Many is More, But Not Too Many: Dimensions of Cooperation of Agents with and without Predictive Capabilities

Matthias Scheutz and Paul Schermerhorn
Artificial Intelligence and Robotics Laboratory
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
{mscheutz,pscherm1}@cse.nd.edu

Abstract

This paper examines the tradeoffs between agents that can predict (and, therefore, take into account) other agent's actions and agents that act on their own without taking other agents' actions into account. A simple prediction mechanism allows agents to make reasonably accurate guesses about other agents' future actions, thereby making better decisions about their own actions. Our experimental findings from a multiagent object collection task suggest that not only do predictive agents perform better than non-predictive agents, but there are circumstances in which simple reactive prediction mechanisms perform as well as complex deliberative prediction mechanisms.

1 Introduction

Suppose you need to design a control system for a given number of robots that will gather items in an environment as quickly as possible without any prior knowledge of their distribution and without being able to communicate, while keeping cost (in terms of components) and energy requirements of the controller (as it is running) as low as possible. Problems of this sort could arise in military applications (e.g., when supplies are flown in and dropped from planes, which units should retrieve which supplies, and in which order?), more mundane situations (e.g., when a shipper needs to pick up multiple packages throughout a city, which truck should pick up which packages, and in which order?), or in contrived robotic experiments. What kind of control system would achieve the best performance/cost ratio? Observations of simple reactive agents compared to complex deliberative agents indicate that in certain circumstances complex deliberative planning and prediction mechanisms are not necessary to achieve good performance on such collection tasks. However, without some sort of coordination mechanism, agents may work at cross-purposes, hurting the overall performance os the multiagent system.

In this paper we will investigate multiagent tasks of this kind, where multiple agents together need to collect objects in the environment and their performance is evaluated in terms of the time it takes to complete the task. It is possible, given the positions of the agents, items, and obstacles present in the environment, to solve the problem optimally at the beginning by exhautively enumerating all combinations and choosing the one with the shortest time to completion. Each agent can then be given its assignment of objects to collect (including the order in which they should be collected) and the ojects can be collected in the shortest possible time. However, the problem is intractible for even modest numbers of agents and items. Furthermore, in dynamic environments where the items can move from one location to another, the solution may need to be recomputed during the course of executing the solution. What is needed is a flexible solution that provides good (although not optimal) solutions in dynamic environments at a reasonable cost.

A first pass at a solution to the collection task would be for each agent to always attempt to collect the item that is closest to it. The intuition behind this solution is that going far out of the way to collect an item costs more than collecting the nearby one, making the greedy algorithm attractive. Especially in multi-agent environments, however, this strategy can lead to a great deal of wasted effort. If an agent moves toward the closest item, only to have another agent collect the item just before the first agent arrives, the first agent's effort is wasted.

One obvious solution to this coordination problem is communication; if agents can communicate their intentions to one another, effort will not be wasted by two agents attempting to collect the same resource. However, communication can be expensive, and, furthermore, it does not in itself solve the problem of who *should* retrieve the resource in question, it is merely a tool to be used by whatever control mechanism does solve the problem. The solution explored below does not rely on communication. Instead, agents use predictions of other agents' actions in order to implicitly coordinate their efforts on the collection task. In particular, we will study four different agent types: (1) purely *reactive agents* with no deliberative predictive ability, (2) *deliberative agents* with planning and information storage facilities, (3) *reactive-predictive agents* with the ability to predict the actions of other agents, and (4) *deliberative-predictive agents* with both deliberative and predictive capabilities.

By contrasting the performance of these four agent types, we demonstrate that the more computationally intensive architectures do not always provide as great an advantage as one might expect, particularly when considerations of computational cost are taken into account. In addition, the experiments described here provide insight into the usefulness of different types of control mechanisms in the context of agent cooperation. Much interesting and useful work has been done in the area of foraging and cooperation, in both robotic and simulated environments (e.g., [6, 3, 4, 12, 1]); this research focuses on the utility of the predictive mechanism as an implicit cooperative mechanism.

