

Combining Idleness and Distance to Design Heuristic Agents for the Patrolling Task

ALESSANDRO DE LUNA ALMEIDA¹
PEDRO MACHADO MANHÃES DE CASTRO¹
TALITA RODRIGUES DE MENEZES¹
GEBER LISBOA RAMALHO¹

¹Universidade Federal de Pernambuco – Centro de Informática
{all,pmmc,trl,glr}@cin.ufpe.br

Abstract

A group of agents can be used to perform patrolling tasks in a variety of domains, especially in computer games. In a previous and pioneer work, we presented an initial discussion of multi-agent patrolling task issues. In this paper we deepen this study, including new architectures which use advanced heuristic techniques in order to achieve better performance.

Keywords: *Computer Games, Multi-agent systems, coordination, patrolling*

1 Introduction

To patrol is literally “the act of walking or travelling around an area, at regular intervals, in order to protect or supervise it”¹. This task is by nature a multi-agent task and there are a wide variety of problems that may reformulate as particular patrol task. As a concrete example, during the development of the Artificial Intelligence component of an interactive computer war game, we faced the problem of coordinating a group of units to patrol a given rough terrain in order to detect the presence of “enemies”. The quality of the agent architecture used for patrolling may be evaluated using different measures. Informally, a good strategy is one that minimizes the time lag between two passages to the same place and for all places.

Beyond our specific game, performing multi-agent patrolling efficiently can be useful for various game categories where distributed surveillance, inspection or control is required. This is typically the case of strategy games (such as StarCraft, Civilization III) and RPG (such as Diablo and Ultima Online) for detecting mobile characters and new enemy buildings, protect cities and resources, etc. Multi-agent patrolling can also be useful in combat games

(such as Panzer Command and StarWars Rogue Squadron), in especial in first-person shooters (such as Unreal Tournament and CounterStrike), for the detection of enemies and the discovery of items. Arcade games (such as PacMan) and simulators can directly use multi-agent patrolling algorithms, too. Even collective sport games (such as FIFA Soccer and NBA basketball) can perhaps take profit of some kind of multi-agent patrolling for positioning characters.

Despite its relevance, the patrolling task has not been seriously studied yet. On one hand, the literature has sound works concerning related studies, such as network mapping², the El Farol³, steering behaviors⁴⁻⁶. However, these studies’ characteristics are quite different from the patrolling task, which requires specific solutions. On the other hand, the works devoted precisely to patrolling tasks⁷⁻⁹ do not present a systematic evaluation of the possible coordination strategies, agent models, agent society organizations, communication constraints, and so on. The only work¹³ which considers the efficiency of the patrolling task is very incipient and uses very simple experimental scenarios (non-weighted graphs with six and nine nodes).

In a previous and pioneer work, we presented an initial discussion of multi-agent patrolling task issues, as well as an empirical evaluation of possible solutions¹⁰. In order to accomplish this study, we have proposed some architectures of multi-agent systems, evaluation criteria and experimental scenarios. In this paper we deepen this study, including new architectures using advanced heuristic decision making and pathfinding techniques. The results show the adequacy of each agent architecture according to the characteristics of a given patrol task.

The remainder of this paper is organized as follows. Next section defines precisely what we mean by the patrolling tasks. Section 3 shows the main steps we have followed in our previous work. Section 4 presents the new architectures developed. Section 5 shows the results and the discussion about them. Section 6 draws some conclusions and indicates directions for future work.

2 The Patrolling Task

We consider the patrolling task on a graph which represents an abstraction of the environment to be patrolled. The advantage of adopting such an abstract representation is that the patrolling solutions proposed to it can be applied to different kind of problems, from terrain motion to web navigation. Given a graph, the patrolling task refers to continuously visiting all the graph nodes so as to minimize the time lag between two visits.

There are some variations in the graph to be patrolled. In some situations, such as a terrain containing mobile obstacles, the graph edges may change. In other situations, priorities may be set to some regions covered by sub-graphs. The edges may have different associated lengths (weights) corresponding to the real distance between the nodes.

