Multiagent Workflow Enactment Using Adaptive Pricing Mechanisms

Hrishikesh J. Goradia and **José M. Vidal**

University of South Carolina
Computer Science and Engineering
Columbia, SC 29208, USA
{goradia,vidal}@sc.edu

Abstract

We study the problem of distributed workflow enactment in which new job requests, each composed of a workflow, a deadline, and a payment, arrive at a company at regular intervals. The company must decide which services to perform in which workflows and with which service agents. It must also provide the proper monetary incentives to its selfish service agents so as to align their interests with those of the company. In this scenario we evaluate various pricing strategies and show that an adaptive pricing mechanism is required because it is a dominant strategy and it increases revenue.

Introduction

Workflow languages such as the Business Process Execution Language for Web Services (BPEL4WS) are becoming the preferred method for specifying web services workflows. These languages allow companies to state the order in which web services are to be performed so as to achieve a particular business goal. However, current enactment tools employ a centralized architecture that assumes all clients are slaves. They consist of a workflow enactment engine who reads the workflows and invokes the necessary web services on the client machines who must perform as ordered. In contrast to this traditional view, we envision a much more flexible architecture in which the clients are autonomous agents that decide, as a group, which workflows to enact (Vidal, Buhler, & Stahl 2004).

For example, imagine a small company that generates revenue by handling projects requested by clients. Like in most small companies, the employees are versatile and can carry out any of the many tasks required to complete a project. The employees also want to maximize their salaries. The clients pay the company only when the project is finished by the requested deadline. Thus, any work done on a project that was not completed by the deadline is wasted. We must then determine how the company's management rewards employees for their work and how the employees decide what work to do at any particular time. This scenario can be translated to a distributed workflow enactment problem by considering the project to be workflow requests with dead-

lines and the employees to be autonomous agents that encapsulate web services.

Our goal is to develop scheduling mechanism that maximize a company's revenue even in the presence of selfish agents/employees and uncertainty about future job requests. The mechanism should provide the right incentives to the agents so that their selfish actions are aligned with the company's goal of profit maximization. This distributed mechanism has several advantages over the traditional centralized workflow enactment systems.

- There is no central point of failure and no bottleneck.

- It should scale well as new agents are added or removed.

- Workflows spanning across multiple business units can face privacy and security concerns. Distributed solutions naturally resolve these issues by partitioning and allocating workflows to the concerned units.

- Agents and humans can cooperate. Since the system already works for selfish agents there is nothing a selfish human can do to break it.

Note that we assume that the set of all possible workflows is known a priori. As such, there is no need for the agents to use AI planning techniques to create new workflow. Instead, the problem becomes one of workflow (plan) scheduling and mechanism design under uncertainty.

Related Work

This research builds on our previous work on distributed workflow enactment (Vidal, Buhler, & Stahl 2004; Buhler & Vidal 2005; Buhler & Vidal 2004; Buhler, Vidal, & Verhagen 2003). While that work established the feasibility of enacting workflows using cooperative agents, in this paper we drop the cooperative assumption and instead focus on selfish agents and on workflows that have deadlines and payments associated with them. Given the uncertainty in the environment our agents will need to use learning techniques. The techniques we use are extensions on standard multiagent learning research (Vidal 2003).

The idea of using agents to enact business workflows is not new. For example, in (Muth *et al.* 1998) the authors present a technique for distributed workflow enactment using activity charts. (Singh & Huhns 1999) introduces the idea of interaction-oriented programming as a technique for

engineering multiagent systems that enact workflows. The ADEPT system (Jennings *et al.* 2000) implemented agent-based process management for British Telecom. However, these along with most other distributed workflow enactment techniques assume cooperative agents/web services.

The pricing techniques we use here were inspired by those in (Saha & Sen 2005) but differ from those in many respects, mostly due to the fact that they studied supply chain pricing while we are interested in distributed workflow enactment.

The Distributed Workflow Enactment Problem

Formally, we define a distributed workflow enactment problem as consisting of a set of agents A, a set of services S, and a set of workflows W. Each workflow $w \in W$ is defined as a sequence of services so that $w = (s_1, s_2, \ldots, s_{|w|})$. Agent set $A = A_s \cup m$, where A_s is the set of *service agents* and m is the *manager agent* in the system. Each agent $a \in A_s$ can perform a set of services given by $a_s \subseteq S$. New jobs arrive at the company randomly at each time step and are managed by the manager agent m who is responsible for allocating the service requests to agents and for paying the agents. Each job j has a workflow j_w associated with it, a deadline j_d by which the job must be finished, a time j_t at which it appears, and a payment j_p which will be given to the company if the job j is performed by its deadline j_d. We assume that time is discrete and that each service takes one time unit to execute.

