Emergent Coordination through the Use of Cooperative State-Changing Rules

Claudia V. Goldman
Jeffrey S. Rosenschein
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel
clag@cs.huji.ac.il, jeff@cs.huji.ac.il

Abstract

Various researchers in Distributed Artificial Intelligence (DAI) have suggested that it would be worthwhile to isolate "aspects of cooperative behavior," general rules that would cause agents to act in ways conducive to cooperation. The hypothesis is that when agents act in certain ways (e.g., share information, act in predictable ways, defer globally constraining choices), it will be easier for them to carry out effective joint action. Another kind of cooperative behavior, less explored in the literature, is when agents independently alter the environment to make it easier for everyone to function effectively. Cooperative behavior of this kind might be to put away a hammer that one finds lying on the floor, knowing that another agent will be able to find it more easily later on.

In this paper we are concerned with cooperation of this latter type, state-changing behavior that improves the environment for everyone. We examine the effect that a specific "cooperativity rule" has on agents in the multi-agent tileworld domain. This domain is one in which agent actions are highly constrained, and the potential for cooperative behavior is great. We present a sociability rule by which agents are encouraged to increase tiles' degrees of freedom, even when the specific tile in question is not involved in an agent's own primary plan. The amount of extra work that an agent is willing to do in order to free these non-primary tiles is captured in the agent's *cooperation level.*

Results from simulations are presented, comparing the amount of effort expended by agents engaged in tileworld hole-filling with and without this rule of cooperation. Comparisons are also made with the corresponding optimal multi-agent plans. It is shown that a relatively simple, easily calculated rule can be used to greatly improve global system performance. Coordination emerges from agents who use this rule of cooperation, without any explicit coordination or negotiation.

1 Introduction

Distributed Artificial Intelligence (DAI) is concerned with effective agent interactions, and the mechanisms by which these interactions can be achieved. Researchers in DAI have taken many approaches to this overall question, considering in particular explicit coordination and negotiation techniques [15, 10, 9, 2, 8, 4, 16, 17, 7], as well as implicit modeling of other agents beliefs and desires [5, 6].

There have also been repeated attempts by researchers to establish norms of cooperative behavior, general rules that would cause agents to act in ways conducive to cooperation. The search space for multi-agent action is large, and cooperative behavior on the part of agents would ideally act to limit this search space. These investigations into cooperative behavior have been both formal and informal. They have generally taken the approach of shaping agents' plans in particular directions, such that other agents could interact appropriately. An agent that acts predictably, shares its information, and defers globally constraining choices as long as possible, will be an easier one with which to coordinate.

Work in this area includes early research by Davis and his colleagues at MIT [3], and some of the RAND work on cooperative behavior in the air traffic control domain [11]. More recently, Tennenholtz, Shoham, and Moses have considered how social laws for artificial agent societies could be developed and evaluated [18, 14].

While these streams of research have considered how agents' plans could be adapted to maximal cooperative effect, we take a different approach to the question. Instead of asking how an agent might temper its own goal-satisfying behavior to be cooperative, we ask how an agent might *transform the world* in a cooperative manner, at the same time that it is pursuing its own goal. An agent that picks up litter while going to mail a letter is improving the state of the world while carrying out its goal; the cost to this agent may be minimal, compared with the cost of sending out someone special to collect that litter. Another example would be an agent that puts away a hammer that it finds lying on the floor, knowing that it will be easier for other agents to find the hammer later on. The basic principle is that each agent is willing to expend a limited amount of extra effort to change the state of the world in ways that tend to be beneficial.

In Section 2 we present the domain we use for this paper, the multi-agent tileworld. We also present a meta-rule that encourages cooperative behavior on the part of agents, and the concept of *cooperation level* that determines how much extra work agents are willing to take upon themselves. In Section 3 we present the results of simulation experiments that compare various scenarios in the multi-agent tileworld, focusing on the amount of work agents perform when there is no coordination, when the coordination emerges implicitly from the cooperative meta-rule, and when an optimal plan has been formulated cooperatively or by a central planner. Finally, in Section 4, we consider some of the issues this cooperative meta-rule establishment has on cooperative problem solving systems versus the issues it raises for multi-agent systems.

2 Tileworld Interactions

2.1 The Domain

We will be considering agent interactions in a multi-agent version of the Tileworld [13] (see Figure 1).