In our previous work¹⁰, we have reduced the patrolling task to graphs with the following characteristics: non-weighted (the distance between two connected nodes is one), static edges (the set of edges does not change with time), uniform patrolling (the same priority for all nodes). In the present work, we have

considered the real distance between the nodes, representing them as the edge weights. We also propose new architectures to take advantage of the real distance between the nodes in order to achieve better performance. In fact, the “unitary length” constraint is an oversimplification of the terrain representation, especially in application such as games, where the distance or cost between adjacent nodes is not constant.

3 Previous Work

3.1. Evaluation Criterion

One of the contributions of previous work¹⁰ lies in the choice of evaluation criteria for comparing different MAS architectures, since no related work has used performance measures adapted to the patrolling task. The most important evaluation criterion is defined next.

Considering that a cycle is the time unit used in the simulations, the instantaneous node idleness is the number of cycles that a node has remained unvisited. The instantaneous graph idleness is the average instantaneous idleness of all nodes in a given cycle. Finally, the graph average idleness, or simply average idleness, is the average instantaneous graph idleness over an n-cycle simulation.

3.2. Multi-Agent Systems Investigated

In order to define which MAS architectures would be interesting to be evaluated in patrolling, we have explored four basic parameters (basic type, communication, node choice, coordination). We have considered only homogeneous architectures where the agents are all the same. The exceptions are the coordinated architectures, which contain a coordinator and various patrollers.

Straightforwardly, the first parameter we have considered is the classical difference between reactive and cognitive agents: whereas reactive agents simply act based on their current perception, cognitive ones may pursue a goal. This implies that the field of vision of reactive agents is one-node depth, i.e., a reactive agent only perceives the adjacent nodes and chooses one of them to visit. Cognitive Agents can choose any node of the whole graph as a goal-

node and use path-finding techniques (Floyd-Warshall Algorithm in our case) to reach that node.

We also considered the communication among the agents. Taking into account the real-world situations agents may face while patrolling, there are roughly three ways the agents can communicate with each other: via flags, via blackboard, and via messages. In the first case, agents leave flags in the environment^{11,12} which are recognized by themselves or by the other agents. In the second case, the information about the environment is stored in a common shared base. In the last case, agents can communicate with the others directly by exchanging messages. In this work, the agents can only exchange messages with the coordinator, when it exists. Enabling communication among all agents requires complex architectures with negotiation mechanisms for conflict solving.

Another key aspect in multi-agent movement coordination is to use a central coordinator, which chooses the goal-node of each (coordinated) agent, or a decentralized one, where coordination emerges from agent interaction.

Next node choice is also an important point in generating possible solutions. In our previous work, the choice criteria could be random or based on node idleness, where an agent chooses the node with the highest idleness.

Of course, two aspects directly influence this decision making. The first is the agent field of vision, which can be local or global as discussed earlier. The second is the fact that, according the communication possibilities, a given agent can or cannot know what the other agents have been doing. Consequently, the node idleness is individual, when the agent considers only its own visits; shared, when the agent takes into account the movement of all agents; or coordinated shared, when it makes use of a coordinator. Considering these variations of the decision making process, the agents will perform the next node choice according to the strategies shown in Table 1.

Architecture Name	Detailed Next Node Choice
<i>Random Reactive</i>	locally random

<i>Conscientious Reactive</i>	locally individual idleness
<i>Reactive with Flags</i>	locally shared idleness
<i>Conscientious Cognitive</i>	globally individual idleness
<i>Cognitive Blackboard</i>	globally shared idleness
<i>Random Coordinated</i>	globally coordinated shared
<i>Cognitive Coordinated</i>	globally coordinated shared

Table 1 - Detailed decision making strategy used by the old agent architectures, considering the field of vision and communication.

It is worth noting that choosing nodes according to the individual idleness is equivalent to follow a gradient, as this technique is used in multi-agent systems^{5,12}. In our case, nodes with high idleness act as valleys (attractors) and those with low idleness act as mountains (repulsors).

Another interesting parameter, which was not considered in ours works, is the ability of learning. Learning machine techniques can be used to mechanize the coordination among agents. These techniques are very effective especially in dynamic and complex environments¹⁷.

