environment or that for a single contract. When making a decision in the contracting process, the agent cannot depend on any assumption about the other agents (since they are self-interested and it has no control over them): it needs to model what the other self-interested agents think and will do. In addition, it needs to consider the influence of other contracts on its contract (through retraction). Moreover, the information about other agents and other contracts may be known to the contractor only probabilistically.

In this paper, we are asking the following two questions: (1) what are the contractor's and the contractee's decision problems in the contracting situations of the UMDL; and (2) how can the contractor make an optimal decision.

The rest of the paper consists of the following. First, we review related work, and define the contractor's and the contractee's decision problems. Then, we develop the optimal contracting strategy of the contractor. Due to space limitations, we focus on the optimal contracting strategy of the contractor, while skipping that of the contractee in this paper. In addition, we show some early experimental results and summarize the work. The appendix explains the use of Markov process in detail.

Related Work

The strategies of self-interested, rational agents has been a central issue in noncooperative game theory (Kreps 1990) (Rubinstein 1982). In general, game theorists analyze strategic interaction between agents, finding the equilibrium strategies (based on notions, such as Nash, Perfect, or Bayesian equilibria). Self-interested agent's strategies have been studied in DAI as well. Rosenschein, for example, proposes several agent strategies and protocols based on game theory (Rosenschein & Zlotkin 1994). In RMM (Gmytrasiewicz & Durfee 1993), an agent models the other agents in a recursive manner to evaluate the expected utility attached to potential actions. Kraus proposes a self-interested agent's contracting strategy based on contingency contracts (Kraus 1994).

At the risk of oversimplifying, we say that the above research in general focuses on a *single*-task contracting situation of self-interested agents. The UMDL, however, consists of multiple contracts, which demands a contracting strategy that explicitly reasons about concurrent contracts and possible retractions. Unfortunately, there has been little research on self-interested agents' contracting strategies under multiple, concurrent contracts. Sen (Sen 1993) has studied multiple contracts and their interdependencies, but his research primarily deals with *cooperative* domains and lacks explicit notions of payments.

Sandholm proposes self-interested agents' contracting strategies under multiple contracts (Sandholm 1993). Compared to our approach, however, the contractor announces the true cost of its task as the payment, not capitalizing on the opponents' costs or the impact of the other tasks competing in the system. In addition, retraction is not allowed. Recently, he proposes a leveled-commitment protocol that allows self-interested agents to retract from a contract by paying a penalty (Sandholm & Lesser 1995b), but no agent's strategy under such a protocol is developed yet.

In our work, both the contractor and the contractee are able to explicitly consider the opponents, the concurrent contracts, and the possibility of retraction when making decisions in the contracting process.

The Decision Problem

We assume the agents in the UMDL use a simple contracting protocol called Take-It-or-Leave-it (TILI). Under the TILI protocol, the contractor announces the task and its payment to potential contractees, and the potential contractees either accept or reject the offer (no counteroffers). Then, the contractor awards the task to one of the contractees who has accepted the offer. The contractee may want to retract from the contracted task while paying the retraction penalty. Retraction happens, for example, when the contractee receives a more attractive task that cannot be done along with the contractor's task.

Before continuing, let's consider an example from the UMDL where a Task Planning Agent (TPA) desires to receive a monitoring service for a certain period of time from a Remora Agent (RA)[1]. Informally, the decision problems of the TPA and the RA are as follows. The TPA (contractor) needs to figure out what payment to offer and to whom in order to have the task done. The RA (contractee), on the other hand, needs to decide whether to accept the offer or not. In addition, it may want to consider whether to retract some of the currently contracted tasks in favor of a better deal.

Contractor's Decision Problem

In principle, the contractor's decision parameters can be diverse, such as the payment, the retraction penalty, to how many agents to send the offer, and so on. In our initial development, however, we assume that the contractor is only interested in the value of the payment (while broadcasting the offer to all the available contractees and fixing the retraction penalty).