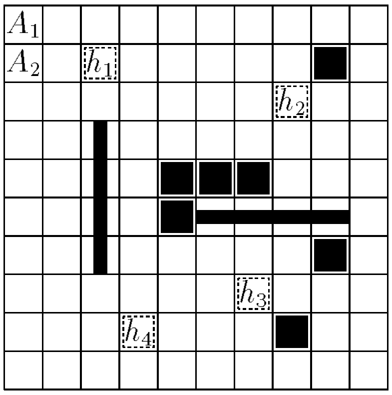

Figure 1: The Multi-Agent Tileworld

We use a simple variation of the multi-agent tileworld introduced in [19]. A hole in the grid is represented by a dashed line surrounding the letter h_i. Tiles are represented by black squares (■) inside the grid squares. Obstacles are represented by thick black lines (▮).

Agents can move from one grid square to another horizontally or vertically (unless the square is occupied by a hole, an obstacle, or another agent, i.e., two agents cannot occupy the same grid square at the same time). The domain is discrete, in that agents and tiles can be in one grid square or another at time t_i, but not partially in one grid square and partially in another. While moving, the agent can push a tile in the direction of movement, unless it would then run into an agent or an obstacle; tiles cannot be pulled. When a tile is pushed into a hole, the hole is filled and becomes navigable as if it were a regular grid square. The domain is static except for changes brought about by the agents. When there is a potential collision between two objects that are attempting to move into the same grid square at the same time (i.e., agent and block, or block and block), neither motion can be performed. When two *agents* attempt to move to the same location, one of the agents will be arbitrarily allowed to move into the spot.

In the original Tileworld, there was only one agent. We are interested in the case where two or more agents are pursuing their goals concurrently within the same grid. Agents are denoted by A_i. The cost of moving from one square to another is 1, whether or not the agent is pushing a tile or chain of tiles.

The agents have been programmed to roam the grid and push tiles into holes. We consider two ways in which the agents decide which tile to go after. In one variation, they choose the closest (Manhattan distance) tile, compute and traverse their path to it, then push it to the hole that is closest to the tile (again, Manhattan distance). The computation of distance is a (lower-bound) heuristic, since it doesn't take into account barriers, other agents, and tiles (but it is quick to compute).

In the second variation, agents are assigned tiles and holes arbitrarily by the programmer(s), and stop when they finish their assignments.

2.2 Strongly-Coupled Interactions

The goals of various agents in the multi-agent tileworld appear to be highly interconnected, primarily because the constraints on movement are so severe. Agents can only push (not pull) tiles, need to laboriously go around barriers to get into position, and need to be positioned precisely in order to push the tile into a hole. The number of steps that an agent has to carry out to get a tile into a hole, even when they are near one another, can be very large just because of the positioning requirements.

Consider, for example, the simple interaction shown in Figure 2 (a variation on an example from [19]).

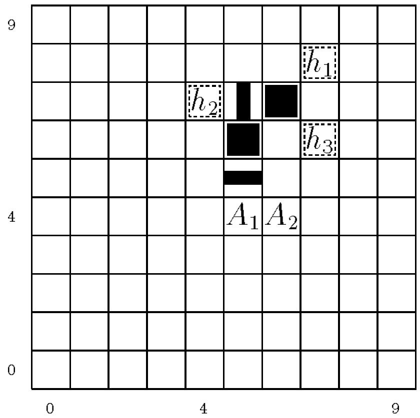

Figure 2: Strongly-Coupled Interactions

For the moment, let's assume that agent A_1 wants to fill holes 1 and 2, and that agent A_2 wants to fill holes 2 and 3 (this assumes that the agents were assigned these goals *a priori*). For either agent to accomplish its goal, it would need to carry out a large number of movements to position the tiles and itself appropriately. For example, for A_1 to fill its holes alone, it would need to move 17 times (this assumes A_2 is not in its way). Similarly, A_2, alone in the world, would need to move 26 times to fill its holes. Working together, however, they could satisfy A_1 completely (and A_2 partially) by total movement of 12,[1] or alternatively satisfy A_2 completely (and A_1 partially) also by total movement of 12.[2]

The highly constrained nature of the multi-agent tileworld provides ample opportunity for cooperative behavior, as the above example shows (e.g., one agent pushes a tile away

[1] A_2's part=(6,5); (6,6); (5,6); (6,6); (6,7); A_1's part=(4,4); (4,5); (4,6); (4,7); (4,8); (5,8); (6,8).

