USC-CIT Technical Report 98-02

Benevolent Agents

Abdulla M. Mohamed and Michael N. Huhns

October 1998

Center for Information Technology
Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC, U.S.A. 29208
(803) 777-5921 or {mohamed,huhns}@engr.sc.edu
(803) 777-8045 FAX

Keywords: Benevolence, Software Agents, Multiagent Systems, Agent Behavior Testbed

Abstract

This paper describes an analysis of benevolent behavior among software agents. We first present a definition of benevolence that is appropriate for software agents. We then describe requirements for the structure and behavior of benevolent agents, and construct a simulator that can analyze and verify such requirements. We conclude with results from a set of simulations of benevolence.

1. Introduction

Software agents can exist alone or in a society. Depending on the characteristics of its member agents, the society can be either homogeneous or heterogeneous. Typically, each agent in a society can be characterized as having a list of goals or tasks that it will attempt to accomplish. Similarly, a society can be characterized as having a list of goals or a global "mission" list that it is attempting to accomplish, where each member agent contributes some effort to achieve part of the mission. The agents' goal list and the society's "mission" list are both usually ordered in term of importance and priority.

The interactions that an agent has with other agents, both direct and indirect, are determined by its characteristics. In addition, an agent's contribution to its society is controlled by its behavior. Autonomy, sociability, adaptability, friendliness, benevolence, and cognition are some examples of an agent's characteristics. Based on the agent's goal list and its society "mission" list, a combination of characteristics could be attributed to the agent.

Benevolence is the characteristic on which this paper focuses. A dictionary definition of benevolence is "The doing of a kind action to another, from mere good will, without any legal obligation. It is a moral duty only, and it cannot be enforced by law." When this definition is applied to a software agent, it means that a benevolent agent is an agent that helps others when it does not have to and there is no immediate reward or "benefit" to its action. A classic example of benevolence is *the mattress on the road*: a mattress in the middle of a road can cause a traffic jam, because vehicles will have to slow down to maneuver around it. It results in a delay for everyone. A benevolent agent will stop and move the mattress out of the way, so everyone can proceed on his or her way without any further delays. Such an action would cause the benevolent agent more delay than if it just tried to avoid the mattress like everyone else, and the agent receives no immediate reward or compensating "benefit" for its action.

Under these circumstances, why would an agent act benevolently and move the mattress? One possible motivation is that the benevolent action will help the overall society of which the agent is a member. Another motivation is that the benevolent agent might believe that other agents similarly would act benevolently in the future, thereby compensating the agent in the longer term. It is important to realize that a benevolent agent can only exist in an environment with other agents, not alone.

Software agents are unlikely to encounter mattresses, so where might a benevolent agent in a computational environment have an opportunity to behave benevolently? The agent could clean up stalled or failed transactions, close sockets that were left open by a process that terminated early, or remove locks set by failed or former processes. When it does not have either the authority or ability to take action, it can simply provide notifications to agents or systems that do.

2. Background

Sen [Sen 1996] investigated the circumstances in which one agent should help another agent perform a given task when the other agent requests help. The decision criterion is that this action should enable the agent who is conducting the help action to perform more effectively in the long run. For his experiment, Sen uses the principle of reciprocity, which means agents only help those agents who helped them in the past or can help them in the future. Sen's analysis and experiments show that reciprocal behavior improves the individual agent performance with the long run over the selfish behavior.

Cesta, Miceli, and Rizzo [Cesta, Miceli, and Rizzo 1996] believe that social agents, i.e., those who provide and ask for help, are the most successful agents. But their paper focuses on "how does the coexistence of exploiting agents, who ask for help but never give help, and the social agents, who give help no matter what, affect the performance of both the entire system and the social agents themselves?" Interesting simulation scenarios of agents with different attitudes were conducted on a grid-based testbed. The simulation results show that the robustness of the social attitude is confirmed. One interesting result is that it is less the presence of exploiters, than the absence of filters against exploitation, that can put the social system in danger. Another fascinating result is the relation between the robustness of the social strategy and its usefulness: the more robust the strategy is the more useful the social agents are to the entire system compared to the exploiting agents.

Hales [Hales 1998] states that elegalitarianism and cooperation between agents is not enough to achieve altruism. He also observes that human societies succeeded in controlling coordination and group organizational problems in too short a period of time to be accounted for by genetic evolution. His experiments conclude that an altruistic society is more efficient in terms of group computing than a genetic society.

Castelfrenchi and Conte [Castelfranchi and Conte 1996] discuss the notation of teamwork and join activity. The current notion of teamwork states that a team of agents have a joint persistent goal if they mutually believe that the goal is not reached yet, they mutually know they all want the to reach the goal, and it is true that until they come to believe either that the goal is reached or the goal will never be reached, they will continue to mutually believe that they each have the goal as a weak achievement with respect to the team. Castelfrenchi and Conte think that the current notion is not complete, because it lacks the concept of mutual dependence that makes the commitment to participate unmotivated. Thus, they introduce an alternative view of the teamwork that includes social dependence. This social dependence includes mutual and reciprocal dependence. Reciprocal dependence occurs when two agents realize each other's goals. One of the advantages of the new notion is that it eliminates competition between agents, since they mutually depend on each other.

3. Analysis of Benevolent Software Agents

The basic question we would like to answer is "When is benevolence useful for the agent and its society, and when is it not?" Benevolence is beneficial to a society as a whole, and thus to each of its members, when it leads to an overall improvement in efficiency or results, such as in the mattress situation. However, it can be harmful if agents spend all of their time in trying to perform benevolent actions and never make any progress towards their own goals. It can also be harmful if only one of a society's members is willing to take any benevolent action and the rest of the members are not. But if all or many of a society's members are willing to undertake benevolent actions for the goodwill of their society, then benevolent behavior will definitely be useful. For an individual agent, depending on its goals, benevolence might or might not be the appropriate behavior. For example, a broker "business" agent that needs to make the best deal on a contract will not take any benevolent action that will help its competitor agents. On the other hand, a search agent might perform a benevolent action such as updating some search engine information, which over time might reduce the overall traffic on the Internet, benefiting all of the Internet's users.

