Trading Risk in Mobile-Agent Computational Markets

Jonathan Bredin, David Kotz, and Daniela Rus
Department of Computer Science
6211 Sudikoff Lab
Dartmouth College
Hanover, NH 03755, USA

Abstract

Mobile-agent systems allow user programs to autonomously relocate from one host site to another. This autonomy provides a powerful, flexible architecture on which to build distributed applications. The asynchronous, decentralized nature of mobile-agent systems makes them flexible, but also hinders their deployment. We argue that a market-based approach where agents buy computational resources from their hosts solves many problems faced by mobile-agent systems.

In our earlier work, we propose a policy for allocating general computational priority among agents posed as a competitive game for which we derive a unique computable Nash equilibrium. Here we improve on our earlier approach by implementing resource guarantees where mobile-agent hosts issue call options on computational resources. Call options allow an agent to reserve and guarantee the cost and time necessary to complete its itinerary before the agent begins execution.

We present an algorithm based upon the binomial options-pricing model that estimates future congestion to allow hosts to evaluate call options; methods for agents to measure the risk associated with their performance and compare their expected utility of competing in the computational spot market with utilizing resource options; and test our theory with simulations to show that option trade reduces variance in agent completion times.

1 Introduction

Faster, cheaper computing hardware and pervasive networking allows more powerful computational abstractions. One abstraction that we promote is a mobile agent, a user thread that may autonomously migrate from one host to another. Mobile code supports flexible architectures that transcend traditional client-server systems [LO99]. It is possible to inject agents offering a new service into the network to replace an existing service without interrupting other services. Collections of mobile agents can better utilize network resources by relocating computation closer to the location of the data and other resources and evenly distributing computation among many hosts. A query agent may relocate to be closer to a disk or an agent operating on a palm-top computer may jump to a more powerful desktop machine to perform large computation or network operations. Other examples of mobile code use are proxy distribution in the Jini service lookup [AOS+99] and complex queries carrying SQL code to be executed on a database server.

While a mobile-agent system's flexibility can speed application development, decentralized control often hinders deployment of mobile-agent applications. Without additional structure it is likely that mobile agents will crowd some resource. Furthermore, existing mobile-agent systems provide no incentive for resource owners to provide access to arbitrary agents.

We promote computational markets where agents purchase their computational resources from hosts and sell services to hosts, users, and other agents [BKR99]. As one would expect, markets expose agents to risk from fluctuating prices and, in this paper, we propose that hosts and agents trade risk through hosts issuing call options on computation. We present a method for a host to price an option and an algorithm for an agent compare the value of a call option with the prospects of competing in the spot market for computation. We demonstrate in simulations that agents are able to avoid volatility in itinerary completion times by using "computational options."

2 Motivation

Mobile-agent systems provide clean abstractions to represent protocols and distribute resource consumption over a network. Mobility delivers design flexibility, but often at the expense of a loss of centralized control. We promote markets as a structure with which to regulate resource allo-

In Proceedings of Sixth International Conference on Computing in Economics and Finance, July 2000.

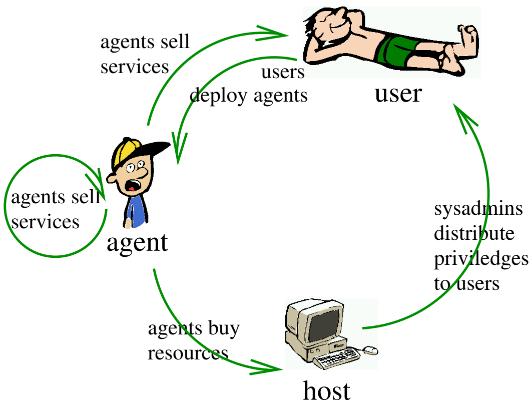

Figure 1. The exchange of currency, services, and resources in a mobile-agent computational market.

cation and consumption in mobile-agent systems. Through prices, markets impose additional constraints upon agents and fluctuating prices can expose an agent to additional risk. We therefore propose that hosts issue reservations for resource consumption to agents and forecast consumption using option-pricing theory.

2.1 Markets

If an agent can assess and is charged for the cost of its actions, then the agent can make rational decisions regarding resource consumption. It is possible that the agent contracts another agent to complete a subtask. The agent's host can collect the charges made to the agent. The host's administrator endows its users with the proceeds, completing the cycle of currency exchange. The exchange of resources, services, and currency establishes a market for computation in the network. Figure 1 shows the cycle of currency exchange.

A user endows its agent with a finite supply of currency that represents the agent's potential to consume resources and load the network. Because the agent's endowment is limited, so is its ability to congest the network. The agent's limitation enforces a form of fault tolerance and prevents errant programs from running amok. An agent that chooses to irresponsibly consume resources curtails its lifetime. The effect is reinforced by the market's correlation between the price and demand of a resource.