[2] Agent A_1 does 8 while agent A_2 does 4. A_2's initial part=(6,5); (6,6); (5,6); A_1's part=(4,4); (4,5); (4,6); (4,7); (4,8); (5,8); (6,8); (6,7); A_2's final part=(6,6).

from a barrier, while the other then pushes it in a perpendicular direction). However, *finding* complicated, optimal plans (like the one above) is a very difficult task. The amount of search that would be required to identify these clever interactions is huge.

Instead, we examine the possibility that the cooperative behavior exhibited in the above example can be approximated by giving the agents an inclination towards sociable behavior. We want to identify a rule in the multi-agent tileworld that causes the agents to help one another implicitly, without having to search for an optimal multi-agent plan. Ideally, when the agents are acting sociably, their combined activity will approach the optimal solution that they would have found had they carried out full multi-agent planning, but at a fraction of the computational cost.

2.3 A Rule for Sociable Behavior in the Tileworld

In general, a rule for sociable behavior can take several forms. In the simplest case, an agent may have two courses of action that he perceives as equivalent; if other agents would prefer him to carry out specifically one of those courses of action, he might do so in order to be cooperative (this captures the notion of pareto optimality, where agents prefer pareto optimal solutions over non-pareto optimal solutions). Often, however, we may design our agents to actually carry out extra work, in order to improve the environment for other agents. The amount of extra work that they carry out is subject to the designer's discretion.

The idea of this inclination towards bettering the environment, even if one doesn't benefit directly, can be thought of as a "meta-rule" that is imposed on the agent's normal behavior by its designer. Obviously, if one designer is building all agents, as in a cooperative problem solving scenario, then such a cooperative meta-rule will have clear utility if it improves overall system performance. If separate designers, with separate goals, are building the agents, there may still be justification for their putting in such a meta-rule, under certain circumstances; we discuss this issue below in Section 4.

In the tileworld domain, agents try to fill as many holes as they can with tiles that are spread throughout the grid. The holes themselves may have a worth value, in which case each agent might be trying to maximize its utility (points from each hole filled). Since movements on the grid have an associated cost, it is better for every agent to have tiles freely movable, i.e., less constrained by the barriers, so that the tiles can be pushed into holes more easily.

Each tile in the tileworld has a *degree of freedom* associated with it, the number of directions in which the tile can be pushed. The degree of freedom of any tile in our domain can be zero, two, or four.

Whenever the tile is trapped between barriers, so that there is no way for an agent to push it, the degree of freedom is zero. Figure 3 shows an example.

Generally, when a tile is next to a barrier, its degree of freedom is two. It can be pushed to the right or to the left if the barrier is horizontal, and up or down if the barrier is vertical. See Figure 4 for an example.

The last case is when the tile is not next to a barrier, nor at the boundary of the grid. In that case, it can be pushed in all four directions (left, right, up, and down). See Figure 5.

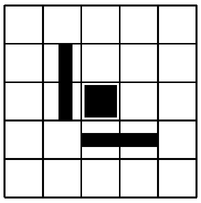

Figure 3: Zero Degrees of Freedom

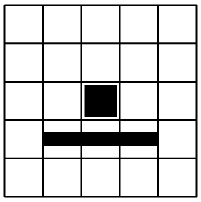

Figure 4: Two Degrees of Freedom

When we want agents to act cooperatively, we can induce them to free tiles; whenever an agent frees a tile, its degree of freedom changes from two to four. Acting cooperatively in this domain means that agents will act to increase the number of tiles with degree of freedom of four in the grid. We will call a tile with degree of freedom two "constrained," while a tile with degree of freedom four we will call "free."

The concept of "degrees of freedom" as a measurement of how easily goals can be achieved is specific to the tileworld, but the general principle is applicable in other domains as well. For example, in the common Blocks World domain, it is generally desirable to have blocks be clear; cooperative agents would act so as to clear blocks, making the achievement of others' goals easier. The key point here is not the specific degree of freedom concept, which is appropriate for this tileworld domain (and some others). It is, instead, to identify the aspect of a domain's configuration that allows goals to be achieved more easily. Cooperative agents are those that tend towards moving the world into these more conducive, less constrained configurations. In the tileworld, these are agents that are pushing tiles away from barriers.

