Synthetic Social Construction for Autonomous Characters

Eric Baumer and Bill Tomlinson

Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-2775
ebaumer@ics.uci.edu, wmt@ics.uci.edu

Abstract

Borrowing ideas from the notion of social construction of self, this paper puts forth the idea of synthetic social construction - multiagent systems in which agents socially construct each others' roles and behaviors via their interactions with one another. An example implementation of synthetic social construction is presented demonstrating one use of this method to facilitate social learning. Synthetic social construction represents a novel approach to adaptive behavior in multiagent systems informed by human behaviors.

Introduction

Multiagent systems have proven themselves a useful tool in accomplishing certain types of computational tasks, such as bidding in auctions, societal simulations, or control of multi-robot systems. Rather than having a single, unified computational entity with which to work, designers can now employ many individual autonomous computational entities. While not a panacea, these multiagent systems are more adept at solving certain types of problems than their single agent counterparts.

However, in order to take full advantage of these abilities, it may be beneficial to approach these systems both as collections of interacting individuals and holistically as societies. By borrowing ideas from sociology, anthropology, and philosophy, we can gain new insight into different methods and techniques that might be used when developing multiagent systems (or societies). This type of thought is already evident in other multiagent work (López y López, Luck and d'Inverno 2004) (Mao and Gratch 2004).

One such idea comes from social constructionism (Berger and Luckmann 1966): the social construction of self. This theory posits that we define ourselves in terms of interaction with others. When one is the recipient of another's actions, one changes one's self concept depending on what actions were taken; if I am a frequent recipient of complements and respect, I may begin to hold myself in higher regard. When another is the recipient of similar actions, one sees one's self as similar to that other; if someone else is also repeatedly complemented and respected, I could consider myself similar to him or her. When such similarities are in place, one takes one's cues for social action (or inaction) from those that one considers to be similar to one's self. If the other person who had been regularly complemented and respected became thankful and humble (or if the other person remained tacitly aloof), I might be inclined to have a similar reaction (or lack thereof).

Admittedly, this description is not a full and complete notion of social constructionist theory, nor can such a theory explain the entirety of social interaction. However, the social construction of self has useful implications for designers of multiagent systems, particularly in regard to agent interaction and learning.

Previously, most learning in multiagent systems has been done via forms of reinforcement learning, memory-based reasoning, or model learning (Enembreck and Barthès 2005). While these methods have been both formally and experientially proven suitable for systems with a single agent, they have not been shown to have similar efficacy in the learning problem for multiagent systems. Some of these techniques have been modified for use by multiagent systems, but generally with some sort of caveat – either assumptions must be made about the conditions of the environment, agents must be given some a priori knowledge, multiple methods must be combined to be effective, or, most commonly, formal proof of their validity and efficacy has not been shown.

This paper presents a method for generating adaptive behavior in multiagent systems, *synthetic social construction*, wherein an agent adapts its behavior based on both the actions other agents take towards it and the interactions other agents have with one another. This paper does not offer a formal proof of the technique's perfection or completeness. Rather, it presents one example implementation in a domain where synthetic social construction is particularly pertinent and demonstrates results obtained there. We close with a discussion of advantages, limitations, and future work, both of the particular implementation and the method in general.

Related Work

This work brings together ideas from several other projects and synthesizes concepts that span multiple disciplines.

Social Construction

Autonomous agents have already been used to simulate a number of different social phenomena, including normative reasoning (López y López, Luck and d'Inverno 2004), social judgment (Mao and Gratch 2004), and others (Stirling 2004) (Hales and Edmonds 2003). Here, we use agents to simulate social construction (Berger and Luckmann 1966), specifically, a limited notion of the social construction of self. Using the idea that agents in societies take their cues from other agents that they see as similar to themselves, we enable agents to learn which actions to take by observing which actions are taken by similar agents.

Learning

Learning, both in AI and autonomous agents, is a well-known, important, and difficult problem. Q-learning (Watkins and Dayan 1992) is a common, robust technique for single agent learning. Although it has been noted that Q-learning is not particularly well-suited for dynamic, multiagent environments (Tuyls, Verbeeck and Lenaerts 2003) (Chalkiadakis and Boutilier 2003), modified versions have been adapted to and incorporated in multiagent learning (Nunes and Oliveira 2004) (Tuyls, Verbeeck and Lenaerts 2003) (Weinberg and Rosenschein 2004).