A benevolent software agent will have a list of goals that it needs to accomplish. In the same time, it will also work on achieving some of its societal missions. The actions taken by the agent should not in anyway have negative impacts on its societal mission. While the agent is working toward a goal, it might encounter a situation where a benevolent action is needed for the good of the others, but it is not part of the goal that the agent is driving toward. For example, an agent's main goal might be to clean a nuclear facility by picking nuclear waste up from the floor and dropping it at safe dumping areas. While the agent is carrying some waste and moving towards the dump area, it might encounter some obstacles. The agent will take a benevolent action by moving the obstacle out of the way. Such an action will cost the agent some time delay, because it could simply just avoid the obstacle. But this benevolent action will clear the way for other agents, so they do not have to waste their time trying to avoid it. Thus, this benevolent action helps all of the society's members as well as the benevolent agent itself in the long run.

The reward for a benevolent action is not immediate, and the results on the society will be observed over the long term. That is why a measurement of such an action is not straightforward. But in general, the benevolent actions should assist the society of agents to accomplish their objectives, and at the same time not prevent the individual agents from reaching their goals.

Using the above example, a measurement of the benevolent action of moving the obstacle might be improvement in the speed in which the agents will reach their goal state, i.e. depositing all waste at safe locations. Such benefits could be amplified when combining benevolence with other behaviors such as cooperation. Using the nuclear waste example, the agents might report the clear paths they took to the dumping area after moving all the obstacles out of the way as a cooperative and benevolent behavior. Such a combination of behaviors will result in creating obstacle-free roads for other agents that will ensure efficient transportation for the nuclear waste. Other characteristics could be also combined, but a careful consideration to the agent and its main societal goals must be taken into account to avoid any negative impact that might be caused by such a combination.

4. Agent Behavior Testbed

The Agent Behavior Testbed (ABT) is a tool we have constructed to simulate agent behaviors. ABT consists of agents, boxes (which could represent nuclear waste), and targets (drop zones for boxes). Each agent's main goal is to pick up the boxes in its environment and deposit them at one of the targets, generally the nearest target. This is an example of a task-oriented domain [Rosenschein and Zlotkin 1994], where each agent has all the resources it needs to achieve its goals. Each agent society can have its own behaviors and timing specifications. A testbed user can instantiate as many agents from a selected agent society as desired. Each agent is implemented as a Java thread, so agents execute concurrently. Agents follow a set of rules based on their selected behaviors. Based on the simulation result, the agents' behaviors can be modified and the same scenario can be run again to obtain comparative results.

Figure 1 shows the environment of ABT, which consists of an NxN grid of cells (default15x15). Each cell could have an agent (A#), a box ([]#), an agent carrying a box (A#[]#), or a target ((+)#), or it could be empty. The cell's label and color illustrate the content of the cell graphically. Figure 2 shows some of the menu items available for agent simulations. The "Insert" menu is used to insert boxes, targets, and agents of different types (different societies) by selecting the object and clicking on the desired cell where the object should be placed. The "Random" menu item is used to insert N numbers of a selected object at random locations, and "Previous Scenario" is sued to repeat the last simulation scenario. The "Property" menu has only one item, "agent", which is used to edit the properties of the selected type of agent – from the "Insert" menu. The "Simulation" menu contain two items, "Run" to run the current scenario, and "Reset" to reset the environment (floor).

Figure 1. ABT graphical user interface, showing an agent environment in the form of a grid, with cells of the grid occupied by either agents, boxes, targets, agents carrying boxes, or nothing

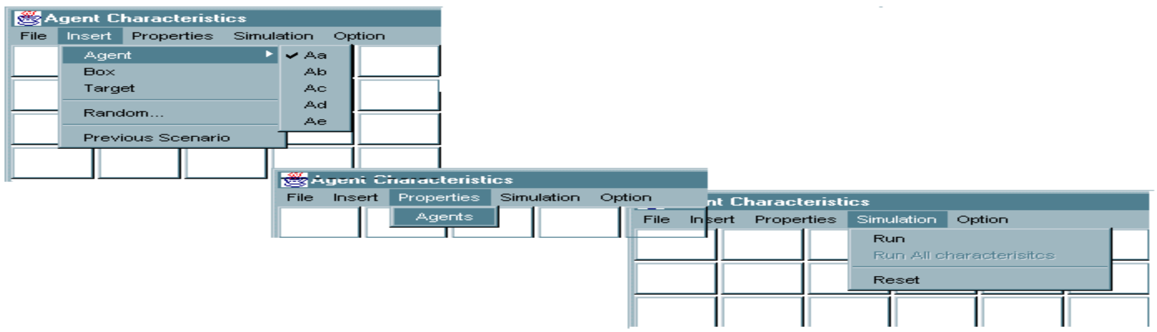

Figure 2. ABT menu items for specifying parameters of agents, their environment, and the simulation

When a simulation starts, all of the agents will try to achieve their main objective of moving all of the boxes to the appropriate targets. Agents do not know the locations of the boxes in advance; they only know the location of the target areas. So, agents will begin by searching their immediate neighborhood for boxes, starting at the top left corner of their neighborhood and ending at the lower right corner in a clockwise fashion. If no box is found, and agent will move to an adjacent location and repeat its search. Agents can only move one step at a time in any direction, vertical, horizontal, or diagonal. Once an agent locates a box, it will pick it up and move along the shortest path to the nearest target. Once a depositing operation is completed, the agent will reenter the floor, and start searching for other boxes. Agents will stop when all boxes are deposited at the assigned target areas. A simulation report will then be generated that includes the total amount of search, number of moves, and number of boxes picked up, dropped, and deposited. In addition, each agent's activities will be logged in a separate file.

