Fighting Fire with Agents

Daniel Moura Eugénio Oliveira

Faculdade de Engenharia da Universidade do Porto,
Rua Doutor Roberto Frias, 4200-465 Porto, Portugal,
{daniel.moura,eco}@fe.up.pt

Abstract

In this paper we propose a model for coordinating teams of computational agents. This model is especially aimed for coordinating agents performing in a simulated environment of forest firefighting, although it may be used in other domains. We will start by introducing the Pyrosim platform where we are carrying out our experiments. Pyrosim is a tool developed in our laboratory that simulates a forest fire environment where software agents act under the role of firefighters that have to cooperate in order to control the fire. We will proceed by presenting a model for team coordination. With this model it is possible to define firefighting tactics that originate different team approaches to the fire. These tactics are conducted by a single agent (the Leader) that communicates high level tasks to the other agents. Agents have local autonomy and are able of cooperating locally for carrying out their tasks without using communication. Finally, we will present some results of our experiments using the proposed coordination model in two different scenarios. We will use these results to address the problem of automatic tactic selection where we are currently working on.

1 Introduction

Forest fires are an everyday problem of society. Considering South European countries only (Portugal, Spain, France, Italy and Greece), in the year of 2005, forest fires burned more than 556 thousand hectares of forest [1]. This problem particularly affects Portugal where more than 325 thousand hectares of forest burned for the same year, which is more than an half of the total burned area in the South European countries.

Despite the seriousness and challenges of this problem, there is not much work in this domain in the area of Artificial Intelligence. In fact, we only found one work that uses a Multi-Agent System (MAS) to tackle the problem of coordinating a firefighting team to attack a forest fire. This work in being developed by Wiering *et al.* [10, 11] and is concerned with coordinating heavy machinery (bulldozers) to build a line around the fire to prevent it from spreading. The authors use machine learning to elaborate plans according to the scenario situation, which are distributed to agents (bulldozers) at the beginning of the simulation. However, in Portugal and other countries, using heavy machinery is most of the times impossible because of the terrain geography that is highly irregular. Additionally, heavy machinery may not be always available, or may not be the best solution (e.g. in small fires). Therefore, we choose to coordinate a team of firefighters that use water jets to try putting out the fire, although we intend to include other kinds of firefighting agents in the simulation platform.

In this paper we present a model for coordinating a Team of computational Agents that performs in a Forest Firefighting simulator. Agents play the role of firefighters that have to combat fire in an organized way in order to control it. With the proposed model it is possible to define Firefighting Tactics similar (although simplified) to the ones that are used by real firefighting teams. Additionally, it is possible to experiment new tactics or variations of commonly used tactics for trying to understand the outcome of using that tactics in different scenarios. For accomplishing this, the model provides a flexible structure that enables to define different approaches of the team to the fire.

The remainder of this paper is structured in the following way: in section 2 we present the forest firefighting simulator that we are using for our experiments. Then, in section 3 we describe the proposed coordination model for controlling the overall team behaviour. We proceed to section 4 where we detail how agents are able to cooperate locally in order to execute shared tasks. We continue to section 5 where we present some results of our experiments and point out future work directions for automatic tactic selection. Finally, we present our conclusions about this work.

2 The Pyrosim Platform

Before we present the coordination model, we will introduce the Pyrosim Agent platform [7] that we have been using in our experiments. The Pyrosim platform simulates a forest-fire environment where a team of Agents (firefighters) cooperates to control and extinguish the fire, while simultaneously trying to minimize the overall damage and losses. In Pyrosim, Agents have to deal with very dynamic fire fronts, terrain constraints and their own physical and logistic limitations. Each Agent needs to ensure its physical safeness while trying to fight the fire and, at the same time, help colleagues to remain safe. Agents are equipped with a water jet with limited power that allows them to put out the fire, but they are not normally able to do it individually so there is an obvious need for cooperation. Agents may communicate with each other in order to organize team efforts (broadcast and 1-to-1 messages). Agent's Perception System provides information about his own state (physical energy, speed, acceleration, position in the terrain, status of the personal water jet) as well as several matrix structures named *Visual Maps* that describe close range and medium range surroundings. Visual Maps contain information with different *levels of detail* and *noise* (depending on the distance) about terrain, vegetation, level of destruction, and fire cells. Visual Maps may have cells where no information is available because of the occlusion effect (for instance, when an agent is climbing a hill it cannot see to the other side of the hill). Agents also receive information about visible parameters of other Agents (location, approximate energy level and action). Figure 1 shows a visualization of a simulation in the Pyrosim simulator. Pyrosim creates a complex environment that may be used as testbed for team coordination models, and for simulating firefighting tactics as well. Additional information about the Pyrosim platform and simulation model may be found at [7].