4 New architectures

In this paper we enhance the architectures previously developed by using advanced heuristic pathfinding and decision making techniques

A simple cognitive agent reaches the node using the shortest path. However, the final idleness would be lower if the agent had taken a few longer paths which passed through idler nodes instead. In other words, instead of searching shortest paths we also take into account the instantaneous idleness of the nodes in-between the current location and the goal. That is the idea behind the Pathfinder Agents, one of the contributions of this work. These agents transform the weights of the edges (v,w) of the graph so that they consider not only the distance between v and w, but also the idleness of the node w. They give an idlenessRate (ir, between 0 and 1) of importance to the idleness and (1-ir) to the distance as shown in Equation 1:

$$edgeWeight = (ir \times idleness) + ((1 - ir) \times distance)$$

Equation 1 - Heuristic evaluation of the edges

However, distance and idleness cannot be summed since they use different units of measurement. It is necessary to normalize their values in a common scale. We have done so by transforming into a scale between 0 and 1.

We attribute to the maximum idleness a zero normalized value (we want to traverse edges with high idleness values) whereas to the minimum idleness is attributed a value equals to one. Intermediary values are calculated by means of proportions as shown in Equation 2.

$$normIdleness = \frac{(idleness - maxIdleness)}{(minIdleness - maxIdleness)}$$

Equation 2 - Idleness Normalization

We obtain a similar equation to the distance. The difference is that we attribute to the minimum distance a zero normalized value (we want to traverse edges with short distances) whereas to the maximum distance is attributed a value equals to one. Intermediary values are calculated by means of proportions as shown in Equation 3.

$$normDistance = \frac{(distance - minDistance)}{(maxDistance - minDistance)}$$

Equation 3 - Distance Normalization

Once the graph is transformed, the Pathfinders find the shortest path to the goal node using the A* algorithm and considering the new edges weights^{15,1}.

Apart from developing a new pathfinding technique, we have also enhanced the decision making process. In our previous work, the next-node choice criteria could be random or based on node idleness, where an agent chooses the node with the highest idleness. However, if the node with the highest idleness is too far from the agent, it would be better to choose a near node

with almost maximum idleness. This is the idea behind the Heuristic Agents, another contribution of this work.

These agents consider not only the idleness of a candidate goal node, but also the distance between the agent and this candidate node. They give an idlenessRate (ir, between 0 and 1) of importance to the idleness and (1-ir) to the distance as shown in Equation 4:

$$nodeValue = (ir \times idleness) + ((1 - ir) \times distance)$$

Equation 4 - Heuristic Decision Making

It is also necessary to normalize the idleness and distance values, but we will not show how to do it here since it is very similar to the process applied in the normalization of edges values.

Considering these new variations of the decision making process and the pathfinding techniques, the new agents will perform the next node choice and pathfinding according to the strategies shown in Table 2.

Architecture Name	Detailed Next Node Choice
<i>Heuristic Conscientious Reactive</i>	locally individual idleness + distance
<i>Heuristic Reactive with Flags</i>	locally shared idleness + distance
<i>Heuristic Conscientious Cognitive</i>	globally individual idleness + distance
<i>Pathfinder Conscientious Cognitive</i>	globally individual idleness
<i>Heuristic Pathfinder Conscientious Cognitive</i>	globally individual idleness + distance
<i>Heuristic Cognitive Blackboard</i>	globally shared idleness + distance
<i>Pathfinder Cognitive Blackboard</i>	globally shared idleness
<i>Heuristic Pathfinder Cognitive Blackboard</i>	globally shared idleness + distance
<i>Heuristic Cognitive Coordinated</i>	globally coordinated shared idleness + distance
<i>Pathfinder Cognitive Coordinated</i>	globally coordinated shared idleness

<i>Heuristic Pathfinder</i> <i>Cognitive Coordinated</i>	globally coordinated shared idleness + distance
---	--

Table 2 - Detailed decision making strategy used by the new agent architectures, considering the field of vision and communication.

5 Experimental Results and Discussion

5.1 Experiment scenarios

We created a scenario (Figure 1) that takes many combinations of topology into account. The quantity of node and edges is below 100. There is not a clear partition of the graph into regions, since it is quite connected. There are bottlenecks. There is a relatively huge standard deviation of the weight of edges. We have also considered the same graph without weights on the edges in order to check if the performance achieved when we combine idleness and distance is also better in non-weighted graphs. Consequently, the experiments were done considering a great variety of topology and situations.

Instead of changing the number of nodes, we have equivalently changed the number of agents. We have used populations of 5, 10, 15, and 20 agents in order to keep adequate ratios between the number of nodes (62) and the number of agents.