Depending on the agent's characteristics, agents will behave differently. Figure 3 below shows the different characteristics that an ABT agent can have. Benevolent agents will move any obstacles they encounter out of the way, so the other agents do not have to avoid them. Cooperative agents will report the location of other boxes they encounter to the other agents. Another form of cooperation can be demonstrated when agents report on clear paths (roads) that lead to a target. These are marked with black cells on the grid. In addition, some agents could be smarter in that they will remember all the cells they have previously visited and try not to visit them again while searching for boxes. Agents could have any combination of the above behavior. All of the agent's behaviors are derived from the following rule list:

Picking up a Box

1. IF
   - Agent is not carrying a Box
   - Agent found a Box in its neighborhood

   THEN
   - Agent picks up the box (Agent is carrying a box)

Moving without a Box

2. IF

Agent is not carrying a Box
Agent does not know where Boxes are

THEN

Agent moves to a random empty neighboring

3. IF

Agent is not carrying a Box
Agent know where Boxes are

THEN

Agent moves toward the closest Box using the shortest distance path

4. IF

Agent is moving toward a Box
Agent faces another Box in its path to the target Box

THEN

Agent picks up the obstacle Box

Moving with a Box

5. IF

Agent is carrying a Box

THEN

Agent moves toward the closest Target

6. IF

Agent is moving toward a Target
Agent does not find any reported path that passes through its neighborhood that leads to the same Target

THEN

Agent moves to next cell in the shortest distance path

7. IF

Agent is moving toward a Target
Agent finds a reported path that passes through its neighborhood that leads to the same Target

THEN

Agent moves to next cell in the reported path

8. IF

Agent is moving toward a Target
Agent faces another Box in its path to the Target

THEN

Agent moves the obstacle Box from its path to a close cell to its original Box, then picks up its original Box and continues moving toward the Target

Moving with or without a Box

9. IF

Agent is moving toward a Box (not carrying a Box) or Target (carrying a Box)
Agent faces another agent in its path

THEN

Agent moves to a random empty neighboring cell

Depositing a Box at a Target

10. IF

Agent reaches Target

THEN

Agent deposits Box at Target
Agent reenters the floor at a random neighboring empty cell of the Target

Communicating with other Agents

11. IF

Agent moves to a new cell (carrying a Box or not)

THEN

Agent searches the new cell's entire neighborhood, and reports the location of any Boxes to the other Agents

12. IF

Agent reaches Target
Agent did not follow any reported path to the Target

THEN

Agent reports the path it followed to the target to all other Agents (clear path) starting from the point of Box pickup or from the last obstacle it avoided if it faced one.