3 Overall Team Coordination

In our approach, team coordination is centralized, although loosely coupled, which means that the overall team behavior is controlled by one agent, the *Leader*, but every agent has local autonomy. The Leader responsibilities include being aware of the global situation, reasoning about it, and assigning specific tasks to agents in order to carry out a given plan. There are several reasons that impelled us to use centralized coordination:

- Real firefighting teams use centralized coordination [3];
- One agent, the Leader, may have access to the global situation and make decisions based on global knowledge. In a decentralized solution, the information would be dispersed, and no agent would have a complete understanding of the situation. The alternative would be for every agent to be aware of the other agents' knowledge, but that would increase drastically communications between agents.
- Centralizing tactical/strategic decisions let other agents to focus on their tasks execution.
- Using centralized coordination is not synonym of the Leader being a master ruling a set of slaves. Agents may have local autonomy and are not obliged to carry out tasks that go against their own goals.

Of course that centralizing decision has its drawbacks. First, there is the danger of creating a bottleneck on the Leader agent. To avoid this, teams should not have too many agents, and communication should be used only when needed. Additionally, when coordinating larger teams, one should use a hierarchy of command to ensure that a Leader only controls directly a small set

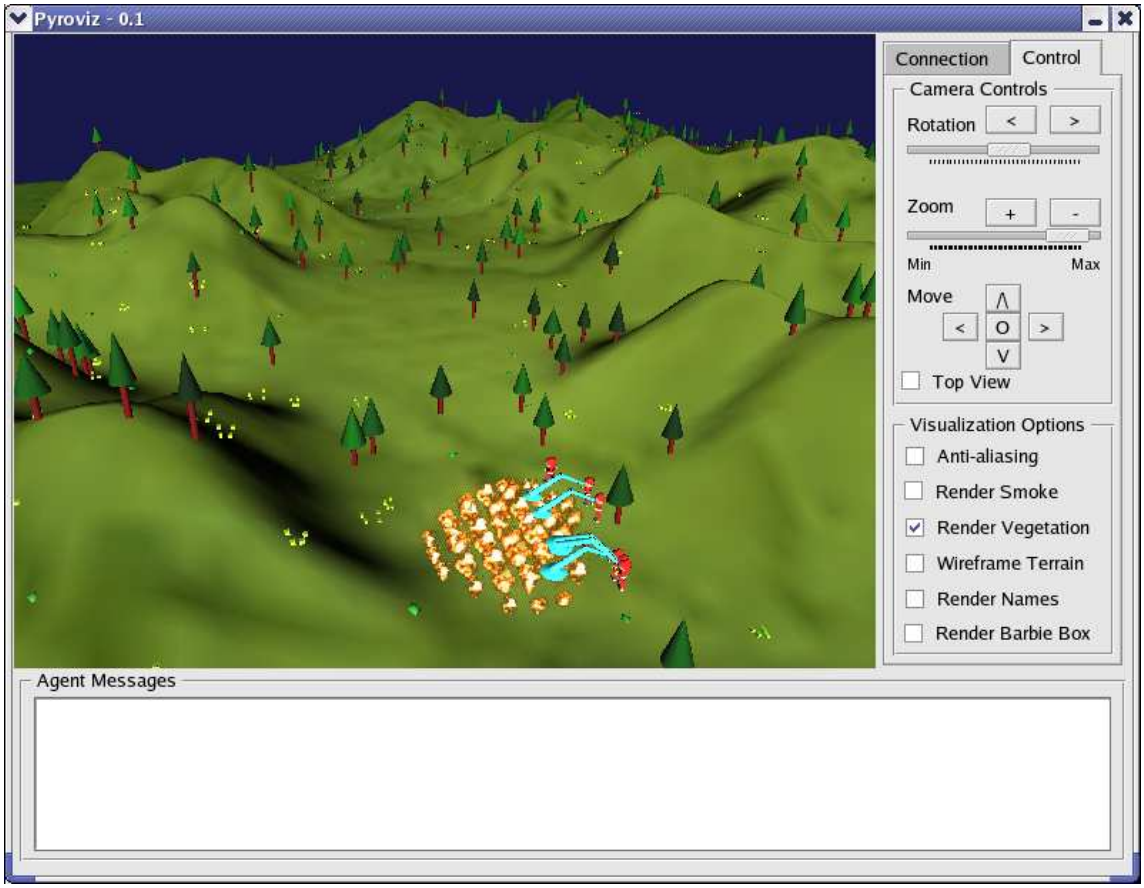


Figure 1: A visualisation of a simulation in the Pyrosim platform

of agents. Another problem with centralized coordination is that if communication fails or if the Leader fails, the team coordination is lost. To prevent the team from losing its Leader we configured the Leader agent's "personality" in order to obtain a more cautious behavior. As result the Leader keeps a higher distance from the flames and takes less chances during the firefight. Currently, there is no mechanism in our implementation for replacing the Leader in case of Leader failure. However, this is not a big issue because our goal is to simulate firefighting tactics and not to create a system for using during real firefighting.

In the next subsections we will make a formal description of the proposed model and then we will instantiate it to the forest firefighting domain.

3.1 Coordination Model