The price-congestion link not only promotes fault tolerance, but it serves to increase system utilization through spatial load balancing. An agent visiting a host with a higher price for computation will relocate to a cheaper, less utilized host. If all sites in the network post high prices, then the agent will go dormant to wait until congestion subsides. The latter effect is temporal load balancing.

Finally, markets provide resource owners with incentive to participate in mobile-agent systems. It is to a host's benefit to collect currency from mobile agents so that the host's administrator may redistribute the potential to her mobile-agent using users. If the services mobile agents provide are valuable enough, hosts can exchange currency used by mobile agents for legal tender.

2.2 Market Volatility and Call Options

Markets are not without problems, however. Efficient participation in markets requires agents to have timely information concerning the state of the network and efficient markets may not shield participants from the volatility of the underlying environment. To reduce risk in agents' performance, we investigate issuing call options on computation.

Relaying information on the state of a host consumes network bandwidth and the latency involved in transferring ages the information particularly when resource congestion is high. The aging is aggravated by the fact that there is additional delay between the event when an agent decides to relocate and the event when the agent arrives at its destination. By the time an agent arrives at a site, other agents may have also relocated possibly changing motives causing the agent to relocate.

An agent with a fixed endowment may find part way through its computation that it is not possible to continue at its current rate or, in dire cases, to continue at all. This possibility adds volatility in a user's expectations of its agents' performance or cost if agents request additional currency from the user.

A reservation offering a mobile agent a resource in the future would be valuable to reduce variance in agents' completion times. In this paper we present mechanisms for mobile agents to purchase options guaranteeing the right to consume a resource during a specified time in the future. We represent these reservations as European call options and leverage existing techniques from finance to evaluate options. We modify an existing model of mobile-agent resource allocation to incorporate hosts' ability to distribute options and show simulation results describing the benefits of using options.

In our previous work [BMI$^+$00], we presented a game-theoretic resource allocation for use in closed environments where users collectively own computing resources and host sites do not play active role in pricing resources. We defined a mechanism that allows agents at a site to collectively negotiate the computational priority of each agent at the site, given that each agent has perfect information regarding its own itinerary. Using our policy, there is a unique, computable Nash equilibrium strategy for agents to use to compute their bids and plan their expenditures.

For agents to plan their itineraries, hosts must publish the local cost of computation to prospective agents in the network. With bursty workloads, even in a steady state, the price of computation fluctuates at each host. A host that announces itself to have the lowest price of computation can look forward to entertaining a flood agents desiring service.

The number of agents responding to the host's announcement scales linearly with the number of sources. Additionally, as network delay increases, so does the number of responses. There are two problems here. Both of these issues are substantial. Mobile-agent systems are promoted for their abilities to scale and operate in networked environments with high network latency.

The limitations of naive pricing motivate us to consider systems where hosts, in addition to selling resources on demand, also sell to agents reservations guaranteeing resource access. In this paper, we consider modeling these reservations as European call options [CW92]. A host sells to an agent a call option that guarantees to the agent that during a specified time period, the agent may consume a specified amount of resources for a price specified by the option. We assume that agents are risk averse and are willing to pay a premium for guaranteed performance.

3 Pricing and Purchasing Call Options

We proceed by describing the environment in which agents operate in Section 3.1 and the nature of changes in computational price in Section 3.2. In Section 3.3 we derive an option-pricing algorithm for use by hosts and we construct an algorithm for agents to purchase options in Section 3.4 to be used in conjunction with existing algorithms to participate in the spot market.

3.1 System Model

We begin by outlining an earlier model that defines our underlying spot market for computation [BMI+00]. We consider a network of mobile agents and their hosts. Each mobile agent has a sequence of computational tasks to complete and each task may be completed at one of several hosts in the network. Every mobile agent is endowed with a finite amount of currency with which to purchase computational resources. For a detailed description of the model and derivation of the bidding functions, we encourage the reader to read our earlier work [BMI+00].

We begin with the assumption that an agent may not return remaining currency to users at the end of the agent's lifetime and that the agent's only goal is to complete its computation as quickly as possible given the available resources. This assumption facilitates planning on the part of the agent and operation of the computational spot market. None of the algorithms presented in this paper, however, are dependent on the mechanism driving the spot market or our assumptions concerning an agent's valuation of currency.

