Incentive to Work: Deriving Cooperation Among Self-Interested Agents (Preliminary Report)

Eithan Ephrati
Computer Science Dpt.
University of Pittsburgh
tantush@cs.pitt.edu

Motty Perry
Center for Rationality
The Hebrew University
motty@coma.huji.ac.il

Jeffrey S. Rosenschein
Computer Science Dpt.
The Hebrew University
jeff@cs.huji.ac.il

Abstract

In this paper we analyse a particular model of control among intelligent agents, that of *non-absolute control*. Non-absolute control involves a "supervisor" agent that issues orders to a group of "subordinate" agents. An example might be an Internet user who issues a query to a group of software agents on remote hosts, or a human agent on Earth directing the activities of Mars-based semi-autonomous vehicles. The members of the subordinate group are assumed to be self-motivated, and individually rational (i.e., they are basically willing to carry out the supervisor's request if properly compensated). This assumption gives rise to the need for a reward policy that would motivate each agent to contribute to the group activity. In this paper we introduce such a policy under certain simplifying assumptions.

1 Introduction

Research on planning in Distributed Artificial Intelligence (DAI) has focused on two major paradigms: *planning for multiple agents* (where plans for multiple agents are devised, usually implying a central agent that specifies actions of others), and *distributed problem solving* (where multiple agents at various hierarchical levels of authority plan in a distributed fashion). We are interested in that important family of scenarios in which the control method combines both aspects. In these scenarios, there are *supervising* and *supervised* agents, where the control of a supervising agent over the supervised ones is non-absolute (see, for example, [2]).

Within the field of DAI there is also a distinction between two research agendas: *Distributed Problem Solving* (DPS) and *Multi-Agent Systems* (MAS). In the first area, agents are assumed to be created by the same designer (or group of designers), and thus work together to solve common goals. In economic terms, the agents have a common preference profile that yields an identical utility function. On the other hand, researchers in Multi-Agent Systems consider agents that may act selfishly towards the achievement of their private goals. Thus each agent may have its private profile of preferences, and a distinct individual utility function. The assumption is made that agents will help one another only when it is in their own best interests (perhaps broadly defined).

This paper addresses the issue of plan execution within Multi-Agent Systems, under the model of non-absolute control.

As an example, imagine that you are an Internet user with a variety of tasks to carry out around the world, collecting information, copying files, and printing documents. You have incomplete information about the world and limited time, so you hand your problems over to software agents whose task it is to do what is necessary to achieve your goals. For example, you may specify the high-level goal of retrieving the latest version of some public-domain anti-virus software. With the goal handed over to a group of agents, a plan is developed to locate the desired software, ensure that it is readable and in a public directory (this may involve receiving permission from a local host to move files between directories and alter their protection), copy the file over to your local host (using the appropriate protocol and format), then notify you that the file is available. If the software was not public domain, a suitable payment may be arranged automatically by one software agent to another. Similar scenarios might involve collecting specific information from remote news servers, or locating a given person's email and regular mail addresses [3].

The key point here is that a group of software agents, acting in concert, can carry out the overall plan that is developed. Moreover, it is possible that these agents are administered by separate organizations on separate machines all over the world. They are truly heterogeneous systems. The agent at the University of Michigan is willing to perform a lengthy search for you, but such work consumes resources. It seems reasonable that this software agent's institution should be compensated for all this effort—the agent will do whatever is necessary, but it expects to be properly motivated.[1] It may be the case that, while you know which agents were involved in

[1] Currently, there are primitive software agents that carry out Internet searches or tasks, such as the Gopher programs and Knowbots [7]. Of course, such work is today carried out for free. However, information is often simply unavailable over the Internet because there is no way to compensate its owners—currently, the information is either free or inacces-

satisfying your goal, you are unable to verify the exact amount of effort that each remote agent expended. Your system of compensation must motivate all the agents to do their share of the work, without detailed information about how much to pay each one.