We can evaluate the current state of the world according to this parameter, how many tiles have degree of freedom of four. In general, a world with more freed tiles is better than one that has constrained tiles. Of course, it may not be cheap for an agent to free a tile. On the other hand, it may be possible to exert a small amount of extra effort, free a tile, and save another agent a large amount of work. The main idea is that if agent A_i is on its way to push a tile into a hole, and there is a constrained tile close by that can be freed, then the agent will free the tile.

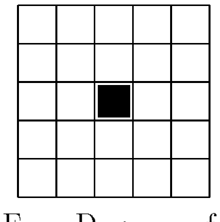

Figure 5: Four Degrees of Freedom

This kind of cooperation does not require that the agents communicate, nor that they have a model of the other agents' plans, beliefs, or goals. Cooperation will emerge out of the sociable behavior of the individual agents, not through explicit coordination. In this paper, agents are assumed only to know the current status of the grid. They have full information about *external* aspects of the domain, seeing where there are holes, tiles, barriers, and agents.

2.4 Cooperation Level

How much extra work should an agent be willing to do in order to act sociably? In other words, how close does a constrained tile need to be in order for an agent to go out of its way and free it?

We can imagine a rectangle that encompasses all minimal paths from an agent to its goal. We consider a legal path as one that never backtracks. In general, the *extra work* that the agent may do is considered to be the movement outside one of its minimal paths and back, the extra movement carried out so as to free a tile. The sum of the lengths of these extra paths is the amount of work the agent is ready to do so as to help others operating in the same environment. We call this amount of extra work the *cooperation level* of the agent.

In the current implementation, the cooperation level is defined when the agent is created, and it is not changed during the execution of the simulation. In each tick of time, the agent tries to free a constrained tile that is different from the one it is pushing into a hole, such that the sum of the costs of the paths from the agent location to the constrained tile, freeing the tile, and going back to its original position is not larger than its predefined cooperation level.

We can distinguish between different kinds of agents, based on their cooperation level being positive, zero, or negative. Cooperative agents are ones with positive cooperation levels. Agents that do not cooperate, and only do their own work, have cooperation levels of zero. Finally, there could exist destructive agents, whose cooperation level is negative. These destructive agents would do their own work, but instead of also freeing constrained tiles when they can, they will try to constrain tiles, pushing them back to their closest barriers. Such a destructive agent, of course, would also be harming itself (by expending energy to no private benefit). In the current simulations, only agents of the first two classes were designed.

3 Experiments

Several simulations were carried out using the MICE distributed agent testbed [12]. We here describe the simulations, and the results of agents carrying out tasks with and without the cooperative meta-rule described above.

3.1 Scenario One

Let's review the example of Figure 2. The starting state can be seen again in Figure 6.

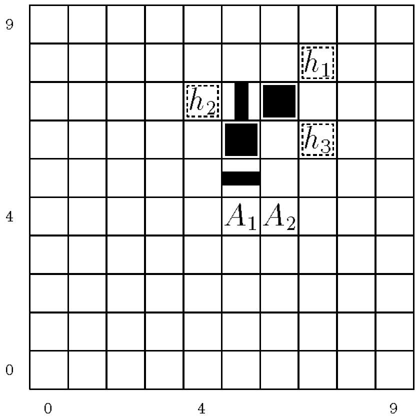

Figure 6: Scenario One

3.1.1 Push Closest Tile First

The aim of each agent in this example (in contrast to the presentation above) is to fill as many holes as it can. Both agents are by design first trying to get to their closest tile. We analyze initially the case where both agents act without any coordination. In this case, the closest tile for both agents is identical, and is the one located at position (5,6). The closest hole for this tile is h_2. Therefore, A_2 moves north while A_1 moves south to go around A_2. A_2 gets to the tile first, and pushes it west. It then leaves the tile (going east) so as to circumvent the barrier to its south.

Now, A_1 is closer to the tile, and eventually pushes it north, into the hole h_2. A_2 still moves towards this tile, until the agent finds that the other tile is closer to it. The distance between h_1 or h_3 to the second tile is the same ($\sqrt{2}$). h_3 was chosen by the program (an arbitrary choice based on the simulation's ordering of the holes). For A_2 to be able to push the second tile into h_3 it has to go around, to the right of the remaining holes. After A_1 has pushed the first tile into h_2, it also starts advancing towards the other tile. A_1 gets to the tile first, pushes it south and goes around (north and west) to be able to push it east into h_3. A_2 does the same work, but lags behind A_1 by one step (so through the end of this simulation A_2 wastes his movements). By the time the second tile has been pushed into hole h_3, the agents have together moved 34 steps (each one 17).