We also assume that once an agent migrates to a host, it commits to finishing its immediate task at the host. The agents present at a site competitively negotiate for the price of computation by submitting bidding functions. Each agent receives a portion of the CPU proportional to the rate that the agent pays the host. The time an agent requires to complete a task of size q is:

$$t = \frac{q\theta}{uc}, \tag{1}$$

where u is the agent's rate of payment, θ is the rate at which the host collects payments from agents (including u), and c is the host's capacity, in instructions per second, to process jobs. The cost, e, of execution in the model is the time taken multiplied by the agent's bid rate:

$$e = ut = \frac{q\theta}{c}. \tag{2}$$

Because an agent's share is dependent on the competing bids, we allow agents to condition their bids on the level of competition and agents express their bids as concave functions of, θ, the sum of all bids at the host. When only one agent is present, the price of computation is zero. We derive in our earlier work the optimal bid function for an agent given perfect information concerning the bid totals at all hosts the agent will visit and the agent's job sizes. We also prove that when there are multiple agents at a host, there is unique positive equilibrium allocation that satisfies all agents' bid functions.

We assume that hosts are risk neutral and that all agents are risk averse with utility functions of the form

$$U = -r^{\Sigma_k n_k + t_k}, \tag{3}$$

where $r > 1$ is a parameter describing an agent's preference towards risk, t_k is the time taken to complete the k-th task, and n_k is the network delay incurred in transferring the agent to the k-th site. Since hosts are risk neutral and agents are risk averse it is to each party's benefit to trade risk in the form of computational options. A risk averse agent prefers to pay a higher price for computation at a guaranteed rate rather than take its chances with service with high performance variance.

It is for this reason that we are interested in trading computational options. For now we will restrict ourselves to the set of options that have zero strike price. That is, the options that we will investigate are completely front loaded; agents pay in advance for their computation.

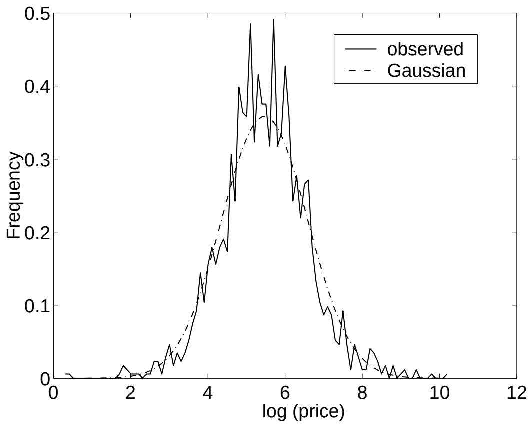

Figure 2. A histogram of the logarithm of the price of computation at a simulated mobile-agent host over time.

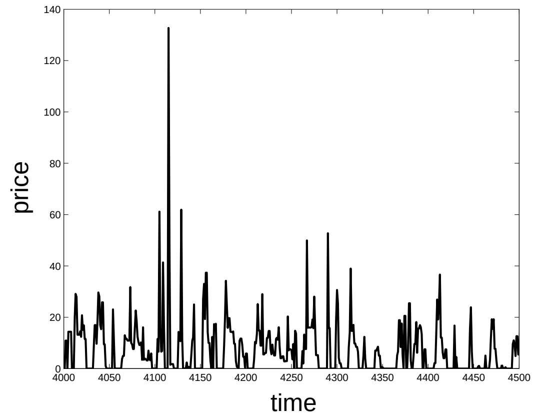

Figure 3. A sample time series of the price of computation at a host in the model.

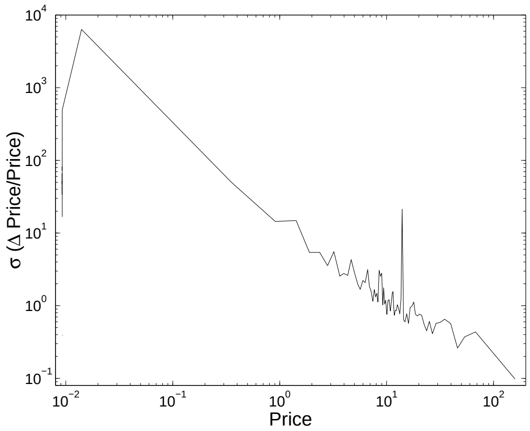

Figure 4. Relative volatility, σ, versus price.

3.2 Price Behavior

A prerequisite for trading options is the ability to forecast the price of the underlying security. We investigate the properties of price fluctuation in our model to derive and forecast of the price agents are willing to pay for computation.

We notice that it is not appropriate to model price movement with relative changes and that the variance in the price of computation is highly dependent on the current price. In simulations, the price of computation at a host is log-normally distributed over time. Figure 2 shows a histogram of the spot price of computation at a host in the absence of an options market. The histogram omits occurrences when the price is zero, but it is not uncommon in a system running at less than 100 percent utilization for computational prices to fall to zero. Figure 3 shows an example time series of the computational spot price of one host in our simulation.

We conclude that price movement cannot be entirely relative. Most option-pricing models assume that price movement is relative to the price, e.g. the price of a security may move up or down at each period by five percent. If we are to model price fluctuation, we must either use absolute price movement or handle the events of prices falling to and rising from zero specially.