As another example, imagine that you have sent a group of autonomous robots to build a space station on the surface of Mars. Though your communication with the group is limited, you are able, from time to time, to send them goals, such as "Build a tower at Region A." Knowing their (possibly differing) capabilities you may reason what their optimal multi-agent plan should be. Since the agents are self-motivated (possibly manufactured by different designers) you also give them some incentive to work. The simplest way to encourage each agent to work would be to pay it (or its designer) the true cost of any operation that it performs plus some extra amount.[2]

However, since communication is limited and your time is expensive, you would not like to treat each agent in the group individually. Instead, you would rather treat the group of agents as a whole and upon completion of the project reward all of its members identically. Unfortunately, such an attitude may lead to undesirable outcomes. Since each agent is self-motivated, it may be tempted to avoid the group's activity altogether and let others do the work for which it will be rewarded anyway. Thus, the project might not be executed at all. In the literature of Economics, this classic problem is known as the *free rider problem* [8].

In this paper, using concepts from Game Theory, we introduce a solution to this problem under certain simplifying assumptions.

Assumptions, similar to ours, about agents' selfish behavior has motivated the development of other economics-based approaches to deriving cooperation among individual agents. For example, Kraus [5] has used a reward method based on a monetary system to convince agents to accept contracts that were suggested by other agents. Wellman [9] has used an iterative bidding mechanism to engender efficient cooperation based on an economic market equilibrium.

In economics, the subfield of principal-agent theory addresses a problem somewhat similar to the one in which we are interested. There, the supervisor (principal) tries to find the right incentives (payment) that will make the agent report true data about the domain in which it operates and, thus, in effect, to inform the principal how to make him more productive [4].

2 The Model

Our scenario involves a supervisor and group, A, of n subordinate agents. The subordinate agents are self-interested utility maximizers. The supervisor is not able to control, or interested in controlling, each member of the group separately. We therefore would like a control method that allows the supervisor to consider the group of agents to be anonymous and symmetric.

Since the agents are not benevolent, given a goal (G) they must be motivated to work. A straightforward policy would be to pay the agents some "extra" amount for any work they carry out. Unfortunately, if the agents are to be paid symmetrically this might not work. Each agent might be tempted to "free ride" the others and be paid for the others' work. Under such circumstances, the goal might not be achieved at all.

We are looking for a reward policy that will induce cooperation among the agents, while answering the following (hierarchically ordered) criteria:

(a) Achievement of the goal should be guaranteed.

(b) Agents should be rewarded with the minimal possible amount of payment.

(c) The goal should be achieved within the minimal possible time.

(d) The work division among the agents should be "just."

The reward policy that we consider in this paper involves two parameters:

(i) The declared final payment, V, which is assumed to be equally divided among the agents if, and when, the goal is accomplished.

(ii) A discrete discount function, $\delta: P \times T \Rightarrow \mathbb{R}$ that reduces the final prize as a function of the development of the optimal plan and the elapsed time ($\delta(T_i)$ will determine the discrete discount for the elapsed execution time T_i).

3 An Example

Consider a simple scenario in the slotted blocks world. There are four slots (a, b, c, d), five blocks $(1, 2, 3, 4, 5)$, and the world is described by the relations: **On**(Obj_1, Obj_2)—Obj_1 is stacked onto Obj_2; **Clear**(Obj)—there is no object on Obj; and **At**$(Obj, Slot)$—Obj is located at $Slot$. The function **loc**(Obj) returns the location (slot) of Obj.

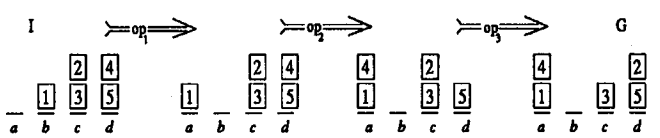

Figure 1: A Blocks World Example of a Serial Plan

There are three agents operating in the world. The start state and the goal state are shown, respectively, at the far left and the far right of Figure 1. The goal's worth is 4.

There is only one available operator:
Move(Obj_1, Obj_2)—place Obj_1 onto Obj_2. This operator can be characterized by the following STRIPS-like lists:
[Pre : $Clear(Obj_1)$, $Clear(Obj_2)$, $On(Obj_1, Obj_x)$],

sible. It is to be expected that more sophisticated means of employing agents will develop in the future.

[2]By "payment" we do not necessarily mean money. A more suitable payment might be resources such as an energy supply or time slots for individual activity or access to your own database.

[Del: $On(Obj_1, Obj_x)$, $Clear(Obj_2)$, $At(Obj_1, loc(Obj_1))$]
[Add: $On(Obj_1, Obj_2)$, $At(Obj_1, loc(Obj_2))$]
[Cost: 1]

Given the goal G, we would like to find a policy that causes the agents to follow the multi-agent optimal plan that achieves that goal.