The company maximizes its profit when the highest-priced and shortest workflows are completed within their deadlines, and when the slack time for all agents is the least. As such, the manager should prioritize services such that it maximizes the company's revenue over time. In this paper we compare several such schemes in order to determine their relative merit. Note that the manager agent does not know which job requests will appear in the future, therefore, finding an optimal solution to this problem is impossible. The best the manager can do is to use heuristics, preferably adaptable heuristics, in order to prioritize the services. We also have to be careful and make sure that the manager's payment scheme is not exploitable by the agents. For example, if the manager decides to pay more for some services than other and offers the lower-priced services first then the agents would want to reject the initial offers in order to get the higher paying services later (of course, this strategy might backfire if all the agents did the same).

Service Allocation Mechanisms

In our mechanisms, we use the contract net protocol (Smith 1981) to allocate services to agents as follows:

1. The manager determines the candidate services for allocation across all available jobs for each time step and orders them as described in the next section.

2. The manager then sends out a request for bids for the service with the highest priority from the set of available jobs.

3. The agents bid on the service.

4. The manager chooses the lowest bid and allocates the service to that agent if the bid is below the manager's reservation price for that service.

5. The service's job is removed from the list of available jobs and we go back to step 2. This process is repeated until either all jobs are allocated or all agents are busy.

We now present the various strategies that can be adopted by the manager and the service agents under different circumstances in our system.

Manager agent strategies

The manager sets a reservation price for each service suitable for execution in the workflows across all active jobs. This is the maximum price that it is willing to pay an agent to get the concerned service performed.

More formally, the manager has a set of jobs J that are being executed at any given instant in time t. For each $j \in J$, it's corresponding workflow j_w is at some stage in its execution and needs to execute its next service s_{j_w}. Let Δ^t be the set of all *available* services at time t, that is, all services that are ready to execute. The manager examines each service $s_i \in \Delta^t$ and assigns it a reservation price, as described in the section. The set Δ^t is then sorted by this reservation price, with the highest price first. The manager then tries to allocate the services in Δ^t in order.

The company can maximize its profit by paying as little as possible to the agents for executing the jobs. One option is for the manager to adopt the **fixed strategy (FP)** where it sets its reservation price to a very small, fixed amount, say ϵ, for all services. In this case, the rational strategy for all agents is to accept that price because their only other alternative is to reject it in which case they do not get any payment. This is true for any fixed price that the manager might choose to provide, such as $j_p/|j_w|$. This strategy is simple to implement but fails to properly prioritize jobs. As such, we expect that it will result in forfeited revenue. For example, under these fixed price schemes a service for a job request that has a deadline of $t + 1$ will be given the same priority (price) as that of a job request with deadline of $t + 100$.

Alternatively, the manager can determine the reservation price for a service dynamically at each time step by considering certain prevalent environmental characteristics such as the expected utility from finishing the job, the deadline for the job, and the current stage of execution in the workflow. However, the manager's payment scheme should not be exploitable by the agents. For example, if the manager decides to pay more for some services than others and offers the lower-priced services first then the agents would want to reject the initial offers in order to get the higher paying services later (of course, this strategy might backfire if all the agents did the same). Based on these adaptation techniques we devised two more manager pricing strategies.

The manager can use the **variable strategy (VP)** to set the reservation price for services. Here, the reservation price s_{rp} for service s is computed at its release time based on the expected payment j_p from the job, the amount j_a^t already paid by the manager to other agents for doing previous services in the workflow j_w until time t, and the number of services

yet to be enacted in the workflow, which can be given by $|j_w| - j_n^t$, where j_n^t is the index of the next service in j_w that the manager must enact at time t. The reservation price can be computed as,

$$s_{rp} = \frac{j_p - j_a^t}{|j_w| - j_n^t} \qquad (1)$$

The computed reservation price s_r is then fixed. That is, if the service fails to be allocated in the current round it will again have the same reservation price in the next round. Unfortunately, this strategy also has its flaws. Consider a case where the job length is 5, of which 4 services have already been allocated for a total cost of 6 for the company. The job yields a profit of 10 on completion. In this case, the reservation price for the final service will be set to 4 and will never change even though the company can afford to raise it up to 10 without incurring a heavier net loss for the job.