We then re-ran the simulation with the agents using the coordination meta-rule in addition to trying to fill holes. Using the same initial state as input (and giving A_2 a cooperation level of two, A_1 a cooperation level of 0), the agents fill two holes after a combined 24 steps (A_1 does 11, and A_2 does 13). Both agents start operating as in the former simulation, but when A_2 gets to the location (6,6), it first frees the constrained tile that is not the one it intends to push into the hole (i.e., the tile located in (6,7)), then goes back one step to the position it was when it decided to free the tile and goes on pushing the first tile to the west. From here on, A_1 does the rest of the work very easily. It goes north to put its closest tile into h_2, then goes up two more steps, arriving at the horizontal row where the second tile has already been pushed by A_2. A_1 advances to the east until it fills h_1. Thus the cooperative meta-rule has engendered implicit cooperation.

The optimal solution, where coordination is explicit (perhaps established by a central planner or by a distributed planning or negotiation algorithm), will give the same solution as in the implicit coordination setup, but the agents will not do any wasted steps (e.g., A_2 will not run after tiles that A_1 will eventually push into holes). Therefore, the optimal solution will consist of only 12 steps (A_2 does 5 and A_1 does 7).

3.1.2 Push Assigned Tile

We consider this scenario again, but with agents having been assigned the tile they are going to push (rather than simply going after the closest tile), and also being told which hole to push it into (this corresponds to individually motivated agents with prior goals). First let us consider the case where there is no coordination at all.

In the first case we assigned A_1 the tile located at (6,7) and the hole h_1, and A_2 the tile located at (5,6) and the hole h_2. A_1 does 15 steps and A_2 10 steps. Each basically works as if it were alone in the world.

If we run the same simulation where implicit coordination can emerge (from use of the coordination meta-rule), A_1 goes 11 steps and A_2 goes 12 steps. The two extra steps of A_2 have saved A_1 4 steps.

The optimal plan in this example (given the *a priori* tile and hole assignments) consists of A_1 doing 9 steps and A_2, 12. A_2 goes up, freeing both tiles. The tile in (6,7) it frees as part of its cooperative activity, but the second tile is the one it is actually going to push into h_2. A_1 does one step south to let A_2 pass, and then goes all the way up from its starting position to its assigned tile.

In the second case, the input consisted of assigning A_1 the tile located in (5,6) and hole h_2, and assigning A_2 the tile in (6,7) and hole h_1. In both simulations (the one with no coordination, and the one with implicit coordination) both agents went 25 steps (A_1 goes 13 and A_2 12). In the implicit coordination case, the cooperation level of two that A_2 has does not help overall performance. The agent decides in which direction to free a constrained tile based on the number of free tiles and the number of holes that are empty in each candidate direction. In this case, A_2 intended to free the tile in (5,6) by pushing it to (6,6), but to achieve that, A_2 would have had to go around to location (4,6), and this path would have cost more than two. The optimal solution in this example has A_1 going 3 steps (to move from its starting location to (4,6)) and A_2 going 10 steps to move upwards, push its tile north, go down one step, push the other tile to the east, wait till A_1 pushes it into h_2, and then pass through these locations to its tile. One additional step then pushes it into h_1.

3.2 Scenario Two

In this scenario (see Figure 7), A_1 is very close to both tiles and holes. A_2, while not very far, has its movement severely constrained by barriers.

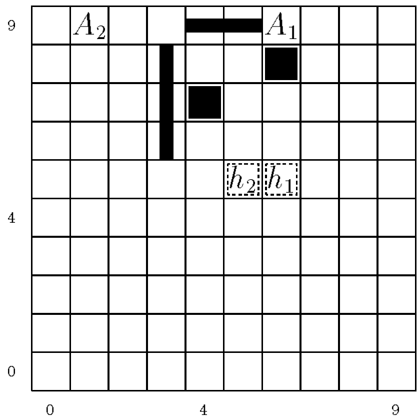

Figure 7: Scenario Two

3.2.1 Push Closest Tile First

In this scenario the tile situated at (6,8) is the closest to A_1, and the tile at (4,7) is the closest to A_2. When the agents don't cooperate, each one will move towards its closest tile. The agents conflict when both want to push the tile located at (4,7), that has been pushed by A_1 to (4,5). The collision function used by the program moves the agent that has caused the conflict back to its original place. In this case, both agents are conflict agents and the program chooses the first one according to the order in which the agents are invoked by the simulation. In this case, A_2 is left at its location while A_1 pushes the tile south. The total amount of work done by the agents is 27 steps, 15 done by A_1 and 12 by A_2.