For developing the coordination model, we used the same principle that was used in the robotic soccer teams by Stone and Veloso [8, 9], and by Reis and Lau [4, 5, 6]. Both approaches defined the team spatial distribution using *roles*, which were then assigned to agents. Roles enable to define generic team formations that may be used with any team of agents. Despite we are using the same principle of these teams, we had to build a model that would be able to handle some issues that are not present in the robotic soccer domain, such as the actuation area size is not delimited (on the other hand, the soccer field has fixed dimensions), the number of agents that constitutes a team is not fixed (on the other hand, soccer teams have fixed size), and the opponent may have very different configurations (fire size, shape and properties may vary a lot).

We will now present a formal description of the proposed model. Every agent has a *Role* (3). There is a predefined number of Roles (2) which define different agent behaviors. The number of Roles is not related with the number of agents. Teams may be homogeneous (all agents have the

same Role), or heterogeneous (agents playing different Roles).

$$Agents = \{Agent_1, Agent_2, Agent_3, \ldots, Agent_{nagents}\} \tag{1}$$

$$Roles = \{Role_1, Role_2, Role_3, \ldots, Role_{nroles}\} \tag{2}$$

$$AgentRole_i \in Roles \quad \forall i = 1..nagents \tag{3}$$

There are several ways of organizing agents to attack fire. Fire may be attacked directly in the tail, head or flanks, or it may be attacked indirectly by constructing lines outside the fire that limit its progression. To implement these attacks we have defined a set of *Attack Plans* (4) that specify how agents should approach fire.

$$AttackPlans = \{AttackPlan_1, \ldots, AttackPlans_{nattacks}\} \tag{4}$$

In the proposed model, an Attack Plan defines a sequence of *Tasks* for a given Role (7). In this way, agents that share the same role will have the same tasks to carry out. In general, an Attack Plan may be carried out by one Role only, although there are Attack Plans that may be executed by a given set of Roles (6).

$$Tasks = \{Task_1, Task_2, Task_3, \ldots, Task_{ntasks}\} \tag{5}$$

$$AdmissibleRoles_i \subseteq Roles \quad \forall i = 1..nattacks \tag{6}$$

$$\begin{gathered} AttackPlan_i = \{Role_j, AttackTask_1, \ldots, AttackTask_{nattacktasks}\} \\ \forall i = 1..nattacks \quad Role_j \in AdmissibleRoles_i \quad AttackTask_k \in Tasks \end{gathered} \tag{7}$$