The paper is organized as follows: first, we define what we mean by "reactive", "deliberative", and "predictive" control. Then, we introduce the different agent types, define their control architectures and discuss the resultant behavioral dispositions. We briefly describe the experimental setup, and then report the results from several series of experiments with these agents. We analyze the results and discuss their implications for behavior coordination in multiagent tasks.

2 Cooperation and Behavior Coordination

Communication is a very powerful method of coordinating actions that lead to cooperative agent behavior (e.g., every agent knows what its goal is and communicates it to all other agents). Even if optimal group behavior may not be achievable (e.g., because of the nature of the task or the computational requirements of the distributed algorithm), there is at least the benefit that agents will *know* what other agents are up to (always assuming they tell the truth, of course).

In the absence of communication, reasoning about other agents' intentions based on observations can be a good substitution. If, for example, every agent in a group constantly observes the behavior of other agents, knows how other agents chose their behavior, and chooses its actions based on its predictions, then the agent group may be able to converge to (several) stable, coordinated behaviors.

While coordination of agent behavior may not be necessary for a task that can be performed by a single agent, it certainly can improve the performance of an agent group. In a simple object collection task like the one mentioned above, for example, a group of agents may be able to improve its performance by coordinating individual goals and actions (e.g., they may be able to avoid situations in which two or more agents chose to collect the same item). If all agents could implement the same algorithm (e.g., an algorithm that uniquely assigns to each agent a goal item, which they should collect) and if every agent cooperates (i.e., reliably follows the algorithm), agents can avoid conflicting goals and significantly improve the performance of the agent group.

While such coordination mechanisms lead to improved performance, the processing cost incurred by their architectural mechanisms can be substantial, as we will demonstrate below. Hence, the gain in task performance may not be worth the loss in processing efficiency. Often, sufficient performance in a multiagent collection task can be achieved either without explicit coordination or with coordination mechanisms that do not involve predicting other agents' behavior, e.g., avoiding coming too close to another agent. Below we describe agents that pursue their goal of acquiring items from the environment without taking other agents into account, in effect working against one another. Yet, they tend to avoid other agents, and in particular they avoid crowded regions. This simple rule causes them to display a certain degree of coordinated behavior, and more importantly, allows them to perform very well relative to the low complexity of architectural mechanisms that control their behavior.

Several questions ensue: When are such simple mechanisms sufficient? Under what environments do they reach a performance peak for a given number of collectible items as measured in terms of number of agents and the time it takes to complete the task? Does the performance depend on the ratio of agents and collectible items?

2.1 Experimenting with Architectures

One goal of this research is to examine the architectural requirements for multiagent cooperation, in particular, the relation between the cost of architectural mechanisms that implement strategies to coordinate agents and the increase in performance caused by agent coordination (if any). Conducting investigations at the architectural level allows us to study functional components and their effects on an agent's behavior. We can compare different kinds of control systems (as defined by their architectures) under different environmental conditions to assess their advantages and disadvantages. Components in a control system will have a cost associated with them, reflecting the energy expenditure to build and maintain them in an operational state. Comparing the performance of the architectures examined here in light of their relative computational demands provides insight into how useful various architectural components really are for a given task.

3 Reactive, Deliberative, and Predictive Architectures

Agent architectures play an important role in the understanding of the development of natural and artificial systems. [10, 9] They can be thought of as blueprints of control systems, where different functional components and their interconnections are depicted (e.g., see [8] for a more detailed definition of "agent architecture"). Since we would like to understand (1) what kinds of components (and arrangements thereof) are required to produce particular kinds of behaviors, and (2) what the relative tradeoffs of different control systems (and their implementations) are, we first need to define what we mean by "reactive" and "deliberative" control, or more to the point, what "reactive" and "deliberative architectures" are.