3.1 Game Theoretic Concepts of Solution

The general kind of interaction that we consider here may be viewed as a game. Each agent i chooses a strategy s_i (within a computer, of course, the "strategy" is simply the program controlling the computer's choices). The strategy tells the agent which action (declaration of preferences, in our case) to choose at each instant of the interaction. The combination of strategies played by the entire group ($S = (s_i, s_{-i})$) determines the outcome of the interaction, and in particular determines the resulting payoff for each agent (π_i).

Game theory has addressed many interactions similar to the one considered here. Such interactions have been analyzed so as to determine what an agent's chosen strategies would be, given the rules of the interaction. Our aim is complementary; it is to *design* rules that would induce the agents to adopt some specific strategy that we consider to be desirable.

All possible developments of the interaction may be represented by a game tree (as an example consider Figure 2). Each node represents a decision choice of some player; each different choice is represented by a different branch. Given the history of the interaction, an agent might not be able to distinguish, among a set of possible nodes, which one is the actual node. This set is called the information set at that particular point. Each path on the tree describes one possible interaction. The end nodes of the tree describe each agent's resulting payoff from that path.

To be motivated to adopt a particular strategy, a rational selfish agent should be convinced that its chosen strategy is superior in some sense to its other, alternative, strategies. The most common solution in game theory derives cooperation as the best response to the other agents' cooperative behavior:

Definition 1 *The strategy combination s^* is a* Nash equilibrium *if no agent has an incentive to deviate from his strategy given that the other agents do not deviate. Formally* $\forall i,\ \pi_i(s_i^*, s_{-i}^*) \geq \pi_i(s_i', s_{-i}^*),\ \forall s_i'$.

This concept of solution was used (within the DAI literature) in [9, 10]. The drawback of the Nash equilibrium is that in general there are several equilibrium points for the same game, and the desirability of a strategy is considered only from a player's viewpoint at the start of the game (not taking into consideration *all* possible paths of the game). Thus, it might be difficult to have the group of agents converge to a specific equilibrium[3] and the equilibrium point may be sensitive to the dynamics of the interaction.

A much stronger concept of solution (the second one in the hierarchy of solutions) derives the desired strategy (cooperation in our case) to be the unique equilibrium along any development (path) of the interaction.

Definition 2 *A* subgame *is a game consisting of an information set which is a singleton in every player's information partition, that node's successors, and the payoffs at the associated end nodes, such that all information sets that include any of that node's successors do not include a node which is not one of its successor nodes.*

A strategy combination is a subgame perfect Nash equilibrium *if (a) it is a Nash equilibrium for the entire game; and (b) its relevant action rules are a Nash equilibrium for any subgame.*

In this paper we focus on a policy that induces a solution which is in subgame perfect equilibrium (for a similar approach, see [1, 6]).

3.2 Assumptions and Definitions

- The *goal* G is a set of predicates. V is the value of the final prize the group of agents will be awarded as a whole upon completion of G.
- There is a (random) order $(a_1, a_2, \ldots, a_n)$ over the n agents. Following this random order, each agent is supposed to contribute to the plan in its turn.
- $P(s_0, G)$ is the optimal multi-agent plan that transforms the initial state s_0 into a state that satisfies the goal G (we will denote this plan by P). The length of this plan will be denoted by $|P|$. The set of operators that construct it is denoted by $[op_1, op_2, \ldots, op_{|P|}]$.
- The time duration of each operator op_i is denoted by t_i. With respect to the construction of the plan P, we denote the minimal accumulated completion time of each operator op_i in P by T_i.
- There exists a *cost function* ($c: OP \times A \Rightarrow \mathbb{R}$) over the domain's operators. With respect to this function we define the cost of a set of operators to be the sum of the costs of its elements. Thus, the cost of a plan, $c(P)$, equals $\sum_{k=1}^{m} c_i(op_k)$ (where i denotes the agent that performs op_k).

  $\hat{c}$ denotes the maximal individual contribution needed by any agent ($\hat{c} = \max_i \sum_{op_j \in P} c_i(op_j)$).
- We assume that $\hat{c}$, as well as the duration of each operator that makes up the plan and the order of execution are known to the supervisor (these assumptions are relaxed in Section 6).