Various other methods have been developed, besides reinforcement learning, including memory-based reasoning and model learning. However, as noted elsewhere (Enembreck and Barthès 2005), many of these techniques do not adapt readily or easily to multiagent systems.

Coalitions

Another societal phenomenon that has been incorporated into multiagent systems is the forming of subgroups, or coalitions, within a society (Sherory and Kraus 1995) (Griffiths and Luck 2003). Most of these coalitions are goal-based, that is, agents form and join coalitions to better accomplish their own goals. While the coalitions are formed, agents explicitly work together to accomplish a temporarily unified goal.

While synthetic social construction forms subgroups within societies of autonomous agents, such groups are not formed to facilitate the accomplishment of a specific task. Furthermore, the agents do not pool their collective skills to achieve a previously unattainable goal. Rather, the groups formed here are made for the purposes of learning. While learning could be seen as a goal, it is a different sort of goal than those typical of coalitions, in that it is not at some point accomplished and then checked off so the next goal can be addressed.

Characters

The specific implementation discussed in this paper builds on work done with autonomous characters by a number of different researchers and groups (Tomlinson et al. 2005) (Tomlinson et al. 2001). While this paper does not bear relevance specifically or only to autonomous characters, the system discussed below was implemented on a platform based on other characters research.

Synthetic Social Construction

This paper presents the idea of Synthetic Social Construction, a method with which agents learn action selection by observing the choices made by other agents. Part of the notion of the social construction of self says that one takes cues on how to act from others to which one sees oneself as similar. When such a similar person takes or does not take an action, one is more or less inclined to take that action as well, respectively.

Similarity

In order for an agent to determine from whom to learn, it may be beneficial that the agent have a means of measuring its similarity to other agents. We provide a similarity metric that differs from Euclidean distance or other such metrics. We do not use these other metrics because they give the same distance regardless of the magnitude of the values. For example, Euclidean distance says that the point (10, 10) is just as far away from the point (20, 20) as the point (110, 110) is from (120, 120). However, we wanted to be able to take range into account. That is, while the exact distance between these two points is the same, the proportions between the actual values are quite different. To capture this idea, we use a similarity metric that bases similarity not only on the difference between values but also on their difference with respect to their magnitudes.

Let us consider two agents, x and y, with various properties p_1, … p_n, which, for measuring similarity, we call dimensions. We use the notation $x.p_i$ to refer to x's i^{th} property p_i. These properties can be of two kinds: discrete or continuous. We now define a similarity function s(j, k) to compare any two agents in a single dimension.

For discrete properties, either the agents have the same value or they do not. For example, a person could be a homeowner or not be a homeowner, but there is not a continuum of partial homeownership such that one person could own more of a home than another person. For such properties, if $x.p_i = y.p_i$, then x and y are similar in that dimension, and $s(x.p_i, y.p_i) = 1$. Otherwise, they are different, and $s(x.p_i, y.p_i) = 0$. This maintains the identity property for metrics.

Continuous properties are values in some range and can be more or less similar depending on their position in that range. For example, a person who is 1.75m tall would be more similar to another person who is 1.8m tall than one who is 2m tall. For such properties, the similarity in that dimension is given by the formula:

$$s(x.p_i, y.p_i) = 1 - \left| \frac{(x.p_i + y.p_i)/2 - x.p_i}{(x.p_i + y.p_i)/2} \right|$$

where $s(x.p_i, y.p_i)$ is the similarity x bears to y in dimension p_i. Note that $s(x.p_i, y.p_i) = s(y.p_i, x.p_i)$, which is significant for two reasons. One, when comparing x and y, we only have to do one calculation, rather than needing one calculation for x's similarity to y and a separate calculation for y's similarity to x. Granted, the calculation involved here is not incredibly demanding and in all reality will most likely not give a huge performance boost. However, in real-time systems, one common application area for autonomous agents, often times every little performance boost makes a difference, especially when dealing with very large numbers of agents. More importantly, though, this equality maintains the symmetry property for metrics.

We also trap the similarity for any given dimension in the range 0.0 – 1.0. s(j, k) has a maximum value of 1 (when j = k), so the only real trapping is done if s(j, k) comes out negative. If agents x and y are vastly dissimilar in some dimension, we do not want that dissimilarity to detract from similarities in other dimensions. So, if s(j, k) comes out negative, rather than keep the negative value we simply replace it with 0. This trapping means that we may not maintain the triangle inequality for metrics. However, as discussed elsewhere (Santini and Jain 1995) (Finnie and Sun 2002), the triangle inequality does not and should not always apply to similarity metrics.