Another useful fact in prediction is price volatility. Figure 4 plots the relative volatility of computational prices conditioned on the price. We see a very strong relationship between the price and the magnitude of its next movement. Several traditional pricing models assume that a price volatility is independent of price, so we must consider the relationship.

3.3 The Crash Pricing Model

Cox *et al.* propose the CRR binomial model as a simplified approach to pricing options as an alternative to the Black-Scholes method [CRR79]. They discretize a security's lifetime into T equal length time periods. The security's price either rises by a factor of r or falls by a factor of d with probabilities p_r and $p_d = 1 - p_r$, respectively. The result of using the binomial model is a Markov chain, from which we can compute the probability of prices. In the limit, increasing the number of time periods to model a security's price movement yields the same result as the Black-Scholes model. Figure 5 is an example of a small Markov chain that allows us to estimate the price two time periods into the future.

The basic CRR model assumes that the probabilities and the relative movements of the security price are independent of the security price, but it is easy to see that the parame-

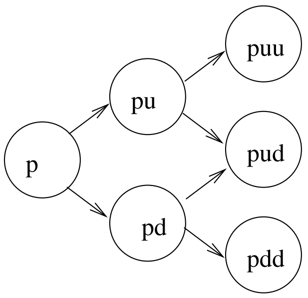

Figure 5. An example of a binomial tree for pricing a security. Each node is labeled with a multiplier for the price of the security.

ters dictating relative price movement could be functions of price. Our pricing model, the Crash Model, builds from the CRR model by adding another event possibility and learning the price behavior of prices conditioned on the current price.

We begin from the standard binomial model and add a new state representing zero price. Each node in the original binary tree has a transition to the new state. From the new state, we have a transition to a state representing a positive price. We analyze price movement history by partitioning the data into groups dependent on price to determine:

- the frequency of the price falling to zero, a crash, from each price group,
- the frequency of and average jump from zero price,
- the frequency and average price increases from each positive price group,
- and the frequency and average decline, excluding crashes, from each positive price group.

The size and ranges of each data group are parameters to the model. Experimentally, we find that dividing the data evenly across the range of prices so that each price range has an equal number of observations works well.

We now describe how we build a Markov chain [Ros97] representing the transitions between states. We label every node in our Markov chain with the pair (θ, t) to represent a possible value for price, θ, at time t. The function $p_c(\theta)$ represents the chance of θ falling to zero in the next time period. Accordingly, each node (θ, t) has a transition to $(0, t+1)$ with probability $p_c(\theta)$. Each node also has transitions to rise in price to node $(r(\theta), t+1)$ with probability $p_r(\theta)$ and descend in price to node $(d(\theta), t+1)$ with probability $1 - p_r(\theta) - p_c(\theta)$.

For θ equal to zero, we compute μ_0 and σ_0^2, the mean and variance of the absolute change in price conditioned the on zero price. The computations of μ_0 and σ_0^2 are straightforward from past observations, $p_c(0) = 0$, and $d(0) = 0$. The remaining parameters to compute are $r(0)$ and $p_r(0)$. We start from the definition of the mean and variance:

$$\mu_0 = p_r(0)r(0) \tag{4}$$

$$\sigma_0^2 = p_r(0)(\mu_0 - r(0))^2 + p_d(0))\mu_0^2. \tag{5}$$

and solve for $r(0)$ and $p_r(0)$:

$$r(0) = \frac{\mu_0^2 + \sigma_0^2}{\mu_0} \tag{6}$$

$$p_r(0) = \frac{\mu_0^2}{\sigma_0^2 + \mu_0^2}. \tag{7}$$

For non-zero price, we compute $p_r(\theta)$ and $p_c(\theta)$ by looking at the proportion of upward, downward, and extreme downward changes from θ. We compute the mean and variance of the relative change in price conditioned on price as $\mu(\theta)$ and $\sigma^2(\theta)$, respectively.

From $p_r(\theta), p_c(\theta), \mu(\theta), \sigma^2(\theta)$, and the definitions of mean and variance, we can compute the upward and downward movements of price. The relative means and variances of the change in price are:

$$\mu(\theta) = p_r(\theta)r(\theta)/\theta + p_d(\theta)d(\theta)/\theta \tag{8}$$

$$\begin{aligned} \sigma^2(\theta) \quad &= p_r(\theta)(\mu(\theta) - r(\theta)/\theta)^2 \\ &+ p_d(\theta)(\mu(\theta) - d(\theta)/\theta)^2 \\ &+ p_c(\theta)\mu(\theta)^2. \end{aligned} \tag{9}$$

We then solve for $r(\theta)$ and $d(\theta)$.