A firefighting crew may perform different attacks simultaneously. To define the attacks distribution among the team, the concept of *Tactic* was created (8). A Tactic defines for each Role how many agents will play that Role, and which Attack Plan will be performed by those agents (9).

$$Tactics = \{Tactic_1, Tactic_2, Tactic_3, \ldots, Tactic_{ntactics}\} \tag{8}$$

$$\begin{gathered} Tactic_{i,j} = \{TacticAttack_j, AgentAssignment_j\} \qquad \forall i = 1..ntactics \\ \forall j = 1..nroles \qquad TacticAttack_j \in AttackPlans \\ AgentAssignmet_j \in [0..1] \quad \sum_{j=1}^{nroles} AgentAssignment_j = 1 \end{gathered} \tag{9}$$

From (9), we gather that agent assignment to Roles is defined using relative quantities (e.g. allocate half of the team to a given Role). However, in our implementation, it is also possible to define agent distribution using absolute quantities (e.g. allocate 2 agents to a given Role). Additionally, agent distribution may be defined dynamically at run-time, which enables to product more complex role assignments.

It is also possible to define Tactics that change the team approach over time. We call these tactics *Dynamic Tactics* (10), and they are composed by a set of Tactics with *Activation Conditions* (11). Activation Conditions define when to switch to a given tactic.

$$DynamicTactics = \{DynamicTactic_1, \ldots, DynamicTactic_{ndynamictactics}\} \tag{10}$$

$$\begin{gathered} DynamicTactic_i = \{ActivationCondition_1, SubTactic_1, ..., \\ ActivationCondition_{nsubtactics}, SubTactic_{nsubtactics}\} \\ \forall i = 1..ndynamictactics \quad SubTactic_j \in Tactics \end{gathered} \tag{11}$$

All tactics are pre-defined at the Leader agent level, and therefore, only the Leader has knowledge about tactical information. The other agents only know their Role and the tasks that they must execute, which are assigned by the Leader using tactical information.

3.2 Defining Attack Plans for Firefighting

For instantiating the proposed model to forest firefighting, we start by defining Attack Plans. We have divided Attack Plans in two major classes: (i) Direct Attacks, and (ii) Indirect Attacks. This division is based on firefighting theory that differentiates Direct from Indirect Attacks [3]. Direct Attacks involve fighting the fire directly in the flames using water or manual tools, like shovels, to swat the flames. Therefore, this kind of Attack Plans specify tasks for approaching the fire and then attack it directly. On the other hand, Indirect Attacks are used to fight fire at distance, especially when the fire is too intense for firefighters to approach it. The most common technique is to build a *fireline*. Firelines are built by dogging the ground to remove all the vegetation in front of the fire to starve it out of fuel. In the current version of the fire simulator it is not possible to built firelines. However, we implemented another kind of indirect attack that consists in creating a *wet-line*. Instead of dogging the ground, firefighters wet the ground with large quantities of water. The effect is similar to building a fireline, although this is just a temporary solution until the simulator supports dogging operations.

Figure 2 illustrates a class diagram of the Attack Plans that we implemented, where the classes with no coloring represent Attack Plans that may be instantiated, and the classes colored in grey represent abstract classes that are used for structuring proposes only. In general terms, Direct Attacks define how to attack a given part of the fire (e.g. the head, the tail or the flanks) and the *WetLineAttack* defines how to create a Wet-Line in a given area.