We also define an **adaptive strategy (AP)** for the manager agent. In this scheme the price is set to a very small amount ϵ at the service's release time s_t but this price is incremented by a constant factor α for each subsequent time step until it is allocated to some agent or it reaches the maximum possible allocation for the service beyond which the company potentially incurs a loss at performing the job. At any given instant in time t, the minimum number of time steps required to complete a job j can be given by $|j_w| - j_n^t$, while the maximum number of time steps available to finish the job is $j_d - t$. Thus, a very rough estimate of the probability that the company will finish the job j before its deadline j_d is given by

$$j_s^t = \max\left(1 - \frac{|j_w| - j_n^t}{j_d - t}, 0\right). \qquad (2)$$

Using this value we can approximate the company's expected income j_{ei}^t for the job j at time t as

$$j_{ei}^t = j_p \cdot j_s^t. \qquad (3)$$

The manager might also want to prioritize a service if it is a part of a high-value job, and the company's estimated success for completing the job within its deadline is getting precariously low. This can be done by presenting the company's best offer for the service. The best offer s_{bo}^t that the manager might want to present the agents for a service s at time t can be given as,

$$s_{bo}^t = \max\left(\frac{j_{ei}^t - j_a^t}{|j_w| - j_n^t}, 0\right) \qquad (4)$$

From the above equations, we can create the manager's strategy for assigning the reservation price s_{rp}^t for a service s at time t as follows:

$$s_{rp}^t = \begin{cases} \epsilon & \text{if } (t = s_t) \\ s_{rp}^{t-1} + \alpha & \text{else if } ((s_{rp}^{t-1} + \alpha < s_{bo}^t) \wedge (j_s^t > \beta)) \\ s_{bo}^t & \text{otherwise.} \end{cases} \qquad (5)$$

for some constant α. That is, if the service has just become available then we start with a price of ϵ otherwise

while the estimate of the probability of getting the job finished in time is greater than β we keep increasing the price by α. If the probability for finishing ever dips below β then we use the best offer price. We also make sure that the price never exceeds the best offer. The β value implements a risk-averse behavior where the company assigns more value to less risky jobs. Setting $\beta = 0$ results in a risk-neutral (therefore rational) manager agent.

Service agent strategies

Since the agents perform information services and the marginal cost of reproduction for such services is zero, we assume that they set a reservation price of zero for performing their services. That is, it does not cost an agent anything to perform a service.

Given that the service agents are selfish we can expect them to adapt their bidding behavior so as to maximize their profit. Specifically, it is in their best interest to use adaptive bidding strategies that exploit prevalent environmental conditions by raising their quotes when the competition for securing services is less, and lowering them as competition increases. We define three adaptive bidding strategies for the service agents. In all three strategies, the agents raise their bidding price for a particular service if they have won the previous auction for that service and lower their quote up to their reservation price if they were unsuccessful in securing the previous contract for the service.

The agents can adopt the **linear strategy (LB)** where if they win the previous contract for a service then they increase their bid for that service by a constant γ from the previous bid. On the other hand, if they lose the previous contract, then they decrease their next bid for that service by γ as long as this new bid is higher than their reservation price for the service. Formally, the new bidding price set by an agent using the linear strategy will be,

$$bid = \begin{cases} b + \gamma & \text{If it won the last contract} \\ & \text{for the current service,} \\ \max(b - \gamma, r_s) & \text{otherwise.} \end{cases} \qquad (6)$$

where, b is the agent's bid for the previous contract and r_s is its reservation price for the service s.

The agents might want to be more aggressive in their bidding and exploiting the current conditions by adopting the **impatient strategy (IB)**. Here, the agents vary their bid sharply after winning or losing contracts. Say an agent has won or lost k successive contracts for some service s. The agent's next bid for the same service will be,

$$bid = \begin{cases} b + (\gamma + (k \cdot \delta)) & \begin{array}{l}\text{If it won the last} \\ k \text{ contracts for the} \\ \text{current service} \end{array} \\ \max(b - (\gamma + (k \cdot \delta)), r_s) & \begin{array}{l}\text{If it lost the last} \\ k \text{ contracts for the} \\ \text{current service} \end{array} \end{cases} \qquad (7)$$

where δ is some constant. The value for k is reset to zero when the agent wins after losing (or vice-versa) the previous bid for a service.

Alternatively, the agents can also use a **defensive strategy (DB)** for bidding. In this case, the agents are more cautious while setting their future bids. If an agent wins or loses k successive bids for a service s, then it's next bid for s will be,

$$bid = \begin{cases} b + \max\left((\gamma - (k \cdot \delta)), 0\right) & \text{If it won the last } k \text{ contracts for the current service} \\ \max\left(b - (\gamma - (k \cdot \delta)), r_s\right) & \text{If it lost the last } k \text{ contracts for the current service} \end{cases} \quad (8)$$

Performance Metrics

Our goal is to determine which combination of strategies would work better for a company and what are the relative tradeoffs in the various combinations. However, determining what is best is not as simple as maximizing revenue because we must also considers issues of solution stability and fairness. As such, we use the following performance metrics for evaluating the different algorithms:

Completed workflows This metric determines the number of workflows that the agents decided to invest in and succeeded in completing. In our context, this is a very important property as businesses would want to fulfill the promises they make while accepting the contracts.