4 Serial Plans

We first examine multi-agent plans that are constructed by a series of consecutive operators. We call such plans *strictly serial*:

Definition 3 *A multi-agent plan is said to be* strictly serial *if it involves the invocation of at most one operator at the same time.*

[3] On the other hand, a single equilibrium point among equivalent ones can be specified and agreed upon ahead of time by agent designers, who might be indifferent among the different points or even all prefer one equilibrium over the others.

The optimal plan that achieves the goal presented in Figure 1 is strictly serial. Note that in a strictly serial plan each operator may be indexed according to its appearance in the sequence that makes up the plan. Thus, the plan cannot be completed at any time prior to $\sum_0^m t_i$.

Theorem 1 *The optimal reward policy for a serial plan is* $V = n \times (\hat{c} + \epsilon) + |P| \times \epsilon$ *and* $\delta(T_i) = \epsilon$.

The first element of the reward is the actual payment to be given to the group. Note that since all agents are treated symmetrically, all should be rewarded with the same value that should suffice to motivate the one that contributes the most ($\hat{c}$) plus a profit of ϵ. The second element will never actually be spent; its role is to motivate the agents to finish the plan execution within the shortest time.

The proof shows that given this policy, there is only one sub-game perfect equilibrium in which each agent performs one operator in his turn along the equilibrium path. Figure 2 shows the game tree that results from invoking the proposed policy on the example given in Figure 1. The equilibrium path is the right branch of the tree (denoted by a thicker line). For simplicity, we assume that if the plan will not be completed at T_3, no payment will be transferred to the group at all. The maximal possible payoffs as a result of each path are shown at the leaves.

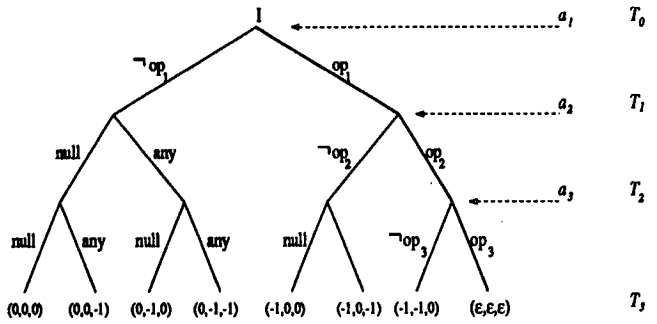

Figure 2: The Game Tree of the Blocks World Example

At the beginning of the process (T_0) agent a_1 may choose to either perform the first operator of the plan (op_1) or any other operation instead (including null). Choosing not to contribute would present a_2 with a choice that would yield a payoff of at most 0 (if in its turn a_2 would not do anything). Following this path, a_3 would also prefer the *null* operator, and no part of the plan will be performed. On the other hand, if a_1 would choose to perform op_1, a_2 would be faced with a choice that could yield him a payoff of ϵ, counting on a_3 to prefer ϵ over any payoff ≤ 0. Thus, a_1 can be assured that performing the first operator would guarantee him a payoff of ϵ. The same reasoning process holds for the other members of the group, which establishes the equilibrium.

Proof. The formal proof is by (backward) induction on k, the number of steps required to complete the plan:

1. $k = 1$; then if the $|P| - 1$ steps of the equilibrium path were followed, a_m (actually $a_{n \bmod n}$) may either perform op_m and complete the plan, with a payoff of ϵ, or perform any other operation (including null) with a payoff ≤ 0. Clearly, the first alternative strictly dominates the latter.
2. We will show that the claim holds for $k = |P|$, assuming that it holds for any $k \leq |P| - 1$. $k = m$ implies that it is a_1's turn to decide whether to perform op_1 or perform null. The second alternative may guarantee a payoff of 0. However, due to the induction assumption the former alternative guarantees ϵ. Therefore a_1 would strictly prefer to contribute op_1. □

Of course strictly serial plans are of less interest in multi-agent environments. The power of multi-agent planning is exploited when sets of operators in a plan may be invoked in parallel. The next section is concerned with parallel plans.

5 Parallel Plans

An example of a parallel plan is given in Figure 3. The goal state is shown at the far right. The optimal plan that accomplishes this goal involves a_1 performing $\langle M(1, b, a) \rangle$ and a_2 *simultaneously* performing $\langle M(4, b, d) \rangle$. When parallel plans are considered there is a need for a somewhat more sophisticated reward policy.