$$r(\theta) = \frac{\theta\mu(\theta)(1 \pm \sqrt{1 + (\frac{p_d(\theta)}{p_r(\theta)} + 1)(1 - \frac{1}{p_d(\theta)} + \frac{\sigma^2(\theta)}{\mu(\theta)^2}))}{p_d(\theta) + p_r(\theta)} \tag{10}$$

$$d(\theta) = \frac{\theta\mu(\theta) - p_r(\theta)r(\theta)}{p_d(\theta)}. \tag{11}$$

It is possible to compute a negative value for $d(\theta)$, so we take the smallest solution to Equation 10 that yields a value greater than $\theta\mu(\theta)$. If $d(\theta)$ is still negative, we assume that all price movement is either upwards or a crash and:

$$p_r(\theta) = \frac{\mu(\sigma)^2}{\sigma^2(\theta) + \mu(\sigma)^2} \tag{12}$$

$$p_c(\theta) = 1 - p_r(\theta) \tag{13}$$

$$r(\theta) = \theta\frac{\sigma^2(\theta) + \mu(\sigma)^2}{\mu(\theta)} \tag{14}$$

$$d(\theta) = p_d(\theta) = 0 \tag{15}$$

This final case assures that $r(\theta) \geq \theta\mu(\theta), p_r(\theta) > 0$ and $p_c(\theta) \geq 0$.

We construct a Markov chain using the transition probabilities and magnitudes, $p_r(\theta), p_d(\theta), p_c(\theta), r(\theta)$, and $d(\theta)$, and represent the Markov chain as a square matrix. Each row of the matrix represents a state, corresponding to a price band, in the chain. The entries within the row denote the probability of jumping to the state represented by the corresponding column. For all but the first row and last rows, there are three non-zero entries. The first and last row have two non-zero entries.

The size of the matrix representing the Markov chain reflects the accuracy and granularity of the prediction. A larger matrix corresponds to conditioning price volatility to larger number of prices. The resulting matrix has the potential to give a finer probability density function for use in pricing, but also requires more data to learn the variance of each price band.

From the Markov chain, we can compute pdf(θ, t, θ'), the probability density function for θ t time periods into the future conditioned on its current value, θ'. The pdf can then tell us the expected loss of revenue at time t of issuing a call option of with a strike price x. We compute the loss of revenue for every period for which the option is valid during $[t_{now} + a, t_{now} + b]$, where t_{now} is the current time. If the market for call options is competitive, this loss of revenue, $C(x, v, t_{now} + a, t_{now} + b, \theta')$, will be equal to the price of an option with strike price x for v of the c instructions per second available from the host, valid from time a until b, conditioned on the current price of computation is θ'. So the value of the option is the expected loss of revenue,

$$C(x, v, t_{now} + a, t_{now} + b, \theta') = \sum_{t=a}^{b} \int_{\theta=cx/v}^{\infty} \left(\frac{\theta v}{c} - x\right) \text{pdf}(\theta, t, \theta') d\theta. \tag{16}$$

The quotient $\theta v/c$ is the amount that the host could have received for $v \leq c$ units of computation per second in the spot market. The integral is over the space $[cx/v, \infty]$ because no rational agent would choose exercise the option if it were cheaper to execute in the spot market.

3.4 The Purchasing Decision

We now investigate how an agent can decide whether or not to purchase reservation. We first discuss why an agent would want to purchase a reservation. This motivation leads us to consider how an agent should decide whether to purchase an option as an alternative to participating in the spot market. We conclude the section by presenting an algorithm that incrementally improves an agent's expected utility by computing the agent's expected utility from participating in the spot market against utility derived from computation backed by reservation.

We begin with the assumption that agents are risk averse and have utility functions of the form of Equation 3. The utility function places more weight on poor performance, so an agent that wishes to maximize expected utility is willing to take some performance degradation in return for a guarantee on quality of service.

An agent needs to be able to compare the expected utility of computing in the spot market with the expected utility of holding a guarantee. To accomplish the comparison, we have hosts in our model publish the mean and standard deviation of the logarithm of the time taken to complete a single unit of computation at the host conditioned on agents' income. We use logarithms, because we empirically observe that the price of computation behaves as a log-normal random variable in Section 3.2.

From the mean and standard deviation of the logarithm of price at a server, the agent can quickly approximate its expected utility and cost through a rough numerical integration as

$$\int_{-\infty}^{\infty} U(t) P(T = t/q) dt, \tag{17}$$

where q is the size of the job at hand, $U(t)$ is the utility derived from executing for time t at a host, and $P(T = t)$ is the probability that the log-normally distributed random variable T yields the value t. The server reports the mean and standard deviation of execution times per job unit executed, so we also normalize the value inside the probability density function in Equation 17. The function $U(t) = -r^{t+n}$, where here n is the latency incurred in jumping to the host, represents the partial utility derived from executing at the host. We justify the function by noting that the derivative of the utility (Equation 3) with respect to execution time at one host in the agent's itinerary is linearly dependent on the execution times at other hosts.