Company revenue This represents the total revenue that the company generates by performing the different workflows. This metric gives us insights on the choices made by the algorithms about which workflows to perform. Clearly, this is one of the most important metrics in our context as businesses always strive to maximize their profit.

Agent profit This is the amount that the service agents get paid by the manager agent. The agents prefer the bidding policies that maximize their profits irrespective of whether they are in the best interests of the company or not.

Company profit This is the company revenue minus the agent profit. That is, it is the amount that is left over after paying the service agents. The company might prefer the pricing schemes that maximize the company's profit as long as they do not affect the total revenue that is generated.

Wasted effort We define wasted effort as the number of services performed by the agents for workflows that were not completed successfully.

Load distribution among agents This measures the number of services performed by the various agents in the system over a period of time. A mechanism that distributes this load evenly among all agents is more preferable as it will scale better.

Revenue distribution among agents This measures the distribution of the total profit generated for the agents.

Experimental Framework

We implemented our simulation software using NetLogo (Wilensky 1999), a programmable modeling environment particularly suited for modeling complex systems that develop over time. Our experiments evaluate the relative effectiveness of the different pricing and bidding strategies under varying environmental conditions, from market settings to monopolistic settings.

We define a set of 5 services for our tests. Each workflow is a sequence of services selected from this set. We use a Poisson distribution function for generating new workflows in the system. The workflows are selected from a predefined but randomly generated set of service sequences. Each workflow has an associated profit value that the company gains at its completion, and a deadline that has a small chance of being very close to the release time. We define 5 agents that are capable of performing a certain set of services, and assume that each service can be performed in one time step without any chance of failure. We set the environment parameters such that there are around 50 live workflows for the 5 agents to consider and vary the capabilities of individual agents to study the performance of the different algorithms.

In our simulations, we set a pricing scheme for the company and a bidding strategy for all the agents, and then run the system for 300 cycles. We test all nine possible pairings. To ensure the correctness of our results that compare the various strategies, the same set of randomly generated data is passed to all the pricing-bidding strategy pairings during each simulation. In each cycle, we randomly generate some workflows (based on the workflow arrival rate set for the simulation), determine the candidate services for allocation, rank these services, and auction off the best ones to the agents. For each of the bidding strategies, we allow the agents to explore and randomly generate a bid 5 percent of the time. We have averaged our results over 50 simulations for each experiment.

Test Results

Our experiments are aimed at ascertaining the relative merits of the different pricing and bidding schemes presented in this paper for the workflow scheduling problem. We represent the fixed, variable, and adaptive pricing strategies used by the manager as FP, VP and AP respectively. The different bidding policies adopted by the agents, linear, impatient, and defensive, are represented as LB, IB, and DB respectively. Thus, strategy pair AP_DB in our results refers to a system where the manager uses adaptive pricing and the service agents bid using the defensive strategy.

Market System

We start by allowing all agents to perform all the services, so the manager has the option to substitute one agent with any other agent to control prices. Figures 1, 2 and 3 show the performance of the various strategy pairs in this setting.

We notice that the FP strategy does not perform well for any performance metric used to evaluate our system. This is because the agents are performing services randomly since
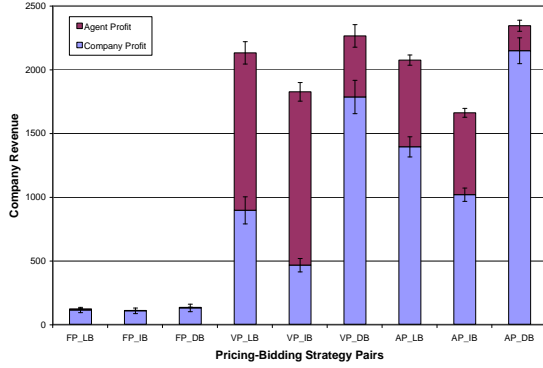
Figure 1: Company revenue for various strategies for market settings. The error bars represent one standard deviation in the results.
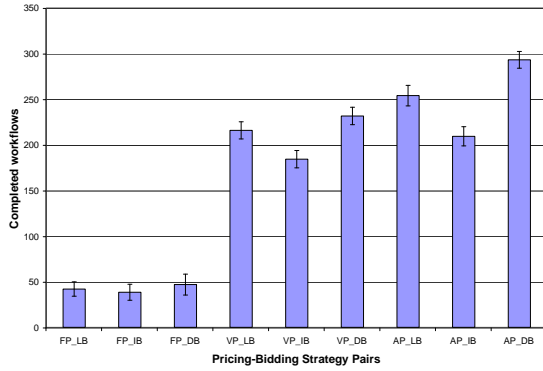