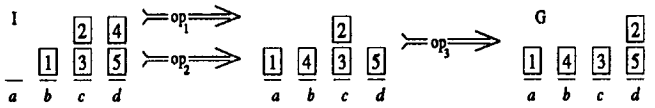

Figure 3: A Parallel Plan

Definition 4 *A multi-agent plan, P, is said to be a* parallel plan *if two or more of the operators that make it up may be carried out simultaneously. We divide the set of operators that make up a parallel plan into consecutive subsets of operators OP_k such that each operator in OP_k should be invoked simultaneously in the optimal plan.*

The agents participating in each OP_k are indexed according to their initial order by $\{a_{k_1}, a_{k_2}, \ldots, a_{k_p}\}$, where $p = |OP_k|$.

Note that it follows from the definition that a parallel plan may be completed in $\sum_k \max_{op_i \in OP_k} t_i$. Also note that a strictly serial plan is a special case of a parallel plan where each OP_k contains only one member.

Figure 4 shows why the previous policy may not be effective. Since a_2 has to act in parallel with a_1, it cannot distinguish between the two alternative nodes of the game tree (a_2's information set includes these two nodes). Therefore, a_1 may not count on a_2 to perform op_2, and thus the equilibrium path becomes the null path!

A straightforward solution would be to let the agents complete the plan as if it were a strictly serial plan, and employ the previous reward policy. But then, of course, the total elapsed time would be longer than it actually

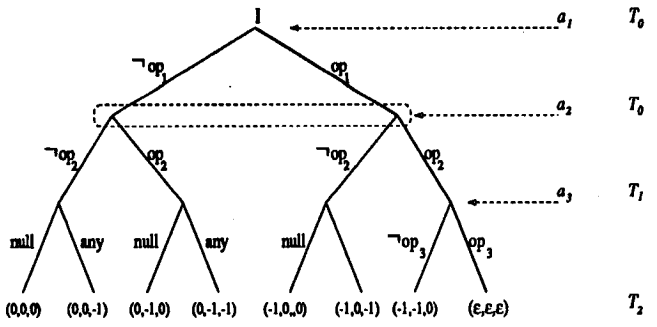

Figure 4: The Game Tree of a Parallel Plan with a Serial Policy

should be, and the advantage of the multi-agent environment will be lost. Fortunately, a refinement of the "serial policy" overcomes this problem.

Theorem 2 *The optimal reward policy for a parallel plan is* $V = n \times (\hat{c} + \epsilon) + (\sum_k |0P_k|) \times \epsilon$ *and* $\delta(T_i) = \epsilon$ *where* T_i *is defined as in the serial case.*

The basic idea of this policy is that in each information set each agent can be assured that even though his contribution may be made simultaneously, he may wait for the preceding agents (according to the order over the agents) to contribute their share, only then perform his share, and still end up with a profit. However, knowing that this fact holds for all other agents in the information set, each agent is motivated to act as soon as possible (i.e., simultaneously) in order to increase its payoff.

Proof.

We prove that given this policy, there is only one subgame perfect equilibrium in which each agent performs one operator in his turn along the equilibrium path. The proof is by induction on k, the number of *simultaneous* steps required to complete the plan:

1. $k = 1$. There are $|OP_1| \geq 1$ operators to be carried out simultaneously, at time T_1^0, in order to complete the plan. Let A_1 denote the ordered group of agents $(a_{1_1}, a_{1_2}, \ldots, a_{1_m})$ that is assigned the operators in OP_1. We follow the recursive reasoning process of each $a_{1_j} \in A_1$ (the agents that should perform the operators in OP_1 according to their sequential order) and show why each a_{1_j}'s best response is to simultaneously perform at time T_1^0 the operator it is assigned:
   (a) Consider a_{1_1}. If it suspects that any other agents in its information set, a_{1_j} (such that $j > 1$), will not perform the operator that it has been assigned (op_{1_j}), then by not performing any operation it may save itself the cost of that operator. However, following the equilibrium path of the strictly serial case, by performing op_{1_1}, it ensures itself a payoff of at least ϵ (since in the worst case each agent in A_1 will wait for its preceding agent to complete his contribution and only then perform its own contribution). Thus, a_{1_1}'s best response is to perform op_{1_1} at T_1^0.
   (b) Now consider a_{1_2}. Following the above reasoning it may count on a_{1_1} to perform op_{1_1} at T_1^0. Therefore a_{1_2} can choose either to wait for a_{1_1} to complete its contribution and then perform op_{1_2}, and thus ensure itself a payoff of at least ϵ, or perform op_{1_2} simultaneously and increase its payoff by $\frac{\epsilon}{|OP_1|}$. Obviously, the second alternative is superior.
   (c) To reach the bottom of the recursion, consider a_{1_m}; Following the same reasoning as the other agents, it may assume that all the other agents in its information set will perform their contribution at time T_1^0. Therefore it is faced with either waiting for the other agents in A_1 to simultaneously contribute their share and only then contribute op_{1_m}, with a payoff of at least $\epsilon + \frac{\epsilon}{n} \times (|OP_1| - 2)$, or perform simultaneously with a payoff of at least $\epsilon + \frac{\epsilon}{n} \times (|OP_1| - 1)$. Again, the simultaneous move is superior.
   Thus the entire group would prefer to contribute simultaneously.
2. Assume that the claim holds for any $j < k$ and consider the k'th simultaneous move. Following the same line of reasoning as in the last simultaneous act, each agent in A_k may reason that by performing his share in OP_k simultaneously, it may increase its payoff by $\frac{\epsilon}{|OP_k|}$.
   Thus at each simultaneous step all members of the information set will follow the equilibrium path. The plan will be completed during the shortest possible time and each agent will gain a payoff of $\epsilon + \frac{\epsilon}{n} \sum_k (|OP_k| - 1)$. □

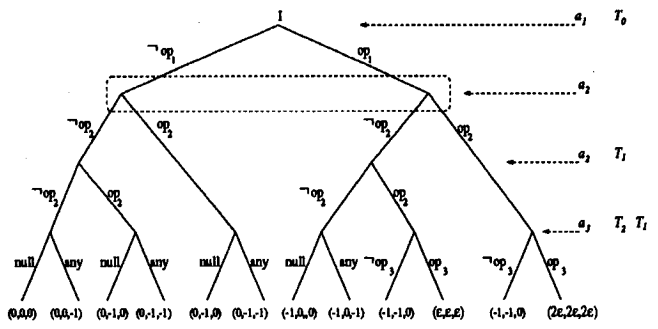

Figure 5: The Game Tree of a Parallel Plan

Figure 5 demonstrates how such a policy would work with the example. Consider a_2 which is trying to figure out whether to perform op_2 at T_0. Assuming that a_1 will carry out op_1 at T_0, his best strategy would be to carry out op_2 at T_0, deriving a final payoff of $2 \times \epsilon$. However, not being sure about a_1's behavior a_2 may also wait to see whether a_1 did contribute op_1, and only then follow his best strategy (which is performing op_2 at T_1, with ϵ payoff).

This line of a_2's reasoning can be followed by a_1. Therefore, his best strategy would be to carry out op_1 at T_0, ensuring itself a payoff $\geq \epsilon$.

In turn, a_2 can follow this last line of reasoning of a_1, and conclude that a_1 *will* carry out op_1 at T_0, and therefore a_2 would be better off carrying out op_2 simultaneously. Thus, the plan will be executed within the shortest possible time.

6 Relaxation of Assumptions

In this section we address some of the assumptions mentioned above.

Pre-Existing Order Among the Agents: Our analysis assumes a certain *a priori* order over the agents in the group. We further assumed that execution is carried out in correspondence to that order. However if agents have differing capabilities and cost functions over the domain's operators, such a restriction may prevent the construction of the optimal plan.

This assumption may be replaced with the assumption that all agents may calculate what the optimal plan is and, thus, what the corresponding order should be. Alternatively, this order may be determined by the supervisor, particularly if there exists more than one optimal plan. Once the order is determined, their corresponding subgame perfect strategy would still remain the same.

Note that there is a tradeoff to be made here by the designer. Following this payment policy, the actual plan may be improved; however the value of $\hat{c}$ (which is to be multiplied by the number of agents in the group so as to determine the final payment) may rise significantly.

The Maximal Individual Contribution of Any Agent: Another assumption that was made was that the supervisor knows the exact $\hat{c}$. In some domains, such an assumption may be reasonable. For example, in the domain of communication networks, it might be plausible to restrict software agents to a certain action cost (i.e., no possible action will ever cost an agent, or its institution, more than some x amount). In such a case the maximal individual contribution may be known.