Note that there are two situations in which the average, and thus the denominator, may be 0, causing the value to be undefined. First, if both $x.p_i$ and $y.p_i$ are 0, then the average will obviously be 0. Here, we introduce a special case to return a similarity of 1, since the two have the exact same value. Second, if $x.p_i = -1 * y.p_i$, then the average here, too, will be 0. Since we are using a similarity metric that bases similarity partially on magnitude and in this case x and y have exactly the opposite magnitude in the dimension in question, we say that they have absolutely no similarity in that dimension and return a similarity of 0.

Now we define a similarity function S(x, y) to compare two agents across all dimensions as given by the formula:

$$S(x, y) = \frac{\sum_i s(x.p_i, y.p_i)}{d}$$

where S(x, y) is the overall similarity of x and y across all dimensions.

Simple Learning

This method allows one agent to learn from other agents based on those agents' decisions. However, the initial learning done by those must be based on some sort of reasonable course of action. Otherwise, an agent will base its actions on the arbitrary decisions of others rather than on decisions informed by those others' experiences. This, this method can only be used in conjunction with some other learning algorithm.

The most intuitively applicable methods are reinforcement learning and memory-based reasoning. In reinforcement learning, the agent receives some sort of feedback after a given decision that makes the agent more or less inclined to make the same decision again. When that time comes, another agent can simply observe the first agent's decision and then adapt its behavior accordingly.

The case of memory-based reasoning is somewhat similar. When an individual agent makes a decision based on its memories, it can update those memories with the result of the present decision and store that information for future decision making. Likewise, when one agent watches another agent make a decision, the first agent can update its memories almost as if it had made the same decision.

Model learning, however, does not apply as easily. It is difficult for one agent to determine if another agent's model is similar to its own and if even makes sense to update its model based on the other agent's actions. That is not to say that model learning could in no way be used here, but rather that other learning methods are more amenable.

Once one agent has learned from its actions and adjusts its decisions accordingly, it becomes possible for other agents to observe these decisions and adapt their behavior to reflect the new decisions.

Social Learning

Based on the similarity metric described above, each agent keeps a record of how similar it is to another agent. When one agent observes another agent take or not take an action about which it cares, the agent in question then modifies its likelihood of taking a similar action based on the degree of similarity to that other agent.

Thus for an agent x, its tendency to take a certain action at some future time, T_{n+1}, after observing some action of agent y is given by the formula:

$$T_{n+1} = \frac{T_n - d_y}{2} * S(x, y)$$

where d_y is the decision of the agent y (1 is performing the action and 0 is not), T_n is agent x's current tendency to take the given action, and S(x, y) is x's similarity to y, as defined above.

There are two important aspects of this approach to note. One is that agent x's tendency approaches agent y's action asymptotically; x's modifies it's tendency by half the difference between its tendency and y's action. Second is that the tendency adjustment is multiplied by the agents' similarity, which is on a scale of 0.0 to 1.0. The greater the similarity between two agents, the greater the influence one will have on the other.

It is important to remember that this method of learning does not attempt to converge over time, or even attempt to reach an optimal value. Rather, the purpose is an attempt at emulating some of the ways that humans perform social construction. Thus, the paper contains no proof or mathematical demonstration of this method's optimality.

Implementation

Currently, work is being done on implementing synthetic social construction in the Virtual Raft Project (Tomlinson et al. 2005). The following section presents the portions of that implementation that have been completed thus far.

The Virtual Raft Project

The Virtual Raft Project centers on a mobile computing interaction paradigm in which stationary computers are islands of virtual space and mobile devices are rafts by which autonomous systems may cross the sea of real space that separates virtual islands. To emphasize this metaphor, the mainstay of the installation consists of three collocated desktop machines on each of which a group of virtual characters live. Tablet PC's act as virtual rafts, which users can physically bring up to one of the islands to allow a character to jump off of the island and onto the raft. If the raft is then brought to another island, the character will jump off onto that island. For a better sense of the system and the interactions involved, please see the video at http://tinyurl.com/5yxkn.