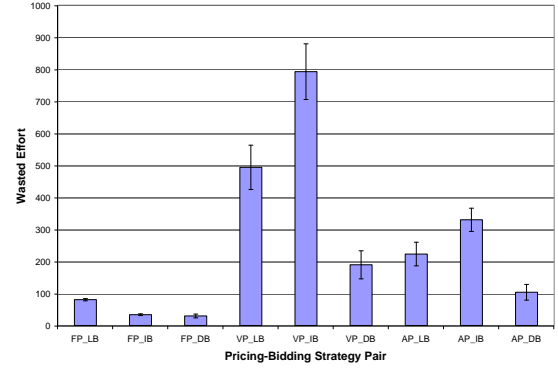Figure 2: Completed workflows for various strategies for market settings.

Figure 3: Wasted effort for various strategies for market settings.

Figure 4: Load distribution between agents for various strategies over all experiments.

the company offers identical rewards for all services. The end result is that although the agents perform almost identical number of services for all pricing strategies, most of the effort is wasted as very few workflows manage to finish all their services within their deadlines. Consequently, the total revenue generated for the fixed strategy is very small.

AP has a better workflow completion ratio than VP for every bidding scheme that an agent chooses. This suggests that the adaptive scheme provides better incentives to the agents for selecting the services belonging to workflows that are nearing completion. However, we notice that finishing more workflows does not translate into proportionally more revenue for the company. In fact, we notice that in spite of finishing fewer workflows, the VP strategy performs nearly as well as the AP scheme when agents use DB and comfortably outperforms the AP schemes when the agents use LB or IB. This suggests a preference for smaller workflows as they are most likely to finish and generate some revenue, in spite of them being low paying, by the AP strategy. On the other hand, the VP strategy adopts a more liberal approach and prefers high paying workflows. This approach also results in the agents making a significantly higher profit when the company adopts the VP strategy as compared to AP. The side effect of the VP strategy is that it also leads to a lot of wasteful service execution. By contrast, the AP strategy results in extremely low wasted effort.

Considering the bidding strategies, the DB strategy con-

sistently generates more revenue and successfully finishes more workflows for the company than any other strategy, irrespective of the pricing scheme selected by the manager. When all agents are defensive in their bidding they are less aggressive in increasing or decreasing their bid, and this patient approach rakes in more profit for the company in the long run. It sustains the phase where agent bids are at or around the reservation prices for services as set by the manager, thereby facilitating many successful auctions for those services in succession. Unfortunately, this approach does not allow the agents to exploit the situations when little or no competition exists by bidding highly. Consequently, they make significantly lower profit with DB as compared to LB or IB.

The load and revenue distributions for the system are plotted in figures 4 and 5 respectively. The rightmost cluster in both graphs represents the market system where all agents can provide all services. It shows the difference between the agents receiving the maximum and the minimum values in the distribution. Since all agents are homogeneous in every aspect, the variance in load and revenue distribution between agents is minimal for true markets. Figure 1 shows the share of one agent (agent 1) from the total generated profit for all agents.

In Figure 6 we have placed the company's and the agents' profits within a game matrix. Each cell in the matrix defines the utilities gained by the agents and the company for their respective choices of bidding and pricing strategies in the
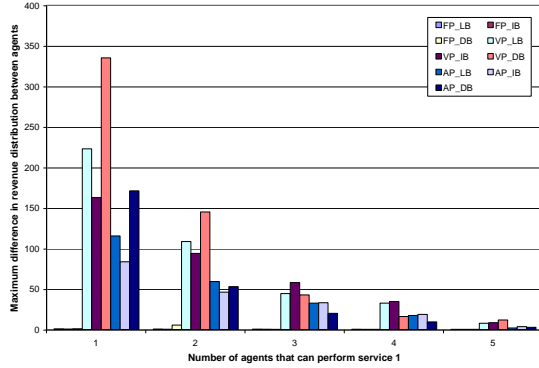
Figure 5: Revenue Distribution between agents for various strategies over all experiments.

		Manager		
		FP	VP	AP
	LB	$9, 111$	$1235, 893$	$680, 1391$
Agents	IB	$4, 105$	$1359, 463$	$642, 1015$
	DB	$5, 128$	$479, 1782$	$196, 2145$

Figure 6: Game theoretic analysis of the system for market system.

given setting. We see that AP is the dominant strategy for the manager in this context and, given that choice for the manager, the rational choice for the agents would be LB. Thus, AP_LB is the dominant solution in the above setting. However, the social welfare solution for the current settings is AP_DB.