Moreover, since the reward policy that was presented in Theorem 2 would still work for any $\hat{c}' \geq \hat{c}$, it is sufficient to determine the upper bound for *any* individual agent (but the associated payment should be issued accordingly).

The Exact Time Steps of the Optimal Multi-Agent Plan: We have also assumed that the discount function relies on the actual steps of the executed plan. In many scenarios the supervisor might not be interested in or able to reason about the actual time-steps of the plan.

Fortunately, any discount function that would reduce the payoff of the agents due to delay in completion of the plan is sufficient. More precisely, let T^* be the upper time bound desired by the supervisor for completion of the plan, and T' be the serial time needed for the completion of the plan. Then any (continuous or discrete) discount function that would reduce the final prize starting at time T^*, at a rate greater than the development of the plan, is sufficient (i.e., a discount that would capture the shortest time-interval which is needed for the execution of any operator). An example of such a discount function might be the following continuous function (that will be invoked only when the execution time reaches T^*): $\frac{T^*+t}{T'-T^*} \times c$ such that t corresponds to the actual (real world) continuous time difference between T^* and the actual completion of the plan and c is the motivating payoff to be paid upon completion of the plan at T^*. We denote any such function by $\hat{\delta}$-discount.

The following theorem summarizes these relaxed assumptions. Note that although the completion of the project within the optimal time is guaranteed, the supervisor would suffer a loss of utility due to his lack of knowledge:

Theorem 3 *The optimal reward policy for a parallel plan presented in Theorem 2 would yield the completion of the plan within the shortest time for any $\hat{c}^* \geq \hat{c}$ (where $\hat{c}$ refers to the maximal individual contribution presented in Theorem 2), any other $\hat{\delta}$-discount policy, and any alternative common knowledge ordering over the group.*

Proof.

The proof is by a straightforward modification of Theorem 2. □

7 Conclusions

We have introduced a method to derive cooperation among self-motivated agents. Through the use of a reward policy, agents find it to be in their own best interests to cooperate with the other members of the group. The reward policy can be used both when the multi-agent plans are strictly serial, and when they are parallel plans. The resulting cooperation is efficient.

Use of these kinds of reward policies will become increasingly important as remote agents, independently designed and maintained by various institutions, will need to work together to carry out distributed tasks. It will be necessary to motivate these heterogeneous agents to work together using appropriate incentives.

Acknowledgments

This work has been partially supported by the Air Force Office of Scientific Research (Contract F49620-92-J-0422), by the Rome Laboratory (RL) of the Air Force Material Command and the Defense Advanced Research Projects Agency (Contract F30602-93-C-0038), and by an NSF Young Investigator's Award (IRI-9258392) to Prof. Martha Pollack. And by the Leibniz Center for Research in Computer Science, and by the Israeli Ministry of Science and Technology (Grant 032-8284).

References

[1] Anat R. Adami and Motty Perry. Joint project without commitment. *Review of Economic Studies*, (58):259–276, 1991.

[2] E. Ephrati and J. S. Rosenschein. Planning to please: Following another agent's intended plan. *Journal of Group Decision and Negotiation*, 2(3):219–235, 1993.

[3] O. Etzioni, N. Lesh, and R. Segal. Building Softbots for UNIX (Preliminary Report). Technical Report 93-9-01, Computer Science Department, University of Washington, 1993.

[4] B. Holmstorm and P. Milgrom. Multitask Principal-Agent analyses: Incentive contracts, asset ownership, and job design. *The Journal of Law, Economics and Organization*, 7:24–52, September 1991.

[5] S. Kraus. Agents contracting tasks in non-collaborative environments. *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 243–248, 1993.

[6] S. Kraus, J. Wilkenfeld, and G. Zlotkin. Multiagent negotiation under time constraints. CS-TR-2975, University of Maryland, College Park, Maryland, 1992.

[7] Ed Krol. *The Whole Internet.* O'Reilly & Associates, Sebastopol, CAL, 1992.

[8] T. Muench and M. Walker. Identifying the free rider problem. In J. J. Laffont, editor, *Aggregation and revelation of preferences*, chapter I(4), pages 61–87. North-Holland, 1979.

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

[10] G. Zlotkin and J. S. Rosenschein. A domain theory for task oriented negotiation. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 416–422, Chambery, France, August 1993.