Both agents can do just 22 steps if they cooperate implicitly and A_1 is given a cooperation level of 8. A_1 starts pushing its closest tile one step, then it frees the second tile located at (4,7), leaving it at (4,5), and continues with its first tile. A_2 now finds its closest tile in a better location, so 4 extra steps of A_1 have saved one step for A_2. They also didn't have any conflicts that could have cause them to waste steps.

The optimal solution in this case is when A_1 does 11 steps, pushing the tile located at (6,8) into h_1 and freeing the second tile to (4,5), and A_2 does only 7 steps to push the tile at (4,5) into h_2.

3.2.2 Push Assigned Tile

Consider the input in which we assigned the tile situated at (6,8) and hole h_1 to A_1, and the tile at (4,7) and hole h_2 to A_2. If the agents work without coordinating themselves, each one will only care about its own goal, yielding a solution of 26 steps (3 for A_1 and 23 for A_2). However, when we let the agents cooperate implicitly (through the coordination meta-rule; A_1 was given a cooperation level of 8), our simulation returned a solution of only 18 steps. A_1's high cooperation level caused it to push its tile one step, leave it, free the other tile (pushing it towards location (4,5)) and then go back to its assigned tile, finally pushing it into hole h_1. A_2 has only to move to location (3,5) to push its tile, with no need to reposition itself during its plan. The implicit coordination simulation actually causes the agents to carry out the optimal solution in this scenario.

3.3 Scenario Three

Consider now the scenario presented in Figure 8.

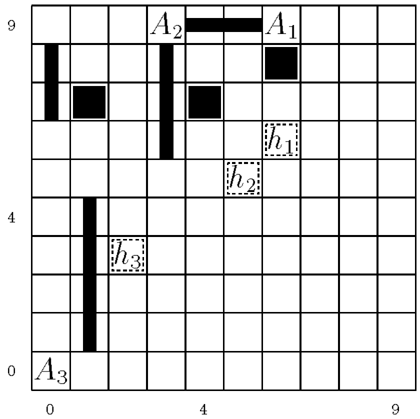

Figure 8: Scenario Three

3.3.1 Push Closest Tile First

If we let the agents operate in this scenario without any coordination, A_1 will succeed in putting its closest tile (from location (6,8)) into the hole h_1 and starts working to push the tile at (4,7) into h_2. At the same time A_3 advances towards this tile and conflicts with A_1. The conflict is solved by letting A_3 push the tile into h_2. A_1 has gone 16 steps, A_2, 17 steps and A_3, 15 steps.

However, when we run the same input with the cooperation meta-rule (implicit coordination) we get a solution of 37 steps instead of 48. A_1 was given a cooperation level of 8 and went 11 steps, A_2 was given a cooperation level of 4 and also went 11 steps, and A_3 has a cooperation level of zero and went 15 steps. A_1 starts pushing its closest tile (6,8) to h_1; on its way it cooperates by freeing the tile located at (4,7). A_2 moves towards the tile that originally was located at (4,7) and on its way it frees the tile situated at (1,7) thus making A_3's work easier.

The optimal solution is a plan with 7 steps for A_1, 9 steps for A_2, and 12 steps for A_3.

3.3.2 Push Assigned Tile

We considered a similar scenario where we pre-assigned the tile at (6,8) and hole h_1 to A_1, the tile at (4,7) and h_2 to A_2, and the tile at (1,7) and h_3 to A_3. The same cooperation levels as before were used. The agents then moved 50 steps in the non-coordination simulation, and only 34 steps in the implicit coordination simulation.

Without coordination each agent pushed its assigned tile, and stopped afterwards.

When the agents are designed to cooperate, A_1 freed the tile that had been assigned to A_2 and A_2 freed the tile that had been assigned to A_3. Therefore, A_1 went ten steps instead of two, but A_2 went only 11 steps instead of 23, and A_3 went only 13 steps instead of 25.