Stopping

13. IF

Agent is not carrying a Box
All Boxes have been deposited

THEN

Agent stops

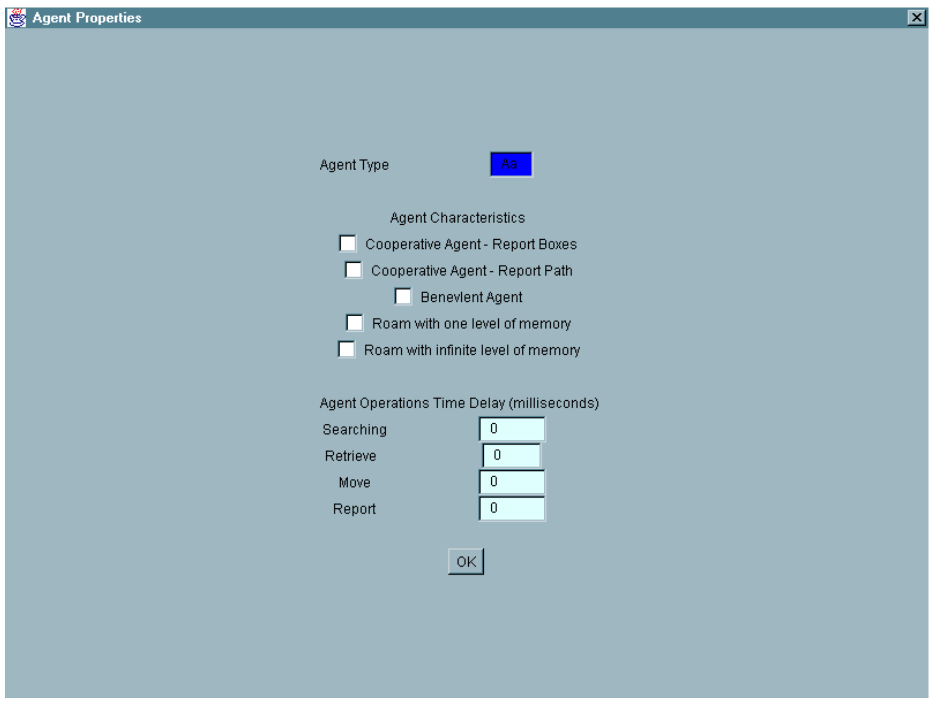

Figure 3. Agent property dialog box

To elaborate on some of the most important rules, let us use Figure 4. Each cell is labeled with its row and column number, by choosing the "Show Coordinates" from the "Option" menu. We illustrate rule # 1 by using agent1 (2,2) and Box1 (3,3). Agent1 will find Box1 in its neighborhood, so it will pick up box1, and then move to target1 using the shortest distance, that is, by travelling along the following path: (4,4), (5,5), (6,6). To demonstrate benevolent behavior based on rule # 8, we use agent1 (13,7), box2 (12,7), and box4 (10,7). Agent 1 will pick up box2 and move along a straight path to the target1, but then face box 4 along its path. As a benevolent agent, agent1 will drop box2 at (11,7), pick up box4, move it out of the way, pick up box1 again, and finally, move toward target1. The policy for moving the obstacle box out of the way is to not move it to a cell in the path to the target, try to move it to a cell that is close to the original box to eliminate extra steps, and if this fails, then move it to any of the other neighboring cells. In this case, agent1 will try to move box2 to (10,6), (10,8), (11,6), or (11,8), but if all of these first choice cells are occupied, then it will try the secondary choices, which are (9,6) or (9,8). But if agent1 is a nonbenevolent agent, then it will just simply choose a random cell from its neighbors (10,6), (10,8), (11,6), (11,8), (12,6), (12,7), or (12,8) and move to it to avoid box4.

Rule # 11 describes the behavior of a cooperative agent. If agent1 was cooperative, then it would report box3 to all other agents, since it is already carrying box2. Agent3 (9,13) is looking for a box, so it moves toward box3, using the path having the shortest distance.

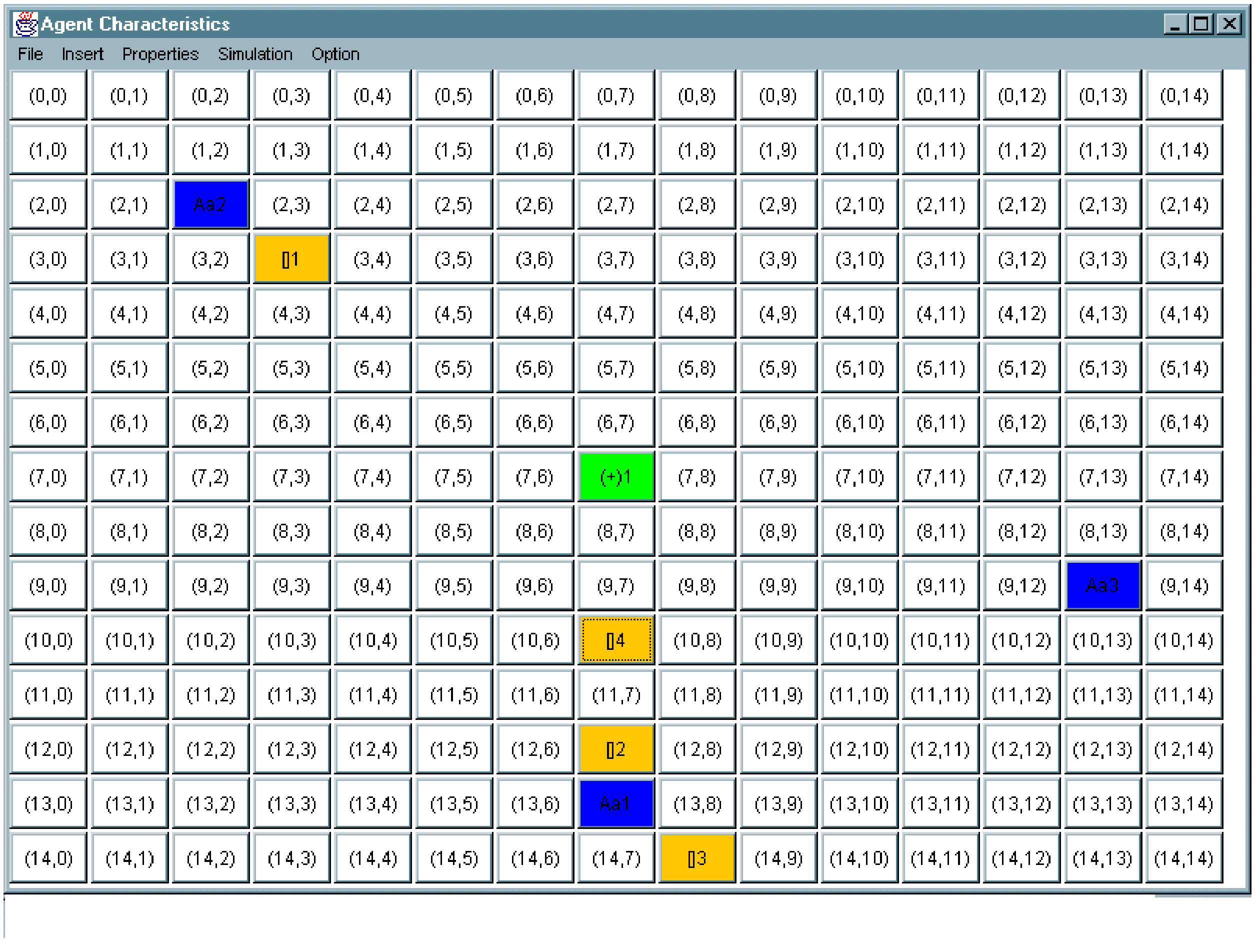

Figure 4. Scenario used to elaborate on some of the most important agent behavior rules

5. Simulation Results

Four scenarios were simulated to show the result of agents taking benevolent action of moving obstacle boxes out of their way while moving toward a target area. In all scenarios, agents were also cooperative by reporting to all of the other agents any box they find in the way. If an agent is moving toward a Box (not carrying a Box) or Target (carrying a Box) and encounters another agent in its path, then it moves to a random empty neighboring cell. But if the obstacle was a box, then the benevolent agent will move it out of the way, where a nonbenevolent agent would just move to a random empty neighboring cell.