3.1 Reactive Architectures

Unfortunately, there seems to be a wide range of definitions of "reactive" that differ in substance (e.g., "reactive" as "stateless" versus "reactive" as "tight sensor-motor coupling"). Hence, it seems that "reactive" is best defined in opposition to "deliberative", i.e., as "not deliberative", which puts the burden on a definition of "deliberative". Since we are interested in demarcating an intellectually interesting difference, rather than trying to say what "deliberative" *really means*, we will construe "deliberative" as "being able to produce and use representations of hypothetical past or future states or as yet unexecuted actions (or sequences of such actions)". Note that according to this (negative) definition of "reactive", reactive architectures may make use of simple representations of the state of the world and/or the agent. But these representations will not explicitly encode goals, hypothetical states of the world or sequences of possible actions. And while we may be able to ascribe intentional states such as beliefs and desires to a reactive agent, the agent architecture contains no explicit representation of these states. For example, an agent which exhibits a behavior that could be described as "avoiding obstacles" can be said to have a goal of "avoiding collisions", even though this goal is not explicitly represented in the agent's control system.

3.2 Deliberative Architectures

As mentioned before, a *deliberative* architecture is one in which there is some consideration of alternative courses of action before an action is taken. Hence, there is need for the capacity to represent counterfactual states referring to hypothetical past or future states or as yet unexecuted actions (or sequences of such actions), in which at least some of the basic operations of the architecture are to produce, read, and write such counterfactual states. Such states include goals (descriptions of states to be achieved), plans (sequences of unexecuted actions), states describing the imagined consequences of performing an action in the current state or some hypothetical state, partial solutions generated during planning or problem solving, the hypothetical states of the agent's beliefs generated during belief revision, and many others. We further require that such states should be influential in the production of actions, in the counterfactual sense that, had the (counterfactual) state not been generated, the agent would have chosen a different action to execute.[1]

To represent counterfactual states, a deliberative agent requires a reusable working memory for the construction and comparison of hypothetical states and some means of deriving the consequences of actions performed in these states. At its simplest, this might be a set of memories of the consequences of performing the action in similar states in the past. The use of a common working memory limits the number of alternative courses of action that can be considered in parallel, and hence the degree of parallelism possible within a deliberative architecture.

All other things being equal, a deliberative architecture must be slower and require more resources than a reactive architecture which encodes a solution to any specific goal solvable by the deliberative architecture, since the generation of alternatives will take time. However, a deliberative architecture will typically be more space efficient than an equivalent reactive architecture, even though it will often require more space than a reactive solution to any given problem instance, since it can solve a *class* of problems in a fixed amount of space, whereas a reactive architecture requires space proportional to the number of problems. We can view this as an example of the standard space-time tradeoff, though in this case there is also the time required to code or evolve all the reactive solutions.

[1]Note that this definition implies no commitments as to whether the states and operations are fine grained, e.g., dealing with partial plans or alternative solutions and their generation and comparison, or whether the states and operations are "coarse grained", e.g., a single "plan" operator which takes a goal and a description of the current state and returns a plan with the rest of the fine-grained states and operators buried in the implementation of the architecture and invisible to the agent program and the agent state. Both cases have at least one counterfactual state and one operator that takes a non-counterfactual state and returns a counterfactual state.

4 Agents: Architectures and Behavioral Dispositions

In the experiments reported in this paper, we employ four different kinds of agents, *reactive*, *deliberative*, *reactive-predictive*, and *deliberative-predictive* agents, where the architectures of the“predicite” agents are extensions of their non-predictive counterparts.

All agents are standardly equipped with exteroceptive “vision” and “touch” sensors. *Vision* is used to detect items and other agents and *touch* to detect (1) impending collisions with agents and (2) items that are within reach for collection. In addition, the touch sensor is connected to a global alarm system, which triggers an automatic reflex-like action pattern, which the agent cannot suppress, to move it away from other agents.

On the effector side, agents have motors for locomotion and turning, and a mechanism for collecting. When agents come to a halt on top of an item, its collection mechanism suppresses the motors for locomotion until the item is collected, which takes one simulation cycle.

While different agents may have different short-term goals at any given time (e.g., reaching an item faster than another agent, or avoiding collisions with other agents), there are two long-term goals that are common to all of them: (1) *collection* (i.e., to acquire as many items as possible), and (2) *survival* (i.e., to avoid colliding with other agents in the environment). In the following, we will briefly describe the employed architectures and behavioral dispositions of each agent kind.