Then, the contractor's decision problem is to find the payment (ρ) that maximizes its expected utility (i.e., $maximize_{\rho}\ u(\rho)$). If the contractor is primarily interested in the payoff of a contract, the expected utility of the contractor for the payment, ρ, is defined as follows.

$$u(\rho) = P_S(\rho) \times U(Payoff_S(\rho)) + P_F(\rho) \times U(Payoff_F(\rho)),$$

[1]Both the TPA and the RA are service-providing agents (called mediators) in the UMDL. Typically, the TPA processes a task from a user, and the RA monitors a UMDL event (e.g., the change of the status of an agent) (Atkins et al. 1996).

where $P_{S/F}$ denote the probability of Success (S) and Failure (F) of a contract, and $Payoff_{S/F}$ denote the payoff of S and F, respectively, given ρ. Note that $Payoff_{S/F}$ is a more-is-better function since in general the contractor will prefer a higher payoff.

The contractor's payoff of a successful contract ($Payoff_S$) is defined as its value of the task (V) minus the payment (ρ) to the contractee and minus the total communication costs (CC)[2]. In addition, any retraction penalties (Δ) paid by the contractees are added[3]. If the contract fails, the contractor may be worse off because of the communication cost. The payoff of failure ($Payoff_F$) is minus the total communication costs plus any retraction penalties accrued, assuming the value of failure is 0. If values, payments, and costs have the same units, the payoffs are defined as follows.

$Payoff_S(\rho) = V - \rho - CC_S + \Delta_S$

$Payoff_F(\rho) = - CC_F + \Delta_F$

If the payment is higher, the probability of success (P_S) would increase (since it is more likely to be accepted by more potential contractees), but the payoff of success ($Payoff_S$) would decrease. A higher payment also implies lower probability of retraction (and thus higher probability of success), since the contractee will be less likely to retract a contract with higher payment.

In addition to the payment, many other factors—such as the contractees' costs of doing the task, the payments of the other contracts, and so on—influence the values of $P_{S/F}$ and $Payoff_{S/F}$. To compute $P_{S/F}$ and $Payoff_{S/F}$, therefore, the contractor needs to model the potential contractees and the other contracts.

Contractee's Decision Problem

As in the case of the contractor, the contractee in principle needs to model the other agents (i.e., the contractor(s) and the other competing contractees). For example, the contractee may model what the contractor thinks and try to influence the contractor: the contractee may reject the task even though it gives a profit, trying to convince the contractor that its cost of doing the task is higher and thus making the contractor offer a higher payment in the next contracting process[4]. At present, however, we consider each contracting process separately such that the contractor's information about the contractees is not changing (i.e., static), and therefore the contractee does not need to model the contractor.

On the other hand, the contractee must be able to model the other agents competing for the same task to compute the probability of getting awarded (which is used in computing the expected payoff).

Under the TILI protocol, the contractee makes two decisions. First, when the task(s) and their payments are announced, it needs to decide whether to accept some of the offers. Second, if contracted, it needs to decide whether to retract some of its contracted tasks. Since our primary focus is on the optimal contracting strategy of the contractor, we do not discuss the detailed way of computing the contractee's payoffs in this paper. Interested readers may refer to (Park, Durfee, & Birmingham 1996).

First, when the tasks are being announced, the contractee needs to find a subset of announced tasks it will accept—the subset of announced tasks which maximizes its expected payoff. For example, let's suppose the RA who has T_1 receives an announcement of T_2. And suppose the RA can perform at most one task at a time. If it accepts T_2 and gets awarded T_2, then it needs to retract one of T_1 and T_2. Therefore, the RA will accept the offer only if the payoff of accepting T_2 even with the retraction of T_1 is higher than not accepting it.

Second, when the contractee has more tasks than it can do, it may need to retract some of its tasks until it satisfies the following total capability (TC) constraint.

$\Sigma_k (C_k/\gamma_k) \leq TC,$

where $k \in$ {contracted tasks that are not retracted}, C_k is the cost of doing the task k, γ_k is the time needed to complete the task k, and TC is the total capability (the maximum cost that the contractee can take on per each time unit). The total capability constraint says that the sum of all the costs spent per each time unit should be less than the maximum cost that the contractee can spend per time unit. It should be noted that all tasks are assumed to be persistent (that is, a task should be started from the time it is contracted and be persistent until its completion). The RA's monitoring service is an example of such persistent tasks.