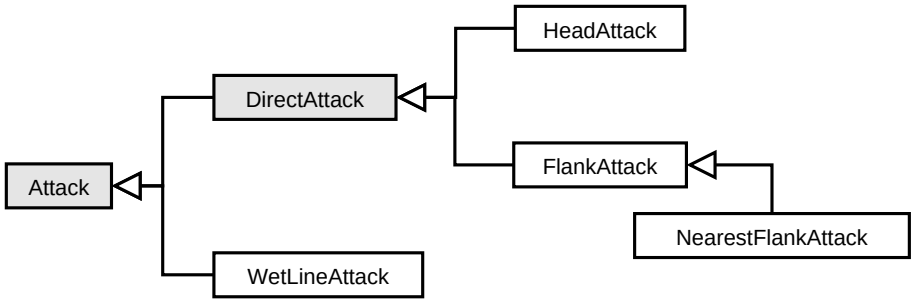


Figure 2: Implemented Attack Plans

Regardless of the kind of attack, Attack Plans always have two stages: the positioning and the attack management. The positioning stage is concerned with placing the agents in positions that enable them to start the attack in good conditions. Once agents are in position, the attack management stage is activated. This stage is concerned with allocating tasks to agents that are performing the attack. In the beginning, agents receive tasks to attack a given area. Every time agents complete their tasks, they receive new tasks to attack a new target area. Every Attack Plan has to define its preferences for positioning and managing the attack. For instance, in a Head Attack it is defined that agents should start by approaching the Head of the fire in the opposite direction to the fire propagation direction. Then, when they arrive to the target area, they should start attacking the foremost burning cell. Finally, when they extinguish the current cell, they should attack the foremost adjacent cell that is burning.

3.3 Defining Firefighting Tactics

As we have seen above, tactics define the Role of every agent and the Attack Plan that the agent will carry out. Using the Attack Plans presented in the previous subsection we are able to build a considerable set of tactics. In table 1 we present some examples of Static Tactics. These tactics may be very simple like tactic ST1 that assigns the same Role and Attack Plan to every agent. Moreover, we may have tactics with the team divided for performing different kind of attacks simultaneously. If we need more flexibility we may define Dynamic Tactics that have the ability to change the team configuration according to the current situation. For instance, tactic DT2 (table 2) defines that when the team arrives to the fire it should build a Wet-Line around fire, and when the fire is controlled all firefighters should concentrate efforts to attack the tail. Additionally, it is

possible to specify Dynamic Tactics that select the best a approach automatically according to the scenario evaluation.

Tactic	*Role*	*Attack*	*Distribution*
All in the Tail (ST1)	tail_attacker	Nearest Flank	All
Split by Flanks (ST4)	left_flank_attacker right_flank_attacker	Flank Flank	1/2 Rest
Surrounding Wet-Line (ST6)	left_wet_line_builder right_wet_line_builder tail_wet_line_builder head_wet_line_builder	Wet-Line Wet-Line Wet-Line Wet-Line	1/4 1/4 1/4 Rest
Surrounding Wet-Line with Tail Attack (ST7)	left_wet_line_builder right_wet_line_builder tail_wet_line_builder head_wet_line_builder tail_attacker	Wet-Line Wet-Line Wet-Line Wet-Line Nearest Flank	1/5 1/5 1/5 1/5 Rest

Table 1: Examples of static tactics

An important issue regarding Dynamic Tactics is the process of role allocation when switching from one tactic to another. Role allocation is done having in consideration (i) the current location of firefighters and (ii) the target location associated to the positioning stage of the Attack Plan associated to a given role. In this way, the role allocation algorithm tries to distribute roles by firefighters in order to minimize the distance that firefighters have to run to get to their new target location. The algorithm starts by doing a greedy allocation which is then optimized using a simple local search. Currently, the local search tries to minimize the maximum distance that a firefighter has to travel, but other approaches are possible like minimizing the total distance traveled by all firefighters.

Activation Condition	→	Sub Tactic
When simulation starts	→	Surrounding Wet-Line
When fire stabilizes	→	All in the Tail

Table 2: Example of a dynamic tactic: Wet-Line and then Attack Tail

4 Local Coordination

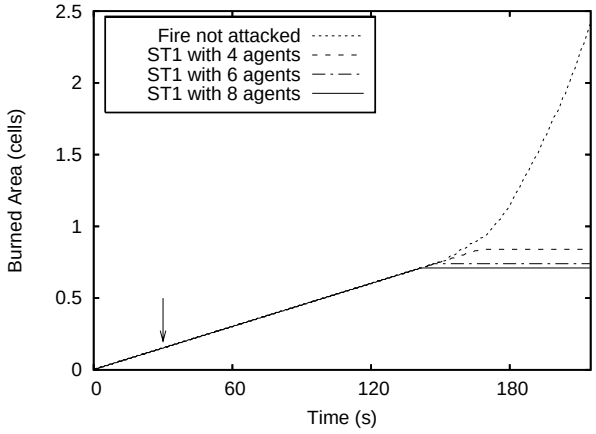
During an attack, agents that share the same Role also share the same tasks. To execute these tasks efficiently they must coordinate their actions. For instance, when two firefighters are attacking two adjacent cells simultaneously (one firefighter per cell) they have more difficulties putting the fire down compared to when they first attack together one of the cells and then the other. To achieve this cooperative behavior we implemented a Local Coordination mechanism for direct attacks. This mechanism uses perception about other agents, and predefined Local Coordination rules.