The Reactive Agents All agents process sensory information and produce behavioral responses using a motor schema-based approach [2]. Let $Ent = \{i, a, o\}$ be an index set of the three types of objects: *items, obstacles* and *agents*. For each object type Ent, a force vector F_t is computed, which is the sum, scaled by $1/|v|^2$, of all vectors v from the agent to the objects of type t within the respective sensory range, where ‘$|v|$’ is the length of vector v. These *perceptual schemas* are mapped into motor space by the transformation function

$$T(x) = \sum_{t \in Ent} g_t \cdot F_t(x) \qquad (1)$$

where the g_t are the respective gain values of the perceptual schemes. The gain values simply scale the effect of sensory input, providing a means by which to prioritize certain inputs (e.g., if collecting items is especially important, the item gain value could be higher than the agent gain value, so that sensing an item has a greater impact on the direction chosen than sensing other agents). These gain values are initialized to values determined to be reasonable via a series of experiments, and are kept constant throughout the life of a reactive agent.

Reactive agents always behave in the same way, given that their gain values are *constants*: their positive g_i makes them employ a greedy collection strategy (most of the time a “collect nearest” strategy [11]), whereas their negative g_o and g_a values make them avoid obstacles and other agents. The effect of g_a on the reactive agents’ behavior is to establish implicitly a “ranking” of who gets to collect an item first if multiple agents attempt to collect the same item: whoever is closest will be more strongly attracted to the item than repelled by the other agents, and hence be able to get to collect the item, whereas the other agents will be repelled more by the presence of agents than they are attracted to the item, and hence will move away. In a sense, g_a implements a simple “coordination” strategy, if only one that is “negatively” determined.

The Deliberative Agents Deliberative agents have several components that allow them to manipulate representations of collectible items in the environment. Most importantly they have a route planner that can determine which item is closest to them and how they can best get to it. It is first and foremost this ability of being able to represent entities in the environment that opens up further possibilities such as storing and retrieving representations, using them in planning and plan execution, etc. None of these possibilities are available to reactive agents, which have access to sensed objects only in a holistic manner (via an agglomerated force vectors).

The planner of the deliberative agents (based on a simplified version of the A^*_ϵ algorithm [7]) is given a list of items known to the agent (i.e., stored in the agent’s memory), and returns a *plan*, which is a list of headings and distances, of how to get to the nearest reachable item. The plan is then passed to a *plan execution mechanism*, which ensures that plan steps are executed. When other agents cross a deliberative agent’s route and the reflex is triggered, “re-planning” is initiated, and the agent will continue by executing the new plan. Re-planning is also performed if the item chosen by the agent has been collected by another agent in the meantime. A further difference between deliberative and reactive agents is that, while the schema-based mechanism of the reactive agents will not pick out the most direct route to an item (because of the influence of other items and agents), and may even move *away* from the nearest goal item (because of a cluster of objects further away in the opposite direction, or a cluster of agents in the direction of the nearest item), deliberative agents will find the nearest item and plan a route directly to it (while avoiding other agents), thus saving time and energy.

```
for all $A \in agentlist$ do
  $closest = infinitelyfaritem$
  for all $I \in itemlist$ do
    if $dist(A, I) < dist(A, closest)$ then
      $closest = I$
    end if
  end for
  if $dist(A, closest) < dist(Me, closest)$ then
    $remove(closest, itemlist)$
  end if
end for
```

Figure 1. Algorithm *eliminate-items*

The Prediction Extension In many cases agents will pursue the same goal, which reduces the overall efficiency of the agent group: an agent should not waste time moving toward the same item another agent is pursuing if the second agent is closer to the item than the first is. The function of the prediction extension is to make an "educated guess" as to which goals other agents might be pursuing, so as to eliminate those items as possible goals. The prediction extension was added to both kinds of agent architectures, creating two new classes of agents: reactive-predictive and deliberative-predictive. The algorithm implemented by the prediction component, which is used to eliminate common items (i.e., items that more than one agent is likely to pursue), is given in Figure 1. It ensures that no two agents will ever be pursuing the same goal item.