The contractee's decision at the contracted state is therefore to find a subset of its contracted tasks with the maximum payoff while satisfying the total capability constraint. That is,

find K with max(Payoff(K)) and $\Sigma_k (C_k/\gamma_k) \leq TC$.

Then, the contractee will retract the rest of the contracted tasks.

Contractor's Optimal Contracting Strategy

In this section, we propose an optimal contracting strategy for the contractor. We have developed a four-step contracting strategy for the contractor to compute the unknown parameters and therefore to find the optimal payment.

(1) The contractor models the contracting process using Markov chains (MC).

(2) It computes the transition probabilities between the MC states.

---

[2] The communication cost represents the total overhead of the contracting process incurred by the contractor.

[3] In our setting, retraction from the contractor side will not happen since its payoff will be the same no matter who it picks.

[4] This is similar to a dynamic game of incomplete information where the contractor tries to obtain information about the contractee, and the contractee tries to influence the contractor's information to its favor.

(3) From the MC model, it computes the probabilities and payoffs of S and F.
(4) Using the probabilities and payoffs of S and F, it finds the payment that maximizes its expected utility.

Step 1: Modeling the contracting process

The contractor can model various contracting processes using absorbing Markov chains with a set of transient states and two absorbing states (S and F). An example of a contracting process model is shown in Figure 1.

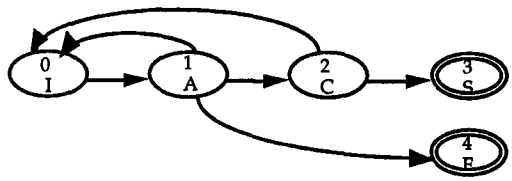

Figure 1: An example of a contracting-process model

In Figure 1, state 0 is the initial state. The contracting process goes to state 1 when the contractor announces the task and its payment to the potential contractees. State 2 is the contracted state where the contractor has awarded the task to one of those who accepted its offer. State 3 and state 4 are the success and the failure state, respectively.

From state 0, the process goes to states 1 by default. From state 1, the process goes to state 2 as long as at least one agent accepts the offer. If no agent accepts the offer, the process goes to state 4. The process may go back to the initial state (state 0) if there are some agent(s) who can perform the task but are busy at the moment. Then, from state 0, the contractor will try to announce the task again with the same payment. The process goes from state 2 to state 0, when the contractee has more tasks than it can perform (i.e., its capability is exceeded) and retracts the contractor's task. When the task is successfully completed, the transition from state 2 to state 3 happens.

Step 2: Computing the transition probabilities

The contractor needs to define the transition probabilities between the MC states. The transition probability from state i to state j, P_{ij}, is a function of the payment (ρ), the retraction penalty (δ), the information about the contractees' costs of doing the task and their capabilities, and the number of other tasks and their payments.

As an example, let's compute the transition probabilities of Figure 1, assuming only a single contract is going on in the system (i.e., no other task is present). P_{01} is 1 by default. Since there is no other task, no agent will be busy at the announced state (yielding $P_{10} = 0$), and the task will always be completed once the contract is made (i.e., $P_{20} = 0$ and $P_{23} = 1$). If there is at least one agent who accepts the contractor's offer, the contract is made (i.e., the transition from state 1 to state 2 happens). Let $f^i(c)$ be the probability density function (PDF) of agent i's cost of doing the task. The contractor can compute the transition probabilities of P_{14} and P_{12} as follows.

$P(A^i)$ = Probability that agent i accepts the payment ρ (i.e., agent i's cost of doing the task is less than ρ)
$= \int_{-\infty}^{\rho} f^i(c)dc$.

P_{14} = Probability that no agent accepts the offer (no agents' cost is less than ρ)
$= (1\text{-}P(A^1)) \times (1\text{-}P(A^2)) \times ... \times (1\text{-}P(A^n)) = \Pi_i(1\text{–}P(A^i))$.

P_{12} = Probability that at least one agent accepts the offer (at least one agent's cost is less than ρ)
$= 1 - P_{14} - P_{10}$
$= 1 - P_{14}$.