An additional parameter that hosts publish in our model is the mean cost of computation. We can approximate the expected cost of executing an agent's job by examining the expected cost of computation, θ. From equation 2, if the agent's job is small in comparison to competing jobs, the expected cost of execution reduces to $q\theta/c$.

For now, we will only allow an agent to purchase options that it is certain to use. Algorithm 1 describes a greedy process for an agent to choose the next host in its itinerary and to purchase computational reservations. The algorithm locates the host at which it will minimize its completion time with the restriction that the agent competes in the spot market. The budgeting process is defined in our earlier

Algorithm 1 `Choose Next Site for Agent` i

$t \leftarrow$ now()
for all unreserved tasks $k = [1 \ldots]$ **do**
    $u_d \leftarrow u_r \leftarrow -\infty$
    $c \leftarrow 0$
    for all hosts j offering service k **do**
        if $u_d \leq E[U(\text{compute at } j)]$ **then**
            $u_d \leftarrow E[U(\text{compute at } j)]$
            $d_k \leftarrow j$
            $c \leftarrow E[\text{cost of computing at} j]$
        end if
    end for
    for all hosts j offering service k **do**
        if $u_r \leq E[U(\text{compute w/ reservation at } j$ at time t with cost $c)]$ **then**
            $u_r \leftarrow E[U(\text{compute w/ reservation at } j \text{ at time } t)]$
            $r_k \leftarrow$ reservation j sells
        end if
    end for
    if $u_d < u_r$ **then**
        break
    else
        buy r_k
        increment t by period of r_k
    end if
end for
if r_1 **then**
    jump to host honoring r_1
else
    jump to host d_1
end if

work [BMI$^+$00]. The agent then attempts to purchase the fastest option contract with cost equal to the cost of computing at the best spot market. The algorithm continues purchasing options until the expected utility of competing in the spot market exceeds the opportunities to purchase options, or the agent reserves all of its computation.

The algorithm is run each time the agent considers relocating to another host. Our algorithm has no performance bounds or gaurantees. It simply attempts to reserve computation that is at least as perferable to riskier spot-market computation.

4 Simulation

We implement our computational pricing model and option-purchasing decision process on top of our existing simulator [BMI$^+$00]. We model agents as having exponentially distributed number of jobs, with individual jobs Pareto distributed. Agents endowments are normally distributed relative to the sum of their job sizes and we model agents' risk prefereneces, r in Equation 3, as a normally distributed random variable. Hosts capacities are also normally distributed. The distributions of all three normal random variables are truncated to generate only meaningful values.

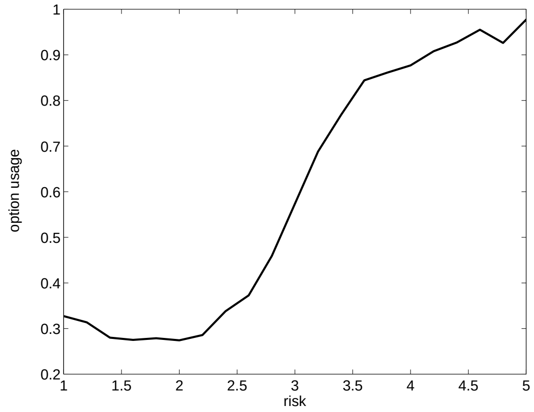

Figure 6. Agent risk versus the average proportion of jobs in agents' itineraries backed by computational options.

Each one of the 100 hosts in the network offers only one of eight services used by agents and constructs its own price model for use in pricing options. Every host constructs an eight by eight matrix to forecast price. Each of the n price bands are represented by the a row in the matrix. Except for the first, each row represents $1/n$-th of the data for prices moving from non-zero values. The first row represents all the price movement away from zero.

Agents attempt to complete a sequence of tasks as quickly as possible given the available resources. Each task that an agent has may be completed at one of eight to nine of the one hundred hosts in the network. Each agent has a fixed endowment to use to complete its set of tasks and uses the algorithms presented in our earlier work [BMI$^+$00] to create its itinerary and plan expenditures.

We would first like to verify that risk-averse agents, ones with high values of r, utilize options. Figure 6 plots the average portion of agents' itineraries that are backed with computational options versus agents' risk parameter. We see that risk-averse agents utilize reservations more frequently.