The Reactive-Predictive Extension The extension is integrated into the reactive agent by applying the algorithm `Eliminate-Items` to the appropriate type in Ent_k before its force vector is computed. This effectively functions as a "perceptual filter" that prevents those perceptual schemes from being instantiated that correspond to items that might be pursed by other agents. Consequently, the agent cannot be attracted to those items any longer and will simply ignore them on its way to pursuing other items. Note that reactive-predictive agents do not entertain or manipulate representations of other agents' goals, nor do they "reason" about other agents' intentions.

The Deliberative-Predictive Extension The prediction extension is integrated into the deliberative agent by applying `Eliminate-Items` to the list of items passed to the planner, which in turn selects its goal. The chosen goal is then the closest item to the deliberative-predictive agent that is not the goal of any other agent (i.e., no other agent is closer to it and not closer to any other item). The planner proceeds as in a normal deliberative agent, and the plan executes to pursue the uncontested goal, until a conflict is detected, i.e., a situation in which the current goal might have been chosen by another closer agent, in which case replanning will be triggered. In sum, the agent ignores items that it believes other agents will pursue and obtain before it can. Note that the deliberative agent does maintain representations of other agents' intentions, i.e, of its beliefs about other agents' intentions to be avoid having to process the intentions of all other agents again and again at every cycle. That way intentions only need to be processed again, when items are collected or collisions occur.

5 Experiments

5.1 Experimental Setup

The simulation environment consists of a continuous, limited two-dimensional surface populated with collectible items and the different kinds of agents. Agents always move at a constant speed until they come to a collectible item, which they then pick up. When all items in the environment have been gathered, the simulation ends and the number of cycles executed is recorded. The simulation is ended prematurely if agents fail to collect all items within 10,000 cycles, in which case they "fail the task".

We conducted experiments to study the performance of the above agent kinds in the object collection task. For all of the following experiments, we limited the world to a squared area of 800 by 800 units, where the objects to be collected are randomly distributed in a subregion of 720 and 720.

Each experimental run is started with a fixed number of agents placed in random locations within the environment and is finished as soon as the last item is collected. The performance measure used (and depicted in the tables below) is the time it took the agents to collect all items (ten total). To be able to compare different agent kinds, the same initial conditions are used for all agent kinds for each run (i.e., the initial locations of agents and the locations of collectible items are the same). All results are averaged over 40 runs to be able to average out effects of initial positions.[2]

5.2 The Optimal Solution

As a benchmark against which to compare the performance of the four agent types, a program that finds the best solution for a given distribution of agents, items, and obstacles was implemented. The premise for the optimal solution is an agent type that can communicate with its peers, can compute as much as it wants, and can see everything. At the beginning of the collection task, one agent is responsible for

[2]We experimented with larger numbers, such as 100 runs, and found that the standard deviation is about the same, hence 40 runs for each experiment for each agent kind turned out to be suffi cient, in particular, since the t-test showed statistical signifi cance for the results discussed below.

```
$optimaldistance = impossiblylongdistance$
$assignlist = gen\_all\_assign(agentlist, itemlist)$
for all $S \in assignlist$ do
  $worstdistance = 0$
  for all $A \in agentlist$ do
    $bestdistance = impossiblylongdistance$
    $permutationlist = gen\_all\_perms(S(A))$
    for all $P \in permutationlist$ do
      $distance = calculate\_distance(A, P)$
      if $distance < bestdistance$ then
        $bestdistance = distance$
        $bestpath = append(A, P)$
      end if
    end for
    if $bestdistance > worstdistance$ then
      $worstdistance = bestdistance$
    end if
    $append(worstpath, bestpath)$
  end for
  if $worstdistance < optimaldistance$ then
    $optimaldistance = worstdistance$
    $optimalpath = worstpath$
  end if
end for
```

Figure 2. Algorithm *optimal-solution*

computing the best solution and communicating to each of its peers the set of items each is responsible for collecting along with the order in which they are to be collected.

The best solution is found by exhaustively enumerating all possibilities to find the one that yielded the shortest time to completion, as shown in Figure 2. Each assignment of items to agents is checked and for each assignment every collection order is checked. The algorithm computes the shortest Hamiltonian path in which some agent "visits" (i.e., collects) each item. For each assignment, the shortest collection order is computed for each agent and its assigned items. Then the longest agent's completion time is chosen as the best time required by the current assignment to perform the task. The shortest completion time of any assignment is the best time overall.