Step 3: Computing the probabilities and payoffs

Having the model of the contracting process and its transition probabilities, the contractor can compute the unknown parameters of the utility function (i.e., the probabilities and payoffs of S and F) using the method we have developed. The detailed explanation can be found in the Appendix.

Step 4: Finding the optimal payment

When the probabilities and payoffs of S and F are ready, finding the best payment from the utility function is an optimization problem. At present, we use a simple generate-and-test: we generate the utility values for various ρ, and choose ρ with the highest utility. We are investigating an appropriate optimization technique that can be used for Step 4.

Experiment

In this section, we report some early experimental results of the proposed contracting strategy. The experimental setting is as follows. There are two tasks in the system: T_1 (the contractor's task) and T_2 (another task being contracted), each of which takes one time unit to complete. The contractor values T_1 as 20 (i.e., $V = 20$), and it knows the payment of T_2 is 10. (It does not have to know about the other contractor's value of T_2). The retraction penalty of both tasks is 2.

There are three potential contractees (A^1, A^2, A^3). The contractor has imperfect models of them, especially their costs which it represents using probability density functions (PDFs). For simplicity, we assume that the total capability of each agent is known to the contractor: let TC of A^1, A^2, A^3 be 16, 18, and 20, respectively. The PDFs of agent i's cost of doing task j ($f_j^i(c_j)$) are as follows.

$$f_1^1(c_1)=\begin{cases}\frac{1}{6} & 5\le c_1\le 11\\ 0 & otherwise\end{cases},\quad f_2^1(c_2)=\begin{cases}\frac{1}{6} & 5\le c_2\le 11\\ 0 & otherwise\end{cases},$$

$$f_1^2(c_1)=\begin{cases}\frac{1}{6} & 6\le c_1\le 12\\ 0 & otherwise\end{cases},\quad f_2^2(c_2)=\begin{cases}\frac{1}{6} & 6\le c_2\le 12\\ 0 & otherwise\end{cases},$$

$$f_1^3(c_1)=\begin{cases}\frac{1}{6} & 7\le c_1\le 13\\ 0 & otherwise\end{cases},\quad f_2^3(c_2)=\begin{cases}\frac{1}{6} & 7\le c_2\le 13\\ 0 & otherwise\end{cases}.$$

If every state transition takes one time unit, the contractor can model the two task contracts as in Figure 2. Note that

state D (done) is used to represent both S and F states of T_2, because the contractor does not care about the result of T_2, and because it is generally a good idea to keep the number of states small.

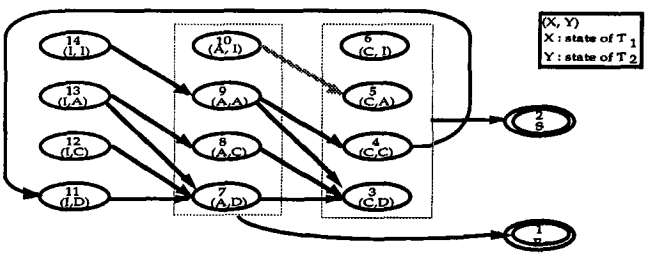

Figure 2: The MC model of the two-task contract with deterministic unit-time state transition

Then, the contractor can compute the transition probabilities of MC states (using the information about the contractees), and the thus utility values. The detailed explanation can be found in (Park, Durfee & Birmingham 1996).

Figure 3 shows the risk-neutral contractor's utility value with different payments of T_2 (ρ_2), assuming the initial state is state 14. Depending on the payment of the other task, the contractor's optimal payment would be different. The best payment of T_1 would be 8.9 when ρ_2 is 10, whereas the best payment would be 8. 5 when ρ_2 is 14.