To Jump or Not to Jump

The main decision these characters are faced with is whether or not they will be jumping onto or off of the raft. Initially, the characters would jump whenever a raft or island was present. The tablet PC's that serve as virtual rafts incorporate accelerometers, which allow the attitude of the tablet to affect the movement of the raft on the tablet's screen, such that tilting the tablet in one direction causes the raft to move in that same direction. If the raft is tilted too far, the character falls in the water and its fire goes out. When the raft moves around, the character attempts to balance on the raft, all the while making a record of how long it was on the raft and how rough the ride was, the later codified by the amount of sliding around the raft does and labeled as the distance the raft traveled. We wanted the characters to form impressions about whether traveling on the raft is a good or bad thing, depending on how their trip went. This could be accomplished with a simple reinforcement learning system. However, once the characters arrived on a new island, we wanted them to pass those impressions on to other characters. In this way, different characters could share their different ideas about the raft and collectively get much more diverse sample than would be possible individually.

When a character is first created, we start it off with a 100% likelihood of jumping onto the raft. This is partly because if the characters did not start off jumping then the installation would not be incredibly engaging, and partly because if the characters did not jump to begin with people would likely assume the behavior was a malfunction rather than a designed behavior.

Once a character has ridden the raft and then disembarks, it adjusts its likelihood to board the raft again based on the quality of the trip. This implementation uses a simple reinforcement learning strategy based on the amount of time spent on the raft, how far the raft traveled, and whether the character fell in the water. After a trip on the raft, the agent's new jump tendency is given by the formula:

$$J_{n+1} = J_n + \left(0.25 + \frac{d}{20}0.25\right) + \left(0.25 + \frac{t}{10}0.25\right) + \left(0.1 - 0.1 * f\right)$$

where J_{n+1} is the new jump tendency, J_n is the previous jump tendency, d is the distance traveled on the raft, t is the time spent on the raft, and f is 1 if the character fell off the raft and 0 if not.

While on the island, the characters observe one another, and each determines its similarity with respect to the others using the similarity metric described above. The current implementation uses five dimensions: the island on which the characters were born, the amount of time they spent on the raft during their last trip, the distance the traveled during that trip, whether they have a fire on their hand, and what color fire they are carrying. Birth island is signified by the color of a character's crown and is a discrete value; we do not say that certain islands are closer to others. While sitting around the fire, the characters take turns "telling stories" to one another, which is signified visually by one of the characters standing up, gesturing as if speaking, and optionally (depending on the installation) verbally and audibly describing their trip to the other characters. Thus, it makes sense that characters would be able to compare themselves to each other based on trip duration and quality, both of which are treated as continuous values for the purposes of the similarity metric. Whether or not a character's fire is lit is obvious and can intuitively be factored into similarity as a discrete value. What color fire the character is carrying can only be known if it has a fire and thus is only factored into the similarity metric if both of the characters in question have their respective fires lit. In this case, fire color is treated as a continuous value, since during the course of the installation the

characters can mix fire colors. Fire color is stored as an RGB value, so the overall similarity in this dimension is the composite of the similarities of the red, green, and blue components of each character's fire color.

As a visual signification of similarity, the characters tend to "hang out" near other characters to which they are similar. Characters are most drawn to others to which they are most similar and are most repelled from those to which they are most dissimilar. This makes it apparent to the user that various groups are being formed within the population on an island.

The learning is actually done when a user brings a raft up to an island. When the raft approaches, all the characters on the island turn around and take notice of the raft's presence. Each character then decides for itself whether it is interested in jumping on the raft or not, based on a pseudorandom number compared to its jumping tendency. If it is in fact interested, the character will visually signify this interest by approaching the foreground of the screen. If it decides it would rather not jump, the character turns back around and sits down again at the fire. When this decision occurs, each character observes the other characters' decisions and modifies their own jump tendency using the social learning formula described above.

Discussion

Here, we list some of the benefits of synthetic social construction, address some of its limitations, mention some possible future directions, and list a few questions that could be addressed in this workshop. Throughout, we mention aspects that were particular to this implementation.

Benefits

Collective Learning. As mentioned at the beginning of this paper, the learning problem for multiagent systems is an important and difficult one. Synthetic social construction offers a novel approach to multiagent learning that takes advantage of different agents having different experiences. Rather than restricting an agent to learning from its own experience, this method allows the agent to learn behavior from others' experiences by watching their decisions. An agent cannot watch another agent make a decision, observe the outcome, and decide if it would make the same decision. Rather, the first agent lets the other agent determine whether the course of action was a beneficial one, observes the other agent modify its decision patterns, and then adapts its own decisions based on those modifications.