The results given below include the results from the optimal solution program. The numbers given are the average of the outcomes of the 40 initial conditions used by the other agents. Note that these results measure only time to completion and do not include the substantial computation time required to compute the best solution.

5.3 Experimental Results and Analysis

Figure 3 presents the results for reactive, reactive-predictive, deliberative and deliberative-predictive agents in 0-obstacle environments. For each agent type, ten experiments were performed, representing environments with one through five agents working on the collection task in two different environments. The first environment contained no obstacles, whereas the second contained five. These experiments are plotted along the x axis. The average number of cycles taken to complete the task is plotted along the y axis. Figure 4 similarly presents the results for 5-obstacle environments.

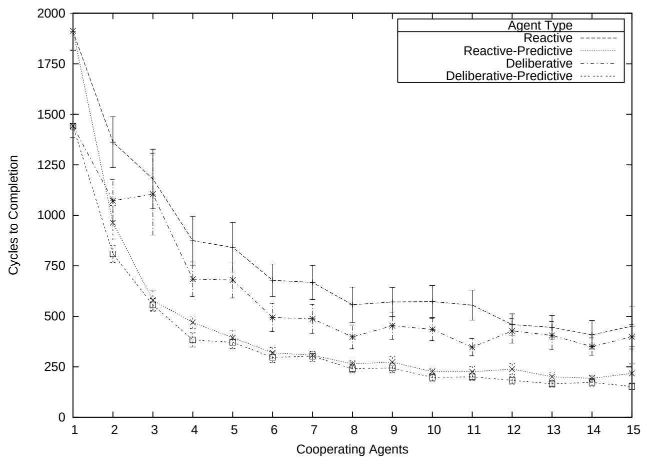

Figure 3. Average time to task completion (0 obstacle environment)

Unfortunately, not every group of agents was able to accomplish its task. Table 1 is a listing of how many failures were experienced by each agent kind for each task environment. The values plotted in Figure 3 do not include these failed tasks.

The above results point to a general performance ordering among the examined agent kinds from best to worst: deliberative agents with predictions, reactive with prediction, deliberative without prediction, and reactive without prediction. However, there are some interesting points illustrated in Figure 3. First notice that with only one agent performing the task, deliberative agents outperform reactive agents by a wide margin. This is because deliberative agents settle on an individual goal and move directly toward it. The field-based mechanism of the reactive agents, on the other hand, often moves agents on a curved trajectory toward items, because other items will also exert influence on the agent. Furthermore, a reactive agent may actually move *away* from the closest item in cases where there are several items clustered in the opposite direction, leading to less efficient collecting behavior. This result is unsurprising.

Interestingly, the two agent case already shows no significant difference between the reactive-predictive agents and the deliberative ones, and the deliberative agents are beginning to perform worse than deliberative predicting agents. That is, increasing the number of agents to just two allows

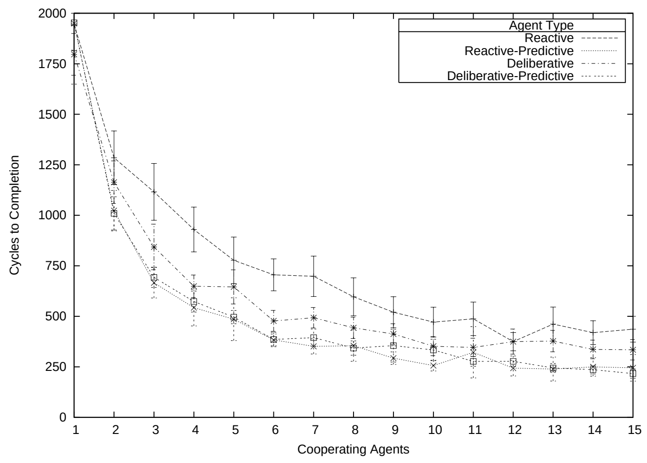

Figure 4. Average time to task completion (5 obstacle environment)