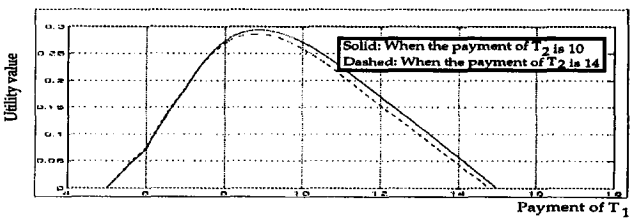

Figure 3: The utility values of the contractor with different payments of the other task

In summary, by modeling the factors that influence the optimal payment, such as the contractees' costs and total capabilities, the communication costs, and the payment of the other task, the contractor is able to find the optimal payment to offer. In fact, it chooses different payments based on the payment of the other task, the communication costs, and different information about the contractee's costs.

Discussion

This paper has defined the contractor's and the contractee's decision problems in contracting situations in the UMDL, and proposed an optimal strategy for the contractor. The utility-maximizing contractor models the future contracting process using Markov chains, and computes the transition probabilities between the states by modeling the other agents and the other contracts. Then, using the MC model, the contractor derives the information needed for computing the utility value and finds an optimal payment.

The contractor's contracting strategy provides a methodology that a utility-maximizing agent can use to find an optimal payment to offer, given (potentially uncertain) knowledge about the possible contractees and the other contracts that have been formed, are being formed, or will be formed among agents. The strategy explicitly takes into account the self-interested contractees and multiple contracts in the system, both of which are primary characteristics of the UMDL.

We have implemented the contractor's contracting strategy using Matlab and performed several analyses for the two-task cases. Early results show that the contractor receives higher payoffs when considering the other contracts in the system and modeling the contractees' decision making.

We are currently developing various MC models depending on the number of tasks being contracted and the amount of information a contractor has. In addition, we will integrate the stand-alone contracting strategy into the UMDL agents.

Acknowledgments

This research has been funded in part by the joint NSF/ARPA/NASA Digital Libraries Initiative under CERA IRI-9411287.

Appendix

In this appendix, we explain the method of computing the probabilities and payoffs of S and F from the absorbing Markov chains.

Before continuing, let's define the Markov chain. Consider a stochastic process $\{X_n, n = 0, 1, 2, \ldots\}$ that takes a finite number of possible values. If $X_n = i$, then the process is said to be in state i at time n. The stochastic process X_n is called a Markov chain if the conditional distribution of X_{n+1} depends only on X_n, the present state, i.e., if

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0\}$$
$$= P\{X_{n+1} = j \mid X_n = i\} = P_{ij} \ .$$

A state i is *transient* iff starting from state i, the process may not eventually return to this state. A state i is *absorbing* iff P_{ii} is 1 (and P_{ij} is 0 for $i \neq j$). A chain, all of whose non-transient states are absorbing, is called an absorbing Markov chain. Figure 1 is an absorbing Markov chain, where I, A, C are transient states, and S and F are absorbing states. From now on, we let T be the set of transient states and let T^C be the set of absorbing states.

The transition probability matrix, $\boldsymbol{P}$, denotes the matrix of transition probabilities P_{ij}. Figure 4-(a) shows the canonical representation of an (r)-state MC consisting of (s) transient states and $(r\text{-}s)$ absorbing states: $\boldsymbol{I}$ is an $(r\text{-}s)\times(r\text{-}s)$ identity matrix (each absorbing state transitions to itself); $\boldsymbol{O}$ consists entirely of 0's (by definition, an absorbing state never transitions to a transient state); $\boldsymbol{Q}$ is an $(s\times s)$ submatrix which concerns the transition only

among the transient states; and R is an $s \times (r-s)$ matrix which concerns the transition from transient to absorbing states. Figure 4-(b) shows the canonical representation of the transition probability matrix of Figure 1 when a single contract is going on.

$$P = \begin{array}{c} \begin{matrix} \overset{(r-s)}{} & \overset{(s)}{} \end{matrix} \\ \left[\begin{array}{c|c} I & O \\ \hline R & Q \end{array}\right] \end{array} \begin{matrix} r-s \\ s \end{matrix}$$

(a) Canonical representation of P

$$\begin{array}{c} \begin{matrix} 4 & 3 & 2 & 1 & 0 \end{matrix} \\ \left[\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ P_{14} & 0 & P_{12} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array} \begin{matrix} 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{matrix}$$

(b) Canonical representation of Figure 1

Figure 4: The canonical form of transition probability matrix

We can use the transition probability matrix to compute the expected number of visits to each MC state, as follows. From the transition probability matrix of any absorbing MC, the inverse of $(I-Q)$ always exists, and

$$(I-Q)^{-1} = I + Q + Q^2 + \ldots = \sum_{k=0}^{\infty} Q^k .$$

In Markov process theory (Bhat 1972), the new matrix $(I-Q)^{-1}$ is called the fundamental matrix, M (i.e., $M = \|\mu_{ij}\| = (I-Q)^{-1}$).