In the first scenario, agents, boxes, and target area were inserted as shown in Figure 5. In this case, benevolent action is a must for agents, otherwise, no agent will be able to reach the target area. The following results were collected:

Operation	Nonbenevolence	Benevolence
Move	Infinite	308
Pick up box	7	45
Drop box	0	14
Deposit box	0	31

Scenario # 1 simulation results

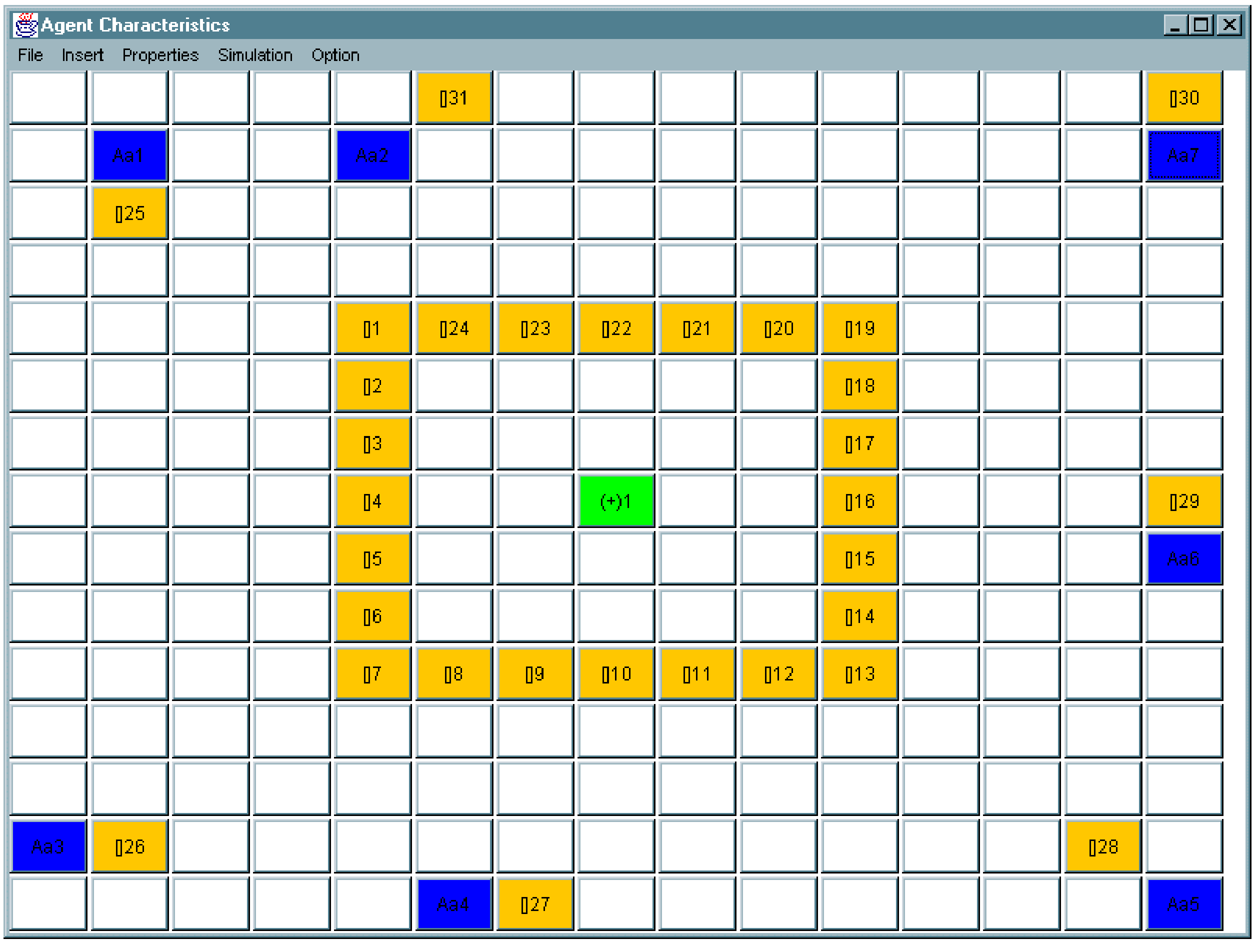

Figure 5. Simulation scenario # 1

In the second scenario, agents, boxes, and a target area were inserted as shown in Figure 6. In this case, benevolent action is not a must in order to achieve the main goal, but it will help the agents to reach the target area in fewer steps. Due to the fact that a nonbenevolent agent avoids any obstacle, box or agent randomly, and a benevolent one avoids other agents also randomly, three identical simulations were run. The following results were collected:

Operation	Nonbenevolence			Benevolence		
Simulation No	#1	#2	#3	#1	#2	#3
Move	558	628	544	409	472	423
Pick up box	31	31	31	45	43	45
Drop box	0	0	0	14	12	14
Deposit box	31	31	31	31	31	31