Table 1. Failed tasks (out of 40 total)

Agents	Reactive		Reactive-Predictive		Deliberative		Deliberative-Predictive	
	0	5	0	5	0	5	0	5
1	0	1	0	1	0	2	0	0
2	0	1	0	1	0	0	0	1
3	0	0	5	6	0	0	1	0
4	0	0	2	1	0	0	7	0
5	0	0	1	1	0	0	3	0
6	0	0	0	0	0	0	5	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	3	0
9	0	0	0	1	0	0	4	0
10	0	0	0	0	0	0	7	0
11	0	0	0	2	0	0	3	0
12	0	0	0	1	0	0	7	0
13	0	0	0	1	0	0	7	0
14	0	0	0	1	0	0	6	0
15	0	0	0	1	0	0	7	0

the predicting agents to overcome the advantage held by the deliberative agents in the one-agent task, and allows deliberative-predictive agents to perform significantly better. Furthermore, increasing the total number of agents just one more to three is enough to allow the reactive-predicting agent to overtake the deliberative agents. From that point on, both types of predicting agents are significantly better than both non-predicting agent types. Observations of all four kinds of agents reveals that agents that do not predict the behavior of others often enter into conflicts over a collection item. If both arrive at the same time, neither is able to get close enough to the item to collect it without triggering its reflex to avoid colliding with the other agent. Both agents move away, each being repelled by the other agent, but when the reflex behavior concludes (a random interval between five and fifteen cycles) they return, often triggering a similar conflict.

This is where the value of cooperation arises. Predictive agents avoid such conflicts and move on to the next item, leaving the item to be collected by the agent nearest to it. Note that there is no altruistic motive here; agents decide to focus on other items because their chances of acquiring an item that a closer agent is pursuing are slim. Therefore, the selfish course of action is to move on to another item. Cooperation emerges via this simple mechanism without the agents knowing that they are cooperating.

Another point of interest is found when the number of collecting agents exceeds ten: The curve of the graph in Figure 3 levels off, indicating that the benefit of additional agents is minimal for this size task. This is fairly intuitive; with more agents than items to be collected, many agents will serve no purpose and might, in fact, do more harm than good by getting in the way of other agents. Of greater interest is the fact that the performance of reactive and deliberative agents becomes nearly identical for numbers of agents over eleven. While the difference in performance between these agent kinds was significantly different only for environments containing six, seven, and eight agents, they are now virtually the same in absolute terms. This suggests that what little benefit is to be gained with deliberative architectures in tasks like these is lost when the environment becomes crowded, perhaps because with more agents around it becomes difficult to execute even the simplest plan without finding another agent in the way.

Finally, it is interesting to note that the reactive-predictive agents performed nearly as well as the deliberative-predictive agents on this set of tasks, but with a much higher degree of reliability. The more complex agents averaged around three or four failures out of forty experimental runs in environments with more than three agents, whereas the simpler reactive agents were reliable in most of those environments. This might suggest that the advantage actually rests with the simpler architecture. However, given the fact that the predictive architectures were the only ones to fail any task, it seems likely that there is some subtle limitation in the algorithm in Figure 3.

The 5-obstacle results presented in Figure 4 are very similar to the 0-obstacle results; the obstacles make the task more difficult, which is reflected in the overall poorer performance of all agents. The relative performance difference between reactive-predictive agents and deliberative agents has gotten smaller. This is because of the deliberative agents' ability to plan routes around obstacles rather than trying several angles until one is found, and so we expect that trend to continue.