Let N_{ij} be the total number of times that the process visits transient state j from state i. Let N_{ij}^k be defined as 1 if the process is in state j after k steps from state i, and 0 otherwise. Then, the average number of visits to state j from state i before entering any absorbing state is $E[N_{ij}]$.

$$\begin{aligned} E[N_{ij}] &= E[\sum_{k=0}^{\infty} N_{ij}^k] \\ &= \sum_{k=0}^{\infty} E[N_{ij}^k] \\ &= \sum_{k=0}^{\infty} \{(1-P_{ij}^k)\cdot 0 + P_{ij}^k \cdot 1\} \\ &\qquad \text{where } P_{ij}^k \text{ is the } k\text{-step transition probability} \\ &= \sum_{k=0}^{\infty} P_{ij}^k \\ &= \sum_{k=0}^{\infty} Q^k \qquad \text{since } i, j \text{ are transient states} \\ &= (I-Q)^{-1} \\ &= M. \end{aligned}$$

Therefore, the (i,j)-th element of the fundamental matrix, μ_{ij}, is the average number of visits to transient state j starting from state i before the process enters *any* absorbing state. The fundamental matrix is very important, since it is used to compute the needed information (i.e., probabilities and payoffs of S and F) in the following.

A.1 Probabilities of S and F

Let f_{ij} be the probability that the process starting in transient state i ends up in absorbing state j. If the starting state is state 0 in Figure 1, for example, the probabilities of reaching S and F are f_{03} and f_{04}, respectively.

Starting from state i, the process enters absorbing state j in one or more steps. If the transition happens on a single step, the probability f_{ij} is P_{ij}. Otherwise, the process may move either to another absorbing state (in which case it is impossible to reach j), or to a transient state k. In the latter case, we have f_{kj}. Hence, $f_{ij} = P_{ij} + \Sigma_{k \in T} P_{ik} f_{kj}$, which can be written in matrix form as

$$F = R + QF,$$

and thus

$$F = (I-Q)^{-1} R$$

$$F = \|f_{ij}\| = MR \quad i \in T;\ j \in T^C.$$

Therefore, the probabilities of S and F of a contract can be computed using the fundamental matrix (M) and the submatrix (R) of the original transition probability matrix.

A.2 Payoffs of S and F

As defined previously, the payoff of S is $(V - \rho - CC_S + \Delta_S)$, and the payoff of F is $(-CC_F + \Delta_F)$. Here, we present the method of computing the total communication cost and the total retraction penalty $(-CC + \Delta)$ of S and F, given ρ. Let's define the reward matrix Ω, where ω_{ij} represents a reward associated with each transition $i \rightarrow j$. The reward of each transition can be either the minus communication cost $(-cc_{ij})$ or the minus communication cost plus the retraction penalty $(-cc_{ij} + \delta)$.

For the time being, let's assume that we can compute $\mu_{oi}^{(S)}$ and $P_{ij}^{(S)}$, where $\mu_{oi}^{(S)}$ is the number of visits to state i starting from initial state 0 before the process enters S; and $P_{ij}^{(S)}$ is the conditional transition probability when the process ends up in S.