Monopolistic System

We now test the performance of our system in monopolistic settings, where there is exactly one agent (agent 1) in the environment that can perform a particular service (service 1). In this case, the manager is at a loss against agent 1 as there is no substitute to this agent for service 1. Agent 1 can exploit this situation and maximize its own profit. We now study the characteristics of the system under such situations. The results are shown in figures 7, 8 and 9.

As can be expected, the system-wide performance drops in a monopolistic setting. The company completes fewer
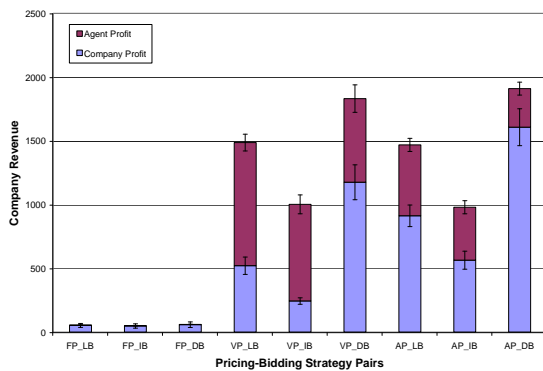

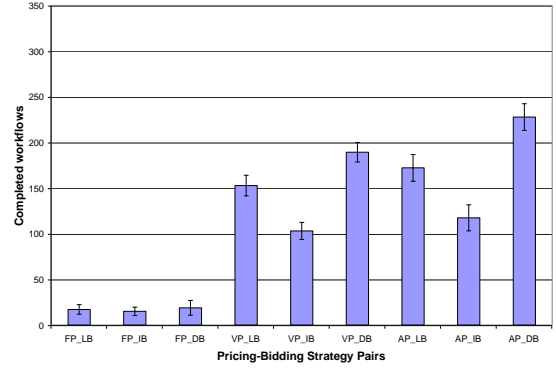Figure 7: Company revenue for various strategies for monopolistic settings.

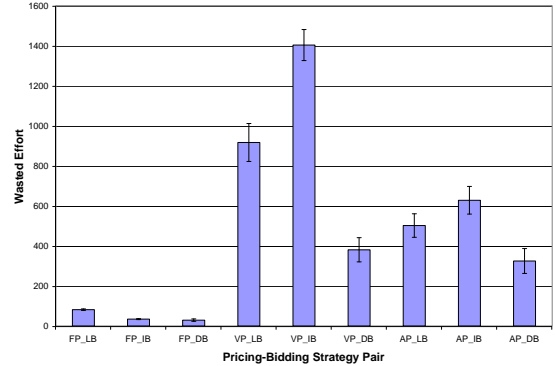Figure 8: Completed workflows for various strategies for monopolistic settings.

Figure 9: Wasted effort for various strategies for monopolistic settings.

workflows and generates less revenue while the agents indulge in a high wasted effort and yield less profit. The only agent that gains from the monopolistic setting is the monopolist. However, the relative performance of the various strategy pairs for the monopolistic setting is still the same as that for the market system.

The biggest difference between monopolistic and market settings is in the load and revenue distribution, as shown in figures 4 and 5 respectively. Figure 7 shows agent 1's share from the total agent profit. Since one agent carries the workload of performing service 1 for all the workflows it is, as expected, busier than the other agents. Even though the services are selected randomly, the proportion of available service 1 services keeps increasing over time and so the agent's probability of being selected for execution also increases. However, we see that agent 1 does not gain significant revenue from performing more services in FP as the compensation for performing the services is much smaller. With VP, the aggressive approach of selecting the highest paying services at each time step results in service 1 being selected for execution only if it pays higher than the others. Since all services in a workflow yield equal compensation in VP, the preference for service 1 is only slightly higher than other services due its larger presence in the available services population. So, the load on agent 1 is not significantly higher than in a true market. However, agent 1 plays a role in most workflows that are completed and so its share of the revenue is very high. The AP scheme allows the company

		Manager		
Agents		FP	VP	AP
	LB	$4, 52$	$966, 520$	$556, 911$
	IB	$2, 47$	$758, 243$	$415, 563$
	DB	$3, 57$	$656, 1174$	$303, 1606$