Scenario # 2 simulation results

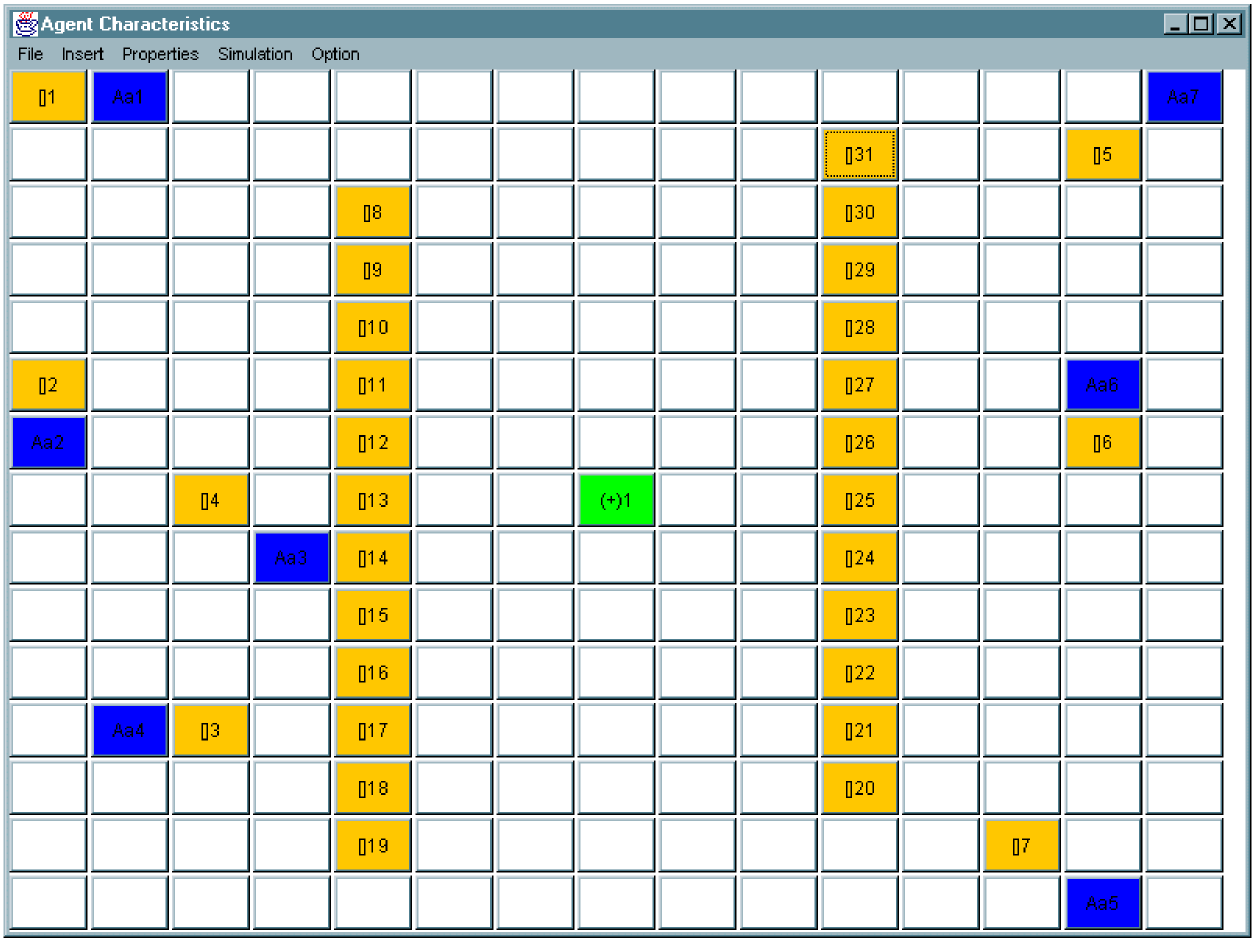

Figure 6. Simulation scenario # 2

In the third scenario, agents, boxes, and a target area were inserted as shown in Figure 7. In this case, there are 21 boxes and 3 agents, i.e., an agent for every 7 boxes. Three identical simulations were run and the following results were collected:

Operation	Nonbenevolence			Benevolence		
Simulation No	#1	#2	#3	#1	#2	#3
Move	243	224	189	258	208	268
Pick up box	21	21	21	27	27	27
Drop box	0	0	0	6	6	6
Deposit box	21	21	21	21	21	21