Then, $\sum_{j \in \{T, T^C\}} P_{ij}^{(S)} \cdot \omega_{ij}$ is the average reward of the one-step state transition from state i when the process ends up in S. Multiplying it by $\mu_{oi}^{(S)}$ (the number of visits to state i starting from 0 until it goes to S), we compute the one-step reward accrued from state i when the process ends up in S ($\mu_{oi}^{(S)} \Sigma_j P_{ij}^{(S)} \omega_{ij}$). Adding this value for every state i, we compute the total reward of S. That is, the total reward of S $(-CC_S + \Delta_S)$ can be computed as follows.

$$-CC_S + \Delta_S = \sum_{i \in T} \left(\mu_{oi}^{(S)} \cdot \sum_{j \in \{T, T^C\}} P_{ij}^{(S)} \cdot \omega_{ij} \right).$$

The reward of F $(-CC_F + \Delta_F)$ can be computed in a similar way.

$$-CC_F + \Delta_F = \sum_{i \in T} \left(\mu_{oi}^{(F)} \cdot \sum_{j \in \{T, T^C\}} P_{ij}^{(F)} \cdot \omega_{ij} \right).$$

Now, how do we compute $\mu_{oi}^{(S)}$ and $P_{ij}^{(S)}$ (and $\mu_{oi}^{(F)}$ and $P_{ij}^{(F)}$)? From the original matrix P, we need to create two new Markov chains, each of which has one absorbing state, S and F, respectively.

The new transition probabilities, $P_{ij}^{(S)}$, are the conditional probability that the process goes to state j from state i when the process ends up in S. Let ϕ be the statement "the original MC ends up in state S". Then,

$$P_{ij}^{(S)} = P(i \rightarrow j \mid \phi) = \frac{P((i \rightarrow j) \wedge \phi)}{P(\phi)} = \frac{P(\phi \mid i \rightarrow j) \cdot P(i \rightarrow j)}{P(\phi)} = \frac{f_{js} \cdot P_{ij}}{f_{is}} .$$

The new MC with the single absorbing state S, $P^{(S)}$, is defined as follows.

$$P^{(S)} = \begin{bmatrix} 1 & O \\ R^{(S)} & Q^{(S)} \end{bmatrix},$$

where $R^{(S)}$ is a column vector with $R^{(S)} = \left\{ \frac{P_{is}}{f_{is}} \right\}$, and $Q^{(S)}$ is the matrix with $Q^{(S)} = \left\{ P_{ij}^{(S)} \right\} = \left\{ \frac{f_{js} \cdot P_{ij}}{f_{is}} \right\}$.

From $P^{(S)}$, the contractor can compute the new fundamental matrix $M^{(S)}$, and therefore, $\mu_{oi}^{(S)}$. Of course, $P_{ij}^{(F)}$ and $\mu_{oi}^{(F)}$ are computed in a similar way.

References

Atkins, D. E. et al. 1996. Toward Inquiry-Based Education Through Interacting Software Agents. To appear in *IEEE Computer*. see http://www.computer.org/pubs/computer/dli/r50069/r50069.htm.

Bhat, U. N. 1972. *Elements of Applied Stochastic Processes*. John Wiley & Sons Inc.

Gmytrasiewicz, P. J., and Durfee, E. H. 1993. Reasoning about Other Agents: Philosophy, Theory, and Implementation. In Proceedings of the International Workshop on Distributed Artificial Intelligence, Hidden Valley, PA.

Kraus, S. 1994. Contracting Tasks in Multi-Agent Environments. Technical Report, UMIACS-TR-94-44, Univ. of Maryland.

Kreps, D. M. 1990. *Game Theory and Economic Modeling*. Oxford University Press.

Park, S., Durfee, E. H., and Birmingham, W. P. 1996. Use of Absorbing Markov Chains to Design a Multiagent Contracting Strategy. Forthcoming. see http://ai.eecs.umich.edu/people/boxenju/absorbing-mc.ps

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press.

Rubinstein, A. 1982. Perfect Equilibrium in a Bargaining Model. Econometrica, 50:97-109.

Sandholm, T. 1993. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In Proceedings of the AAAI, Washington, DC

Sandholm, T., and Lesser, V. 1995a. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In Proceedings of the ICMAS.

Sandholm, T., and Lesser, V. R. 1995b. Advantages of a Leveled Commitment Contracting Protocol. Technical Report 95-72, University of Massachusetts.

Sen, S. 1993. Predicting Tradeoffs in Contract-Based Distributed Scheduling. Ph.D. Thesis, U of Michigan.