The optimal solution for this case consists of A_1 going 10 steps, as in the implicit coordination simulation. A_2 will go only 9 steps, first freeing the tile in (1,7), then moving to its assigned tile and pushing it into h_2. A_3 will do 12 steps.

3.4 Scenario Four

3.4.1 Push Assigned Tile

The scenario in Figure 9 describes an environment in which A_1 is assigned the tile at (2,7) and hole h_1, A_2 is assigned the tile at (8,7) and hole h_2, and A_3 is assigned the tile at (12,7) and hole h_3.

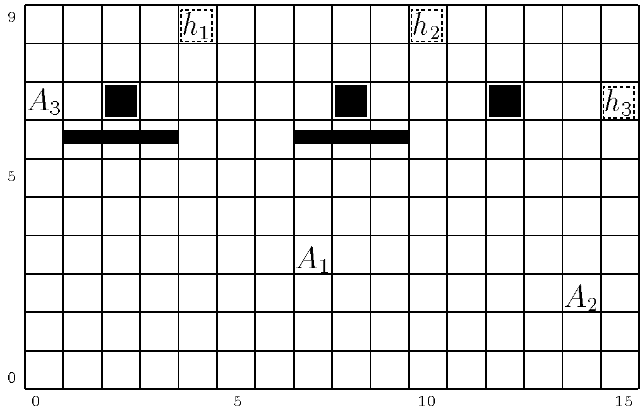

Figure 9: Scenario Four

A_3 intends to push its tile into h_3, and on the way to its tile, the agent passes by the tiles assigned to the other agents. It will benefit A_2 and A_1 if A_3 would do extra work by freeing their tiles from the barriers so that neither A_2 nor A_1 will have to reposition themselves, and will be able simply to push their tiles north.

The result after running the simulation without coordination was 22 steps for A_1, 24 steps for A_2 and 16 for A_3. When the simulation was run with the cooperation meta-rule (implicit coordination), giving A_3 a cooperation level of 4, A_1 had to go 11 steps, A_2 28 steps, and A_3 20 steps. Actually, A_3 helped only agent A_1 in the implicit coordination simulation. After freeing the first tile, A_3 chose a new path to its own primary tile, and its cooperation level was not enough to free the second constrained tile.

4 Sociability in Cooperative Problem Solving vs. Multi-Agent Systems

The idea that agents can be arbitrarily made more cooperative is one well-suited to cooperative problem solving systems. There, a single designer is building the entire system of agents, and there is at least an implicit notion of global utility to be maximized. If one can show that a given meta-rule of cooperation leads to better system performance, that should be sufficient to incorporate it into the agents' programs.

In multi-agent systems, however, we are assumed to have agents that were individually designed. There is no notion, even implicit, of global utility, and it may not be clear at first why agents should bother to be cooperative with one another, unless they themselves will benefit from their behavior.

Here, precisely, is the paradox, well-known in economics and game theory. If *all* agents were cooperative, each agent would benefit—in our tileworld examples, each agent will be delighted to find tiles unconstrained, and will carry out his own goals more easily because of it. But *each* agent remains unmotivated towards cooperative behavior; each agent would prefer that the others be cooperative, while it is not and saves himself the extra work. This is the classic "free rider" problem. In game theory terms, the strategy of cooperation is not stable; assuming everyone else is being cooperative, I benefit by not being cooperative.

The solution to these problems often relies on penalty mechanisms of one sort or another. For example, it may be possible for agents to act uncooperatively in encounters with known uncooperative agents, while acting cooperatively with cooperative agents (a strategy similar to TIT-FOR-TAT [1]). Alternatively, it may be possible to alter the cooperation rule itself so that it becomes stable. In any case, the rule as presented above is not appropriate as it stands for multi-agent systems, and its use in such systems is a topic for future research.

5 Conclusions

We have presented a "rule of cooperative behavior" that can be imposed on agents in the multi-agent tileworld to significantly improve overall system performance. Agents are encouraged to move tiles away from barriers (even when these tiles do not contribute to their primary goal), as long as the amount of extra work required is not too great. The addition of this implicitly cooperative behavior does not require a great deal of extra computation on the part of agents, nor any communication whatsoever. Cooperation emerges cheaply from agents acting sociably, without the overhead of negotiation or coordination. Simulations were run that illustrated the benefits of this emergent cooperation. Although unlikely to produce optimal behavior (except by chance), the cooperative rule does improve performance greatly.