Character Believability. When the Virtual Raft Project began, enhancing character believability was a core concept. By allowing characters to break the screen, the characters are no longer seen as entities that live only within a single device but as entities that maintain permanence across several stationary and mobile devices. When a character maintains this permanence, it becomes less like a part of a computer program bound to one machine and more like a truly autonomous character.

Social Simulation. Simulating artificial societies has become an important area of research, as evidenced by, among other things, an entire journal devoted to the subject, the Journal of Artificial Societies and Social Simulation. If such pursuits are truly intended to imitate human societies, it might be beneficial to incorporate mechanism similar to those theorized to be in place in humans. Furthermore, behaviors that emerge from such synthetic societies may be able to help us understand the human behavior after which it was modeled and help inform new lines of questioning and new areas of research.

Limitations

Not Optimal. This learning method does not seek any optimal solution or equilibrium (at least, not explicitly). This is both a limitation and an advantage. For some applications, agents should be finding the best solution possible. However, in other applications, agents may not be intended to find an optimal course of action, there may not be a clear definition of optimality, or such optimality may not exist. In these situations, learning from other agents may become a distinctive advantage. Also, this method is well suited for applications where agents are being used to model social interactions rather than to accomplish some task.

Homogeneity. For the similarity metrics to work the agents have to be homogeneous, otherwise they won't know how to compare to one another. If the agents are not homogenous, it does not rule out the use of some similarity metric, but it does make the development of such a metric more difficult.

Scalability. As the number of agents increases, scalability quickly becomes a pressing issue. If the number of agents is n, it takes $O(n^2)$ time to compare every agent to every other agent.

Implementation described here includes an optimization that removes the need to run an $O(n^2)$ process to constantly update character similarities. Rather than comparing every character to every other character at every time step, the characters have a simple tracking that stores a previous value for each of the dimensions of interest. At every time step, the current value is compared against the previous value. If anything has changed, all the similarities for that character are recalculated, which takes O(n) time, where n is the number of agents. This is far better than an $O(n^2)$ process every time step, but it does require O(n * m) extra memory, where m is the number of dimensions being used for similarity testing. This is a useful optimization, but ultimately it may prove as a weakness in this method since not every implementation will have so few dimensions or be able to take advantage of dimensions that are not constantly in flux.

Simple Learning. Despite the ability to learn from other agents, this method still requires that some agent learn for itself first, either by reinforcement learning as in the implementation described here or by some other method. Once one agent has learned something, it can demonstrate that knowledge to other agents, but it must learn experientially on its own to begin with.

Future Work

Jump Off. Currently, the characters are only concerned with whether or not to jump on the raft; when presented with an island onto which to jump, they will always jump off the raft. This could be improved by allowing the agents to gather simple data about an island, such as the average similarity of the characters or some information about the norms on that island (López y López, Luck and d'Inverno 2004), and then making a decision as to whether or not that was an island onto which they wanted to jump.

Vicarious Learning. Rather than one agent relying on another agent to learn something and then having the first agent just watch what the second agent does, the first agent should be able to watch the second agent's actions and determine for itself whether the results are desirable. This sort of might get back to something like (Tuyls, Verbeeck and Lenaerts 2003), where they use a type of Q-learning modified for agents.

Trust. After one agent has observed another and learned something from it, the agent could analyze the benefit of learning from that other agents and possibly establish a degree of trust between them that would augment or mitigate further learning.

Permanence and Habituation. Once an agent has learned by watching another agent do the same task a hundred times, that learning should be incredibly ingrained so that if the other agent suddenly does something different, the first does not necessarily modify its behavior immediately to follow suit. Alternatively, once one agent has watched another agent do the same task a hundred times, the agent may have become habituated to that particular stimulus and so it no longer has any effect on the agent.

Questions

There are a number of questions this work has raised that could possibly be addressed in this workshop.

Currently, the system still relies on a very simple reinforcement learning method. Would there be a way to make the learning entirely social? One possibility in this implementation would be to assign all the characters random initial jump tendencies and allow whatever order happens to emerge. However, there would still be no link between the raft trip and the characters' decisions. Does it even make sense to try and remove reinforcement or some other simple learning?

A large part of social construction deals with how individuals act toward each other, but currently the characters in the Virtual Raft Project do not take any actions directly toward one another. What sorts of actions and interactions might make sense? Should they trade? Should they fight (one participant at CHI moved all the characters to one island and then wanted a Battle Royale to ensue)? Should they have tea? Should they just chat?