Figure 1 - The map. Black blocks in the maps represent obstacles. The graph of possible paths is shown. The graph has 87 edges and 62 nodes. This figure is also a snapshot of the simulator.

5.2 Experiments Performed

All the experiments were carried out using a previously developed simulator. It had to be modified in order to support graphs with weights on the edges and the new architectures were implemented. All the eighteen groups of architectures previously mentioned were simulated. Especially for the heuristic and pathfinders agents we chose the following values for the idleness rates: 0, 0.2, 0.4, 0.6, 0.8, 1.0. Using these values, we achieve a total of 122 architectures tested.

For each of the 122 architectures studied, we have run 80 simulations (2 x 4 x 10), corresponding to the map with unitary weights and with non-unitary weights, four different populations of agents (5, 10, 15 and 20) and ten initial positions (the same initial positions for each simulation).

Considering the problem of choosing the appropriate number of cycles for a simulation, a reasonable approach is to visit each node k times. In the worst case, a node will stay WI iterations without being visited, where WI is the worst idleness of the simulation. Hence, a representative approximation is to choose $k * WI$ cycles. A preliminary number of simulations have been done to choose an appropriate value of WI . We have noticed that the mean of WI when considering edges without weight is

approximately 100 whereas it is approximately 1000 when considering the weighted graph. We fixed a value of 10 for k so that each node will be visited at least 10 times. Therefore, each simulation will run for 1000 cycles (10 x 100) for the non-weighted graph and 10000 cycles (10 x 1000) for the weighted graph.

At the beginning of a simulation, we consider that all instantaneous node idleness is zero, as they had just been visited. Consequently, there is a sort of transitory phase in which the instantaneous graph idleness tends to be low, not corresponding to the reality in a steady-state phase. For this reason, the (final) graph idleness is measured only during the stable phase. According to some early experiments, we have noticed that the transitory phase always finishes before the ET (Exploration Time), which is approximately 100 cycles for the non-weighted graph and 1500 cycles for the weighted graph.

5.3 Result Graphics

In the following graphics, each different line type represents a different MAS architecture. At first, we compare agents which combine idleness and distance in the decision making process and their corresponding idleness-based versions for the weighted graph shown in Figure 1 and its non-weighted version. Figure 2 and Figure 3 show these comparisons.

We compare the three classes of architectures where the impact was stronger: Conscientious Cognitive Agent and Heuristic Conscientious Cognitive Agent with idleness rate of 40%; Cognitive Blackboard Agent and Heuristic Cognitive Blackboard Agent with idleness rate of 60%, Cognitive Coordinated Agent and Heuristic Cognitive Coordinated Agent with idleness rate of 40%.

For comprehension (of the graphics) purposes we have adopted these abbreviations: A – Agent, B – Blackboard, C – Cognitive, Con – Conscientious, Cor – Coordinated, H – Heuristic, P – Pathfinder. The real value(s) on the right of heuristic agents abbreviations is their respective idleness rates.

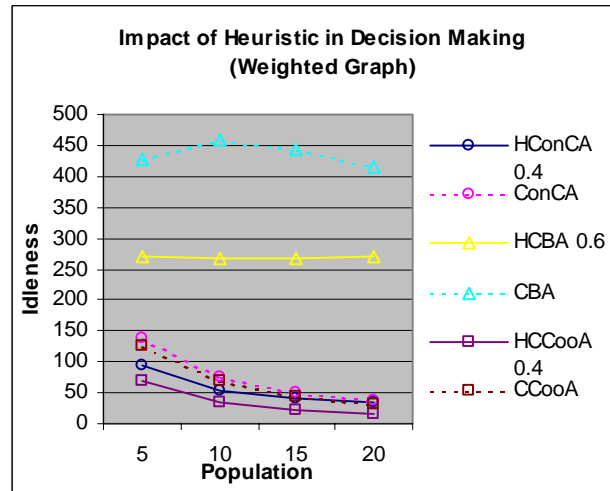


Figure 2 – The Graphic shows the impact of considering agents with heuristic in a weighted graph.