Figure 10: Game theoretic analysis of the system for monopolistic settings.
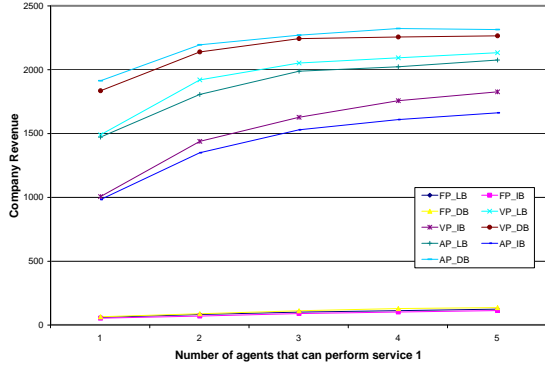

Figure 11: Company revenue for various strategies over all experiments.

to be very selective in its choice of workflows and invest only in those workflows that are most likely to finish. Under such situations, the agents are not functioning optimally most of the time. While functional, the common services are shared uniformly by all the agents, but all the service 1 calls go to agent 1. Consequently, the variance in the distribution of load is large for this scheme, with agent 1 performing a lot more services than the other agents. Agent 1 also accrues more profit than the other agents in this scheme but it is not as pronounced as with the VP scheme. The reason for this relatively low revenue gain for agent 1 is that with AP the agents perform fewer services than with VP. Also, the performed AP services pay lower than those with VP. Considering the results from the perspective of agent choices regarding their bidding strategy, we find that IB produces greater profit for the agents than LB, which in turn is better for the agents than DB. So, agent 1 wins more than other agents in that same order.

An analysis of the monopolistic setting from the game theoretic perspective, shown in figure 10, shows that here too the dominant strategy for this scenario is AP_LB, while the social welfare strategy is AP_DB.

All Systems

In this section, we show the results from all five different sets of experiments with the environment settings ranging from monopolistic to market system. For every pricing-bidding strategy pair in each experiment. Figure 11 shows the total revenue generated by the company which, as we might expect, drops as the system goes from a market system to a monopoly. Similarly, figure 12 shows the total profit made by all agents and we can see that their profit also drops as the system becomes a monopoly. It is interesting to note, however, that some the lines cross which implies that certain
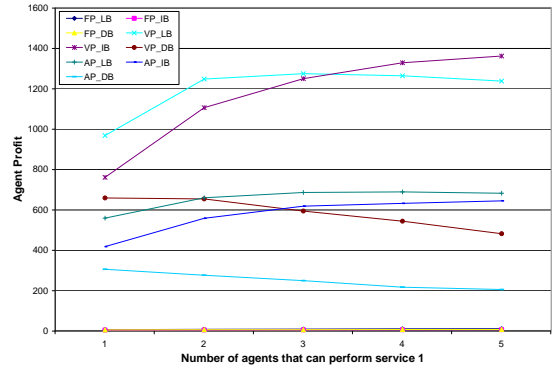

Figure 12: Agent profit for various strategies over all experiments.
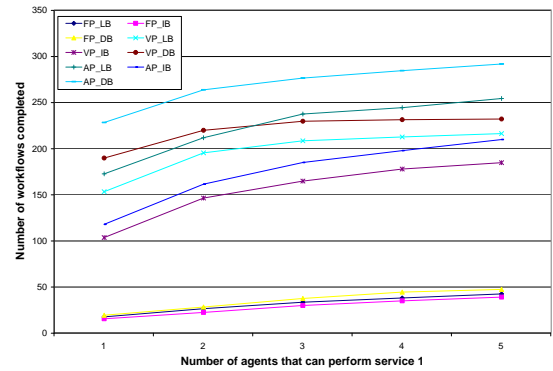

Figure 13: Completed workflows for various strategies over all experiments.

strategies have an advantage in a market system and others in a monopoly. Figures 13 and 14 continue the expected trend showing that the number of completed workflows increases in a market system while the wasted effort decreases.

Figures 15, 16 and 17 show the game matrices for the other experiments. For these scenarios as well we find that the dominant solution is AP_LB, while the social welfare solution is AP_DB.

Conclusions and Future Work

We have presented our adaptive market-based solution for the distributed workflow scheduling problem. We compared
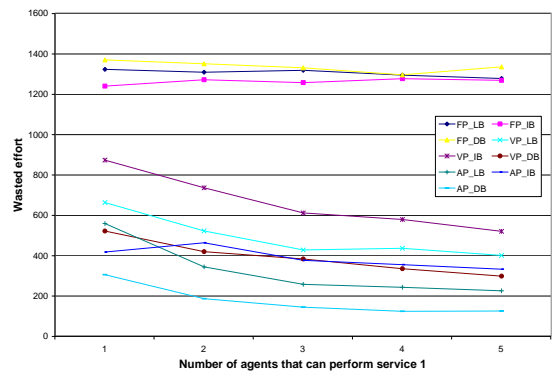

Figure 14: Wasted effort for various strategies over all experiments.