Is there a way to overcome the scalability problems of comparing every agent with every other one? There are classical logic puzzles about passing information among parties, but in this case, the information each agent has is calculated and only pertinent to that agent. Could there be some way to work around this?

What else could we do with synthetic social construction, other than learning? Could we use it for coalition forming? Can we use it for construction of concepts other than self, such as emotion, social roles, or possibly, rather than mimicking other agents, allow agents to mimic humans with which they interact?

Conclusion

This paper has presented synthetic social construction, a method for an agent in a multiagent system to adapt its behavior based on the behavior of other agents. The specifics of one initial implementation in the Virtual Raft Project were described, along with some benefits and limitations of the approach. Admittedly, having autonomous characters decide whether on not to jump on or off a virtual raft seems relatively simple in the scope of possible social behavior. However, applying the basic concepts from this implementation to other systems may have the potential to create complex, compelling, and useful behavior. Synthetic social construction offers a novel approach to multiagent learning, social simulation, and decision-making in multiagent systems.

References

Berger, P. L. and Luckmann, T. (1966). *The Social Construction of Reality: A Treatise on the Sociology of Knowledge*. Garden City, NY, Anchor Books.

Chalkiadakis, G. and Boutilier, C. (2003). Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. *Autonomous Agents and Multiagent Systems*, Melbourne, Australia, ACM Press.

Enembreck, F. and Barthès, J.-P. (2005). ELA -- A New Approach for Learning Agents. *Autonomous Agents and Multi-Agent Systems* 10: 215-248.

Finnie, G. R. and Sun, Z. (2002). Similarity and Metrics in Case-Based Reasoning. *International Journal of Intelligent Systems* 17(3): 273-287.

Griffiths, N. and Luck, M. (2003). Coalition Formation through Motivation and Trust. *International Conference*

on Autonomous Agents, Melbourne, Australia, ACM Press.

Hales, D. and Edmonds, B. (2003). Evolving social rationality for MAS using "tags". *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, ACM Press**:** 497--503.

López y López, F., Luck, M. and d'Inverno, M. (2004). Normative Agent Reasoning in Dynamic Societies. *International Conference on Autonomous Agents*, New York, NY, IEEE Computer Society.

Mao, W. and Gratch, J. (2004). Social Judgment in Multiagent Interactions. *Autonomous Agents and Multiagent Systems*, New York, NY, ACM Press.

Nunes, L. and Oliveira, E. (2004). Learning from Multiple Sources. *Autonomous Agents and Multiagent Systems*, New York, NY, ACM Press.

Santini, S. and Jain, R. (1995). Similarity Matching. *Second Asian Conference on Computer Vision*, Singapore.

Sherory, O. and Kraus, S. (1995). Coalition formation among autonomous agents: Strategios and complexity. *From Reaction to Cognition*. C. Castelfranchi and J.-P. Müller. Heidelberg, Springer-Verlag**:** 57-72.

Stirling, W. C. (2004). A Sociological Framework for Multiagent Systems. *Autonomous Agents and Multiagent Systems*, New York, NY, ACM Press.

Tomlinson, B., Downie, M., Berlin, M., Gray, J., Wong, A., Burke, R., Isla, D., Ivanov, Y., Johnson, M. P., Lyons, D., Cochran, J., Yong, B., Stiehl, D., Soetjipto, R., Zaharopol, D. and Blumberg, B. (2001). AlphaWolf. *Proceedings of SIGGRAPH 2001: conference abstracts and applications*.

Tomlinson, B., Yau, M. L., O'Connell, J., Williams, K. and Yamaoka, S. (2005). The Virtual Raft Project: A Mobile Interface for Interacting with Communities of Autonomous Characters. *Conference Abstracts and Applications, ACM Conference on Human Factors in Computing Systems (CHI 2005)*, Portland, OR.

Tuyls, K., Verbeeck, K. and Lenaerts, T. (2003). A Selection-Mutation Model for Q-learning in Multi-Agent Systems. *Autonomous Agents and Multiagent Systems*, Melbourne, Australia, ACM Press.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical Note: Q-Learning. *Machine Learning* 8(3-4): 279-292.

Weinberg, M. and Rosenschein, J. S. (2004). Best-Response Multiagent Learning in Non-Stationary Environments. *Autonomous Agents and Multiagent Systems*, New York, NY, ACM Press.