When agents receive an assignment for directly attacking a given area, they must choose which cell to attack first. This decision may depend on several factors, such as the distance to the target cell, the fire intensity in that cell, the danger of attacking that cell, and if the cell is being attacked by a teammate or not. When an agent enters in the target area, the first thing he does is to verify if another agent is already attacking the cell. If there is, the agent goes to the teammate location in order to help him fighting that fire. Otherwise, the agent selects the nearest cell inside that area and attacks it. If two agents start attacking two different cells simultaneously, two problems arise: (i) the agents are not coordinating efforts, and (ii) the other agents do not know which agent to help. To solve these problems, agents have predefined rules for handling this kind of conflicts. These rules define criterions for evaluating the quality of the agent position to determine which agent to help. The quality of the position is calculated based on the agent progression inside the

target area. If two or more agents are in positions with the same quality, the agent's name is used to resolve the conflict.

5 Experimental Results and Tactic Selection

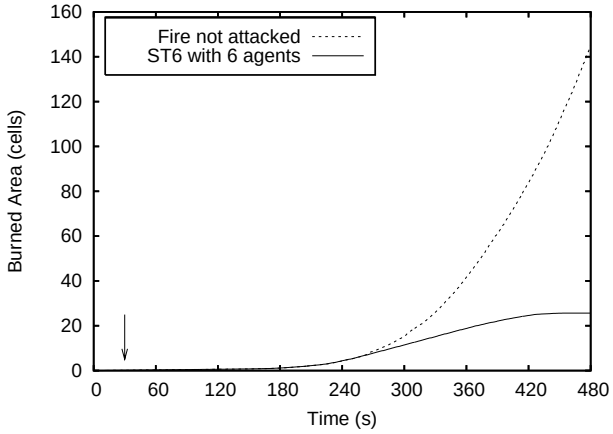
Currently, the developed system works as a platform to test tactics in different scenarios. However, our next step is to enhance the platform in order to automatically determine the tactics that work best in different scenarios. We have already made some experiments that show that there are tactics that achieve positive results in situations where others fail, although the same tactics achieve negative results in situations where others succeed [2]. We are able to observe this in the following example where we present the results of experimenting two different tactics (ST1 and ST6) in two different scenarios (A and B). Tactic ST1 tries to attack the fire by positioning all agents behind the tail of the fire and giving them orders to attack the fire directly with water. Tactic ST6 places firefighters around the fire perimeter and give them orders to wet the ground in order to create a Wet-Line. We have selected these tactics in order to represent (i) tactics mainly based in direct attacks (ST1), and (ii) tactics mainly based in indirect attacks (ST6), which we know from firefighting theory that should be applied to fires with different dimensions (among other factors such as the number of firefighters available, terrain properties and weather conditions). We experimented both tactics in two scenarios with the same characteristics (in terms of terrain geometry, vegetation and weather), but in scenario A agents received the warning sign sooner than in scenario B, and therefore when firefighters arrived to the fire in scenario B they would face a fire in a more advanced state than in scenario A. We observed that both tactics succeeded in scenario A (figures 3 and 4) and that tactic ST1 achieved better results. This happened because in tactic ST1 firefighters start by attacking fire directly and therefore they are able to prevent it from spreading at an earlier stage. However, in scenario B this same tactic (ST1) failed to control fire from spreading (figure 5). ST1 tactic was a good choice for scenario A but that is not the case for scenario B. In scenario B firefighters arrive to the fire at a later stage where using a direct attack is not enough to control fire. On the other hand, tactic ST6 was able to achieve a more stable performance by sacrificing the same area in both scenarios. In this tactic firefighters build a wet-line around the fire area that prevents fire from spreading, instead of fighting the fire directly. Therefore, the total burned area in both scenarios was nearly the same, but in scenario A the damage was excessive. More information about this and other experiments may be found in [2]. An important output from these experiments is that results are coherent with forest firefighting theory in terms of the applicability of direct and indirect attacks.