Figure 7 shows that risk-averse agents are successful in avoiding risk. We plot μ, the mean of the ratio of the actual times taken to complete agents' itineraries and the ideal itinerary completion times in an uncongested network. The line labeled σ represents the standard deviation of completion time and has a downward trend as risk increases, indicating that the volatility in performance is negatively correlated with agents' preference against risk. Being risk averse,

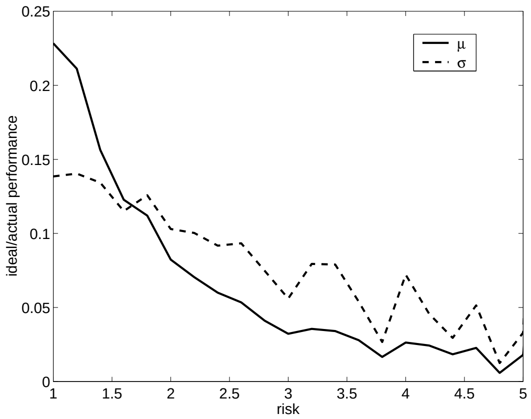

Figure 7. Agents' risk versus the mean and standard deviation performance, μ and σ, repectively. Performance is measured as the ratio of the actual time taken to complete an itinerary and the ideal, uncongested completion time.

however, has a cost. Agents mean performance, represented by the line labeled μ, degrades as their risk parameter increases.

The cost of using options is further illustrated in Figure 8. We plot the the proportion of agents' itineraries backed by computational reservations versus the mean and standard deviation performance. As expected, both metrics are negatively correlated with the use of options. There are few agents that buy options for more than ninety percent of the tasks in their itineraries, so the data points representing these agents are not significant.

5 Related Work

There is a large body of research that leverages microeconomic ideas to increase system utilization in distributed computing environments [KS89, WHH+92, GFS95, CMM97]. These systems consider compute intensive processes that have no ability to migrate once a location is chosen. Regev and Nisan implement distribution of fragmented non-mobile applications over the World Wide Web using computational markets [RN98].

The Geneva Messengers project uses markets to allocate CPU time and memory to mobile agents [Tsc97]. Agents can jointly sponsor areas of memory and prices fluctuate with resource demand. The project does not, however, explore how agents plan in market environments to optimize their performance or operate under budget constraints.

The Xenoservers project provides an operating system with resource usage guarantees as well as the infrastructure

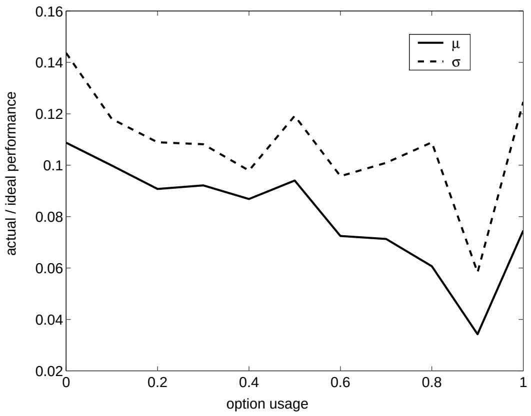

Figure 8. Proportion of itinerary backed by options versus mean and standard deviation performance, μ and σ, repectively.

for assessing costs to applications [RPM+99]. Stratford and Mortier [SM99] present methods for trading resource contracts within market-based operating systems, but do not go so far as to implement pricing algorithms.

Steiglitz *et al.* analyze an synthesized market in which agents participate [SHC96]. The agents produce, trade, and consume two goods, food and gold. The work demonstrates scenarios where agents speculate on the goods yield lower price volatility. It will be interesting to investigate the effects of speculators in our computational markets.

Sandholm *et al.* propose algorithms for calculating the values of *leveled-commitment contracts* [SSN99]. A leveled-commitment contract specifies both the rewards for participants as well as penalties for defaulting. Calculation of rewards and penalties involves full knowledge of participants' utility functions to establish a Nash equilibrium. Our algorithms do not require agents to publicize their utility functions, but leveled-commitment contracts are certainly of interest since their use would allow hosts to over-sell computational options and calculate appropriate amounts to reimburse agents if a contract could not be honored.

6 Conclusions

We apply option-pricing theory to allow mobile-agent hosts to issue reservations gauranteeing computation. We represent a reservation as a restricted type of European call-option and use a modified binomial model to price options that agents substitute for demand consumption. Agents can compare the expected utility of purchasing computation on the spot market and the utility from gauranteed performance to decide whether or not to purchase reservations on computation. Our simulations show that the completion times of risk-averse agents purchasing options are less volatile, but their mean completion time is also lower.

This work exposes several open areas. It would be interesting to explore more general options and better planning algorithms for budgeting agent resources. Much of the work presented in the paper is *ad hoc* and glosses over several topics. We do not consider whether it is in a host's best interest to publish accurate metrics; the reliance of option pricing on a liquid spot-market; or the possibility of secondary option trade among agents. Finally, it will be enlightening to implement computational reservations in the context of a working mobile-agent system.