Scenario # 3 simulation results

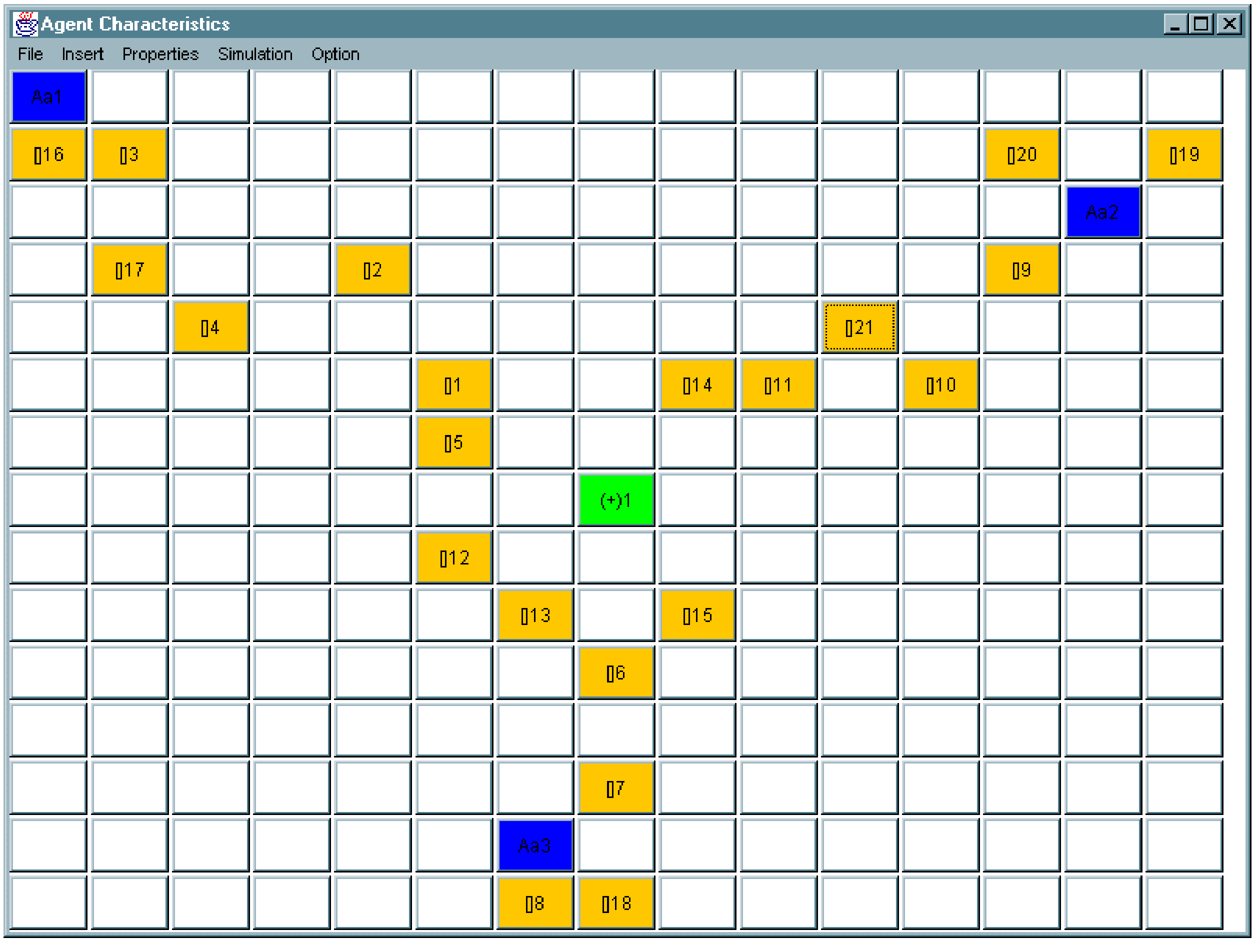

Figure 7. Simulation scenario # 3

In the fourth scenario, agents, boxes, and target area were inserted as shown in Figure 8. In this case, there are 21 agents and 3 boxes only, i.e., approximately 7 agents for every box. Three identical simulations were run and the following results were collected:

Operation	**Non Benevolence**			**Benevolence**		
Simulation No	#1	#2	#3	#1	#2	#3
Move	187	121	117	167	123	130
Pick up box	3	3	3	3	3	3
Drop box	0	0	0	0	0	0
Deposit box	21	21	21	21	21	21

Scenario # 4 simulation results

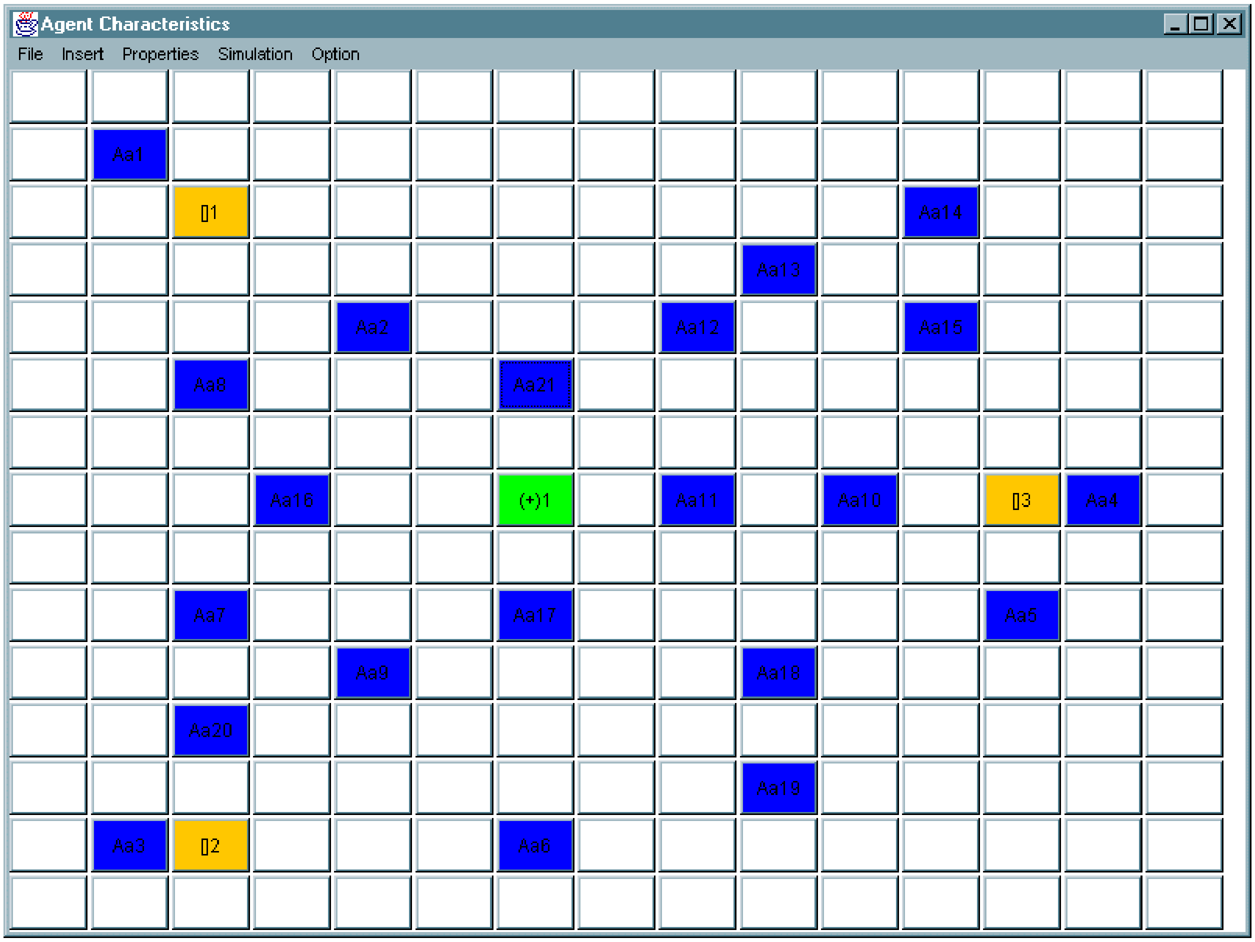

Figure 8. Simulation scenario # 4

6. Discussion of Results

The agent's individual goals did not include moving any obstacles out of the way, but those are optional actions an agent can take to help others and its society. The first scenario is an extreme one, but it demonstrates how benevolent action could be vital. This scenario contradicts the definition of benevolence provided earlier to the extent that the benevolent action is not an optional, but rather a required action in order for the system of agents to reach a final goal state. In addition, it shows that the reward of the benevolent action is immediate rather than a long term one. In short, this case illustrates the maximum benefits the agents and their society can obtain by adopting benevolent behavior.