	Manager		
	FP	VP	AP
LB	8, 100	1262, 827	686, 1332
IB	3, 94	1326, 426	629, 975
DB	5, 120	541, 1710	214, 2103

Figure 15: Game theoretic analysis of the system with 2 agents capable of performing service 1.

	Manager		
	FP	VP	AP
LB	7, 90	1272, 776	684, 1300
IB	3, 82	1247, 376	616, 909
DB	4, 103	592, 1647	247, 2019

Figure 16: Game theoretic analysis of the system with 3 agents capable of performing service 1.

fixed-price mechanisms to various adaptive pricing mechanisms and showed that adaptive agents generally produce greater profit for the company and lead to a scalable load distribution and an acceptable revenue distribution. Furthermore, we showed that a population with adaptive agents is a dominant equilibrium among the possible population combinations. These findings confirm that distributed workflow enactment in an enterprise will require adaptive agents in order to provide both stability and profit maximizations.

This work represents only a first step in the development of distributed workflow enactment protocols. There are several important issues which this work has only started to address but which need to solved. We first need to verify that the stable solutions are indeed immune to exploitation by mutant strategies. We are currently running experiments to verify that these are evolutionary stable strategies. The decision-making abilities of the agents are crude (even if effective). We are developing more complex utility-based decision functions that the agents can use to make their decisions. However, given the large uncertainty in the system—we do not know which new jobs will arrive—it is unclear if more sophisticated decision-making will lead to better agent or system performance.

We also hope to extend the model so as to make it more realistic. We can add features such as: agent failures, the ability to decommit with or without penalty, workflows that parallel BPEL4WS with the use of "fork", "join", "parallel", and "loop" primitives, services of varying time spans, etc. Finally, we are considering the possibility of eliminating the manager agent entirely and have the service agents

	Manager		
	FP	VP	AP
LB	6, 72	1245, 670	657, 1145
IB	2, 64	1103, 331	556, 789
DB	4, 80	652, 1483	274, 1917

Figure 17: Game theoretic analysis of the system with 4 agents capable of performing service 1.

negotiate among themselves to decide who is going to do which service and how the revenue is going to be divided.

We believe that our basic architecture for distributed workflow enactments likely to see widespread implementation in the near future because it re-uses all the workflow knowledge that companies have generated for themselves over decades of work and it provides a clean mapping to the incentives of the employees/services and the company.

References

[Buhler & Vidal 2004] Buhler, P., and Vidal, J. M. 2004. Enacting BPEL4WS specified workflows with multiagent systems. In *Proceedings of the Workshop on Web Services and Agent-Based Engineering*.

[Buhler & Vidal 2005] Buhler, P., and Vidal, J. M. 2005. Towards adaptive workflow enactment using multiagent systems. *Information Technology and Management Journal* 6(1):61–87.

[Buhler, Vidal, & Verhagen 2003] Buhler, P.; Vidal, J. M.; and Verhagen, H. 2003. Adaptive workflow = web services + agents. In *Proceedings of the International Conference on Web Services*, 131–137. CSREA Press.

[Jennings *et al.* 2000] Jennings, N. R.; Norman, T. J.; Faratin, P.; O'Brien, P.; and Odgers, B. 2000. Autonomous agents for business process management. *International Journal of Applied Artificial Intelligence* 14(2):145–189.

[Muth *et al.* 1998] Muth, P.; Wodtke, D.; Weissenfels, J.; Dittrich, A. K.; and Weikum, G. 1998. From centralized workflow specification to distributed workflow execution. *Journal of Intelligent Information Systems* 10(2):159–184.

[Saha & Sen 2005] Saha, S., and Sen, S. 2005. Aggressive pricing to exploit market niches in supply chains. *IEEE Intelligent Systems* 20(1):36–41.

[Singh & Huhns 1999] Singh, M. P., and Huhns, M. P. 1999. Multiagent systems for workflow. *International Journal of Intelligent Systems in Accounting, Finance and Management*.

[Smith 1981] Smith, R. G. 1981. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* C-29(12):1104–1113.

[Vidal, Buhler, & Stahl 2004] Vidal, J. M.; Buhler, P.; and Stahl, C. 2004. Multiagent systems with workflows. *IEEE Internet Computing* 8(1):76–82.

[Vidal 2003] Vidal, J. M. 2003. Learning in multiagent systems: An introduction from a game-theoretic perspective. In Alonso, E., ed., *Adaptive Agents: LNAI 2636*. Springer Verlag. 202–215.

[Wilensky 1999] Wilensky, U. 1999. NetLogo: Center for connected learning and computer-based modeling, Northwestern University. Evanston, IL. `http://ccl.northwestern.edu/netlogo/`.