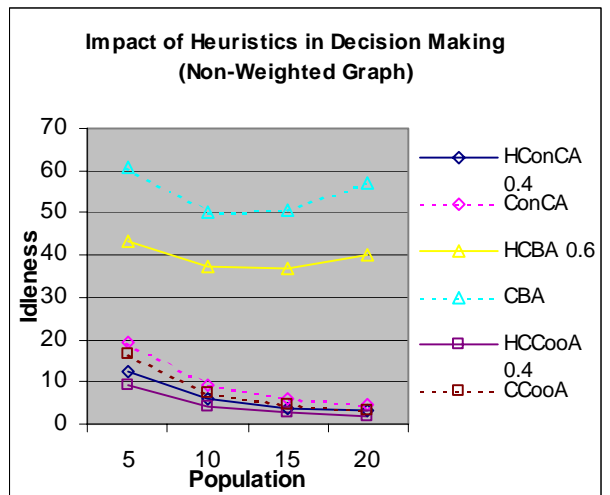


Figure 3 – The Graphic shows the impact of considering agents with heuristic in a non-weighted graph.

Now, we compare agents which also take into account the idleness of the intermediary nodes in the pathfinding with their corresponding versions which only consider the distance for the weighted graph shown in Figure 1 and its non-weighted version. Figure 4 and Figure 5 show these comparisons.

Again, we compare the three architectures where the impact was stronger: Heuristic Cognitive Coordinated Agent with idleness ratio of 40% and Heuristic Pathfinder Cognitive Coordinated Agent with idleness rates of 40% and 20% for decision making and pathfinding, respectively; Cognitive Blackboard Agent and Pathfinder Cognitive Blackboard Agent with idleness rate of 40% for pathfinding; Heuristic Conscientious Cognitive Agent with idleness ratio of 40% and Heuristic Pathfinder Conscientious Cognitive Agent with idleness rates of 40% and 80% for decision making and pathfinding, respectively.

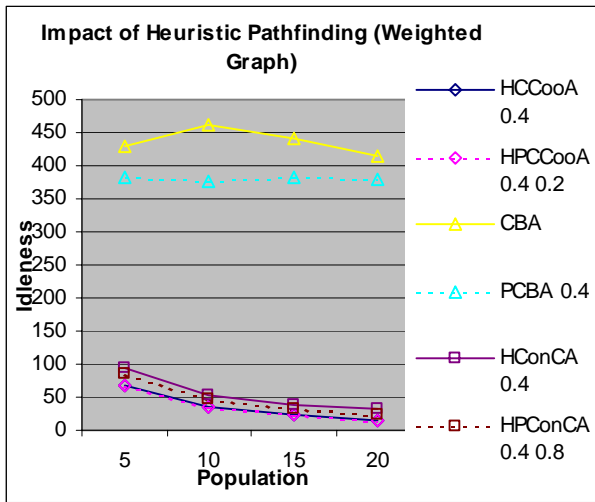


Figure 4 – The Graphic shows the impact of adding an heuristic pathfinding component on agents in a weighted graph.

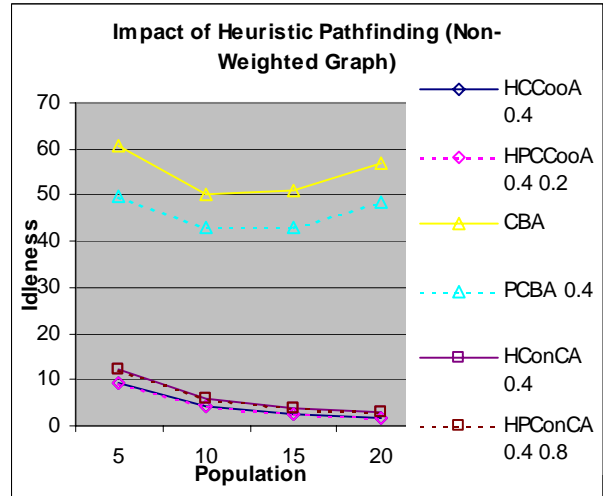


Figure 5 – The Graphic shows the impact of adding an heuristic pathfinding component on agents in a non-weighted graph.

5.4 Discussion

As expected, we noticed that agents which combine idleness and distance in the decision making process are better than their corresponding idleness-based versions. This occurs in both weighted and non-weighted graphs. In some cases (coordinated agents), we were able to improve the performance in about 50%.

When we compare agents which also take into account the idleness of the intermediary nodes in the pathfinding with their corresponding versions which only consider the distance, we can also notice an improvement in the performance. In some cases (conscientious and blackboard cognitive agents), we were able to improve the performance in about 25%.