The principle of cooperative behavior extends to arbitrary domains, where system designers can identify aspects of global state that are generally desirable. In the tileworld, it is generally desirable that tiles be unconstrained by barriers. In the blocks world, it is generally desirable that blocks be clear. The designers of agents need to manufacture rules of sociable behavior that encourage agents to carry out state transformations that tend to be socially beneficial. There are other aspects of sociable behavior beyond the question of state transformations; for example, agents might choose their goals in a way that was socially cooperative. This paper, however, focused on the state-transformation issues of social cooperation.

This introduction of social behavior is clearly beneficial in cooperative problem solving systems, where a group of agents has been centrally designed. It is, in addition, an important possible component in multi-agent systems, where agents are individually motivated. In the latter case, however, it is necessary to consider the stability of the rule of cooperative behavior.

Thus, in the case of multi-agent systems, there needs to be further research into how agents will be motivated towards cooperation, and how the system will deal with the free rider problem. Additional future research will examine benefits to the system when the cooperation level of agents changes dynamically over time (for example, as a penalty mechanism aimed at uncooperative agents), and how the subdivision of labor (e.g., which agents move which tiles) might also be affected by cooperative meta-rules.

References

[1] Robert Axelrod. *The Evolution of Cooperation.* Basic Books, Inc., New York, 1984.

[2] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[3] Randall Davis. A model for planning in a multi-agent environment: steps toward principles for teamwork. Working Paper 217, Massachusetts Institute of Technology AI Laboratory, October 1981.

[4] Edmund H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, Boston, 1988.

[5] Michael R. Genesereth, Matthew L. Ginsberg, and Jeffrey S. Rosenschein. Cooperation without communication. In *Proceedings of the National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, Pennsylvania, August 1986.

[6] Piotr Gmytrasiewicz and Edmund H. Durfee. A logic of knowledge and belief for recursive modeling: Preliminary report. In *Proceedings of the National Conference on Artificial Intelligence*, pages 628–634, San Jose, California, July 1992.

[7] Sarit Kraus and Jonathan Wilkenfeld. Negotiations over time in a multi agent environment: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 56–61, Sydney, Australia, August 1991.

[8] T. Kreifelts and F. Martial. A negotiation framework for autonomous agents. In *Proceedings of the Second European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds*, pages 169–182, Saint-Quentin en Yvelines, France, August 1990.

[9] Kazuhiro Kuwabara and Victor R. Lesser. Extended protocol for multistage negotiation. In *Proceedings of the Ninth Workshop on Distributed Artificial Intelligence*, pages 129–161, Rosario, Washington, September 1989.

[10] T. Malone, R. Fikes, and M. Howard. Enterprise: A market-like task scheduler for distributed computing environments. In B. A. Huberman, editor, *The Ecology of Computation*. North-Holland Publishing Company, Amsterdam, 1988.

[11] D. McArthur, R. Steeb, and S. Cammarata. A framework for distributed problem solving. In *Proceedings of the National Conference on Artificial Intelligence*, pages 181–184, Pittsburgh, Pennsylvania, August 1982.

[12] T. A. Montgomery, J. Lee, D. J. Musliner, E. H. Durfee, D. Damouth, and Y. So. *MICE Users Guide*. Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan, February 1992.

[13] Martha E. Pollack and Marc Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings of The National Conference on Artificial Intelligence*, pages 183–189, Boston, Massachusetts, August 1990.

[14] Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the National Conference on Artificial Intelligence*, pages 276–281, San Jose, California, July 1992.

[15] Reid G. Smith. *A Framework for Problem Solving in a Distributed Processing Environment*. PhD thesis, Stanford University, 1978.

[16] K. Sycara. Resolving goal conflicts via negotiation. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 245–250, St. Paul, August 1988.

[17] Katia P. Sycara. Argumentation: Planning other agents' plans. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 517–523, Detroit, Michigan, August 1989.

[18] Moshe Tennenholtz and Yoram Moses. On cooperation in a multi-entity model (preliminary report). In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 918–923, Detroit, Michigan, August 1989.

[19] Gilad Zlotkin and Jeffrey S. Rosenschein. Compromise in negotiation: Exploiting worth functions over states. Technical Report 93–3, Leibniz Center for Computer Science, Hebrew University, 1993.