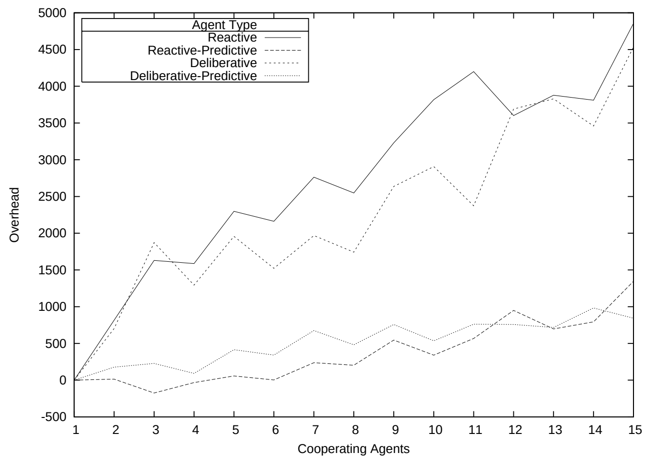

Figure 5. "Coordination" overhead

6 Discussion and Conclusion

One of the most interesting outcomes of this work is the emergence of what could be called "coordinated behavior" among independently acting agents using very simple predictive mechanisms. Of course, if one wants to characterize coordination as "the process by which an agent reasons about its local actions and the (anticipated) actions of others to try and ensure the community acts in a coherent manner" [5], then these agents' actions are not coordinated, as predictive-reactive agents do not reason about anything (after all, they are still reactive in the sense defined previously, but merely filter their perceptual input using a simple criterion).

Regardless of one's definition of "coordination", the experiments in the previous section demonstrate that predictive-reactive agents (i.e., reactive agents equipped with the a "perceptual filter") perform as well as "coordinated deliberative agents" that reason about other agents' intentions and maintain representations of various environmental states.

It is interesting to look at the results also from another perspective. If we define the notion of "cooperation overhead" in a task as the number of agents of a group times the time it takes them to complete a task minus the performance time of a single agent, then reactive-predictive agents have a lower overhead than the predictive deliberative agents, sometimes even significantly lower (see Figure 5), even though their performance is slightly worse. What seems to crystallize is a "region" of performance as determined by the number of collectible items in the environment and the number of agents in a group, where simple reactive-predictive agents are clearly superior to deliberative-predictive agents especially once the cost of performing deliberative processing is taken into account. However, we would like to point out that at this point we see the need for many more experiments with different reactive and deliberative control systems in different environments to be able to assess the potential of simple reactive mechanisms for multi-agent task and coordination. For example, we conjecture that communication among deliberative agents would not yield significantly better results for them, a claim that needs to be verified experimentally. We are also working on a more detailed analysis of the notion of "cost of a component in an agent architecture", which will allow us to get a finer-grained break-down of the net benefit of different functional components with respect to the overall agent behavior.

References

[1] E. G. Araujo and R. A. Grupen. Learning control composition in a complex environment. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 1996.

[2] R. C. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotic Research*, 8(4):92–112, 1989.

[3] J. Carmena and J. Hallam. Improving performance in a multi-robot task through minimal communication. In *Proceedings of the 7th Symposium on Intelligent Robotic Systems (SIRS)*, 1999.

[4] A. Drogoul and J. Ferber. From Tom Thumb to the Dockers: Some experiments with foraging robots. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, 1992.

[5] N. R. Jennings. Coordination techniques for distributed artifi cial intelligence. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 187–210. Wiley, 1996.

[6] D. McFarland. Towards robot cooperation. In *From Animals to Animats 3. Proc. of the Third International Conference on Simulation of Adaptive Behavior*, 1994.

[7] J. Pearl. A^*_ϵ—an algorithm using search effort estimates. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 4, pages 392–399, 1982.

[8] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 1995.

[9] M. Scheutz. The evolution of simple affective states in multi-agent environments. In D. Cañamero, editor, *Proceedings of AAAI Fall Symposium*, pages 123–128, Falmouth, MA, 2001. AAAI Press.

[10] M. Scheutz and P. Schermerhorn. Steps towards a theory of possible trajectories from reactive to deliberative control systems. In R. Standish, editor, *Proceedings of the 8th Conference of Artificial Life*. MIT Press, 2002.

[11] E. Spier and D. McFarland. Possibly optimal decision making under self-suffi ciency and autonomy. *Journal of Theoretical Biology*, 189:317–331, 1998.

[12] E. Stergaard, G. Sukhatme, and M. Mataric. Emergent bucket brigading - a simple mechanism for improving performance in multi-robot constrainedspace foraging tasks.

In *Proceedings of the 5th International Conference on Autonomous Agents*, May 2001.