As in our previous work, random agents do not perform very well. The same applies to agents with shared node idleness (Reactive with flags and Blackboard Cognitive and their heuristic variations) since every agent tends to go to the same places at the same moment. Consequently, those groups somehow behave as a single agent.

The usage of coordination is another means of improving performance, as noticed when

comparing Conscientious Cognitive Agents and the corresponding Cognitive Coordinated Agents.

In fact, the best architecture is the most sophisticated one: the one which uses explicit coordination and combines idleness and distance in the decision making process and in the pathfinding (Heuristic Pathfinder Cognitive Coordinated Agent).

6 Conclusions

This work presents some original contributions to the problem of multi-agent patrolling. We have proposed some new MAS architectures in order to enhance our preliminary typology (this typology follows the same approach we have been using in other tasks¹⁴).

The simulator we have developed is available upon request for other researchers wanting to experiment their patrolling strategy.

In the future, we intend to augment the complexity of the agent and MAS architectures. This includes features such as statically partitioning of the graph into regions by graph theory algorithms. We will also enable messages exchange among all agents, opening the opportunity to explore negotiations mechanisms for conflicts resolution.

Additionally, we are working in a taxonomy of patrolling problems in order to study similar problems such as the detection of static and mobile targets.

Finally, we will study how the topology of the graph influences the performance of the agents. In order to accomplish this task, we will propose graphs with different characteristics, such as: levels of connectivity, presence/absence of cut nodes and cut edges, variance of the edge-weights, presence of clearly distinct interconnected regions, amount of nodes and graphs based on real-world environments (commercial centers, cities ...).

7 References

1. Abate, F.R. The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers, Oxford Univ. Press, 1996.
2. Minar, N., Hultman, K. & Maes, P. Cooperating Mobile Agents for Mapping Networks, In the Proceedings of the First Hungarian National Conference on Agent Based Computing, 1998.
3. Arthur, W.B. Inductive Reasoning and Bounded Rationality (The El Farol Problem), American Economic Review, Vol. 84, pp. 406-411, 1994.
4. Arkin, R.C. Behavior-Based Robot Navigation for Extended Domains. Adaptive Behaviors, Vol. 1, pp. 201-225, 1992.
5. Balch, Tucker & Arkin, R.C. Behavior-Based Formation Control for Multi-robot Teams, IEEE Transactions on Robot and Automation, 1999.
6. Reynolds, C.W. Steering Behaviors for Autonomous Characters, Game Developers Conference, 1999.
7. Howland, G., A Practical Guide to Building a Complete Game AI, http://www.lupinegames.com/articles/prac_ai_2.html (08/08/2003).
8. Pottinger & Dave, C., "Coordinated Unit Movement," in *Game Developer*, vol. January, 1999, pp. 42-51.
9. Pottinger & Dave, C., "Implementing Coordinated Unit Movement," in *Game Developer*, vol. February, 1999, pp. 48-58.
10. Machado, A., Ramalho, G., Zucker, J.-D. and Drogoul, A. Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures. Multi-Agent Based Simulation (MABS'2002), Bologna, 2002.
11. Drogoul, A. & Collinot, A. Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: a Case Study in Robotic Soccer, *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 1, pp. 113-129, 1998.
12. Dorigo, M., Maniezzo, V. & Coloni, A. The Ant System: optimization by a colony of cooperating agents, *IEE Trans. System, Man and Cybernetics*, Vol. B26, pp. 29-41, 1996.
13. Tangamchit, P., Dolan, J. M., Khosla, P. K.: Learning Based-Task Allocation in Decentralized Multirobot Systems
14. Zucker, J.-D. and C. Meyer. Apprentissage pour l'anticipation de comportements de joueurs humains dans les jeux à information complète et imparfaite: les "Mind-Reading Machines". *Revue d'Intelligence Artificielle* 14(3-4). (2000). 313-338
15. Rabin, S. *AI Game Programming Wisdom*, 2002
16. DeLoura, M. *Game Programming Gems*, 2000
17. Riedmiller, M., Merke, A. *Using Machine Learning Techniques in Complex Multi-Agent Domains, in Perspectives on Adaptivity and Learning*, LCNS, Springer, 2002