We examine a restricted subset of European call options in this paper. The purchase price of an option reflects all of its value and we do not consider options with positive strike price. In part, we consider fully front loaded options to eliminate another dimension in agent planning, but without interest rates, there is nothing to be gained by a conservative agent holding an option with a positive strike price.

An option with positive strike price would be valuable in many other applications, however. An agent may wish to minimize response time of infrequent calculations. Such an agent would benefit from purchasing many options discounted to reflect a higher strike price. The use of more general options necesitates more involved agent planning and host pricing algorithms, possibly allowing hosts to oversell their resources in much the same way airlines overbook seating.

There are other issues concerning hosts. We ignore, in this paper, the incentives for hosts to publish accurate parameters describing agent-performance history. It is possible that an implementation would rely on a third-party service tracking host performance. Another role for third parties would be to trade options, possibly smoothing price volatility through the quest for arbitrage opportunities. Along with looking at the effects of secondary option trade, it is important to consider the role of the spot market. In our current pricing model, option trade replaces a portion of the spot trade, but our method fails to price options in the absence of a spot market.

Finally, this work would not be useful if it did not have applications. An implementation will be interesting, especially since accurate descions on the part of agents require accurate priors on the distributions characterizing congestion. In running our simulations we found that the option trading agents' performance is sensitive to the accuracy of their performance forecasts.

The financial option abstraction is powerful. We show that it is useful for reserving computation time and lowering the variance of mobile agents' execution times. The same pricing model could be applied to other resources to predict congestion and guarantee quality of service.

Acknowledgements

This work is supported by Department of Defense contract MURI F49620-97-1-0382 and DARPA contract F30602-98-2-0107. We would also like to thank Andrew Samwick for his insight and advice for representing resource reservations as call options.

References

[AOS+99] Ken Arnold, Bryan Osullivan, Robert W. Scheifler, Jim Waldo, Ann Wollrath, and Bryan O'Sullivan. *The Jini Specification*. Addison-Wesley, Boston, MA, 1999.

[BKR99] Jonathan Bredin, David Kotz, and Daniela Rus. Economic markets as a means of open mobile-agent systems. In *Proceedings of the Workshop "Mobile Agents in the Context of Competition and Cooperation (MAC3)" at Autonomous Agents '99*, May 1999.

[BMI+00] Jonathan Bredin, Rajiv T. Maheswaran, Cagri Imer, Tamer Başar, David Kotz, and Daniela Rus. A game-theoretic formulation of multi-agent resource allocation. In *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, June 2000. To appear.

[CMM97] Anthony Chavez, Alexandros Moukas, and Pattie Maes. Challenger: A multiagent system for distributed resource allocation. In *Proceedings of the First International Conference on Autonomous Agents*, Marina Del Ray, CA, 1997. ACM Press.

[CRR79] John C. Cox, Stephen A. Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7:229–263, 1979.

[CW92] Thomas E. Copeland and J. Fred Westion. *Financial Theory and Corporate Policy*. Addison-Wesley, Reading, MA, third edition, 1992.

[GFS95] Ross A. Gagliano, Martin D. Fraser, and Mark E. Shaefer. Auction allocatio of computer resources. *Communications of the ACM*, 38(6):88–102, June 1995.

[KS89] James F. Kurose and Rahul Simha. A microeconomic approach to decentralized resource sharing in distributed systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.

[LO99] Danny B. Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, March 1999.

[RN98] Ori Regev and Noam Nisan. The POPCORN market— an online market for computational resources. In *Proceedings of the First International Conference on Information and Computation Economies*, pages 148–157, Charleston, SC, October 1998. ACM Press.

[Ros97] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, San Diego, CA, sixth edition, 1997.

[RPM+99] Dickon Reed, Ian Pratt, Paul Menage, Stephen Early, and Neil Stratford. Xenoservers: Accountable execution of untrusted programs. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*. IEEE Computer Society, March 1999.

[SHC96] Ken Steiglitz, Michael L. Honig, and Leonard M. Cohen. A computational market model based on individual action. In Scott H. Clearwater, editor, *Market-Based Control*, chapter 1, pages 1–27. World Scientific, Singapore, 1996.

[SM99] Neil Stratford and Richard Mortier. An economic approach to adaptive resource management. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*. IEEE Computer Society, March 1999.

[SSN99] Tuomas Sandholm, Sandeep Sikka, and Samphel Norden. Algorithms for optimizing leveled commitment contracts. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 535–540, Stockholm, Sweden, 1999.

[Tsc97] Christian F. Tschudin. Open resource allocation for mobile code. In *Proceedings of The First Workshop on Mobile Agents*, pages 186–197, Berlin, April 1997.

[WHH+92] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, February 1992.