Studying the results of the second simulation case, it is obvious that the total number of move operations taken by benevolent agents is less than that of nonbenevolent ones, which also leads to a smaller number of search operations. On the other hand, the number of pick up and drop box operations is higher for the benevolent agents, due to its actions in moving an obstacle. Comparing the average number of these operations, the benevolent agents saved about 142 move operations, whereas the nonbenevolent agents saved about 13 pick up and 13 drop box operations. From this comparison, we can conclude that the benevolent behavior indeed saved more operations for agents, which enabled them to reach their final goal state faster.

The third and forth simulation scenarios do not show large differences caused by adopting benevolent behavior. The difference between the number of agents and boxes is large, but that did not make a big difference when comparing the number of moves of the nonbenevolent and the benevolent agents. There are other many factors that should influence the dynamics of the system, such as the locations of the agents, boxes, and targets.

7. Conclusion

There has been a lot of research on cooperation among agents, but only a little on benevolence. However, benevolence might be considered cooperation between an agent and its society, in which case some of the research and results on cooperation could be apply to benevolence. An agent will cooperate with its society by taking a benevolent action to help other agents within its societal bounds, without looking for immediate reward from other members of its society. Such an action will help the benevolent agent only if other agents conduct similar behavior, which is why benevolence only makes sense within a multiagent system. The benefit of benevolence is directly related to both the location and the number of objects within an environment.

One of the main areas where benevolence will play a major rule is in Internet computing. Many agents will soon be populating the Web, and if they behave benevolently by helping each other search for information, for example, it will greatly reduce the traffic on the Internet, thereby benefiting everyone.

References

[Castelfranchi and Conte 1996] Cristiano Castelfranchi and Rosaria Conte, " Distributed Artificial Intelligence and Social Science: Critical Issues," *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, 1996.

[Cesta, Miceli, and Rizzo 1996] Amedeo Cesta, Maria Miceli, and Paola Rizzo, "Help Under Risky Conditions: Robustness of the Social Attitude and System Performance," *Proceedings ICMAS-96*, Kyoto, Japan, December 1996, pp. 18-25.

[Genesereth and Ketchpel 1994] Michael Genesereth and Stephen Ketchpel, "Software Agents," *Communications of the ACM*, Vol. 37, No. 7, 1994, pp. 48-53.

[Hales 1998] David Hales, "Selfish Memes & Selfless Agents – Altruism in the Swap Shop," *Proceedings Third International Conference on Multi-Agent Systems*, Paris, France, IEEE Computer Society Press, July 1998, pp. 431-432.

[Maes 1994] Pattie Maes, "Agents that Reduce Work and Information Overload," *Communications of the ACM*, Vol. 37, No. 7, 1994, pp. 31-40.

[Montgomery and Durfee 1990] Thomas A. Montgomery and Edmund H. Durfee, "Using MICE to Study Intelligent Dynamic Coordination," *Proc. IEEE Conference on Tools for Artificial Intelligence*, IEEE Computer Society Press, 1990.

[Rosenschein and Zlotkin 1994] Jeffrey S. Rosenschein and Gilad Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, MIT Press, Cambridge, MA, 1994.

[Russell and Norvig 1995] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[Sen 1996] Sandip Sen, "Reciprocity: a foundational principle for promoting cooperative behavior among self-interested agents," *Proceedings ICMAS-96*, Kyoto, Japan, December 1996, pp. 322-329.

Dictionary, http://www.onelook.com/

Abdulla Mohamed is currently a Ph.D. student in the Electrical and Computer Engineering Department at the University of South Carolina. His research interests include studying the characteristics of agents and multiagent systems. He received his BS and ME degrees in computer engineering in 1992 and 1997, respectively, from the University of South Carolina. He is a member of Tau Beta Pi, Eta Kappa Nu, Golden Key, and IEEE.

Michael Huhns is a professor of electrical and computer engineering and director of the Center for Information Technology at the University of South Carolina. Prior to this he was a senior researcher at the Microelectronics and Computer Technology Corporation (MCC) and an adjunct professor in computer sciences at the University of Texas, Austin. Dr. Huhns received the B.S.E.E. degree in 1969 from the University of Michigan, Ann Arbor, and the M.S. and Ph.D. degrees in electrical engineering in 1971 and 1975, respectively, from the University of Southern California, Los Angeles.
Dr. Huhns is a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, ACM, IEEE, and AAAI. He is the author of over 100 technical papers in machine intelligence and an editor of the books *Distributed Artificial Intelligence, Volumes I and II* and *Readings in Agents*. His research interests are in the areas of cooperative information systems, DAI and multiagent systems, enterprise integration, and heterogeneous distributed databases. Dr. Huhns is an associate editor for IEEE Intelligent Systems, the Journal of Autonomous Agents and Multi-Agent Systems, and IEEE Internet Computing. He is on the editorial boards of the International Journal of Cooperative Information Systems and the Journal of Intelligent Manufacturing. He is a founder and treasurer of the International Foundation for Multiagent Systems.