# Agents	Burned Area	Elapsed Time
4	0.83	186
6	0.76	173
8	0.74	169

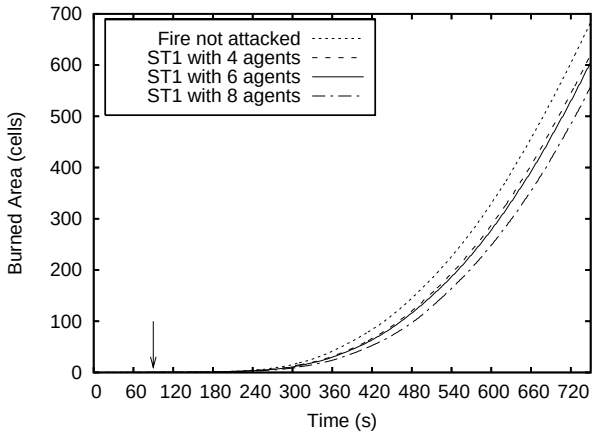
Figure 3: Scenario A: Burned area using the ST1 tactic (All in the Tail)

The experiment described above is just an example of the influence of the scenario configuration in tactic selection. Much others factors like the terrain geometry, the type of vegetation in the terrain, and the weather may influence tactical decisions. Therefore, one of our main lines for future work is to use machine learning for figuring out which tactics are best according to the scenario properties.



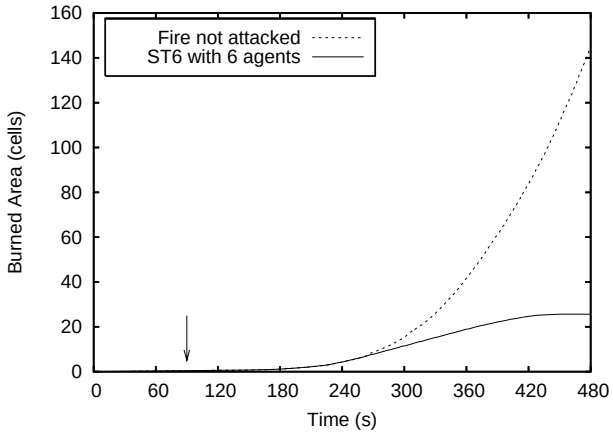
# Agents	Burned Area	Elapsed Time
4	25.69	473
6	25.57	473
8	25.66	474

Figure 4: Scenario A: Burned area using ST6 tactic (Surrounding Wet-Line)



# Agents	Burned Area	Slow Down
4	625.49	9%
6	606.30	12%
8	569.60	17%

Figure 5: Scenario B: Burned area using ST1 tactic (All in the Tail)



# Agents	Burned Area	Elapsed Time
4	26.10	482
6	25.66	473
8	25.69	474

Figure 6: Scenario B: Burned area using ST6 tactic (Surrounding Wet-Line)

6 Conclusions

In this paper we have presented a model for coordinating a team of agents. We have instantiated this model to the forest firefighting domain where we were able to implement some tactics similar to the ones that real firefighting teams use. The model we presented enables to define much other tactics in a rather flexible way. In our implementation, the overall team coordination is controlled

by a single agent, the Leader, who is responsible for carrying out tactics. However, we have shown that agents have local autonomy and are able to cooperate locally without the Leader intervention.

We also presented some results of our experiments that demonstrate that, like in reality, different fire scenarios require different firefighting tactics in order to minimize fire damage. Additionally, the tactics that performed best in the tested scenarios are the ones that we were expecting according to firefighting theory. However, we are aware that we are still far from providing meaningful information for real firefighting teams. For achieving this, the simulator should be properly validated and more experiments should be done.

We also pointed out an interesting line of research regarding possible uses of machine learning for automatic tactic selection based on the results of tactic experimentation in different scenarios.

Acknowledgements

The authors would like to thank to Luís Sarmento for his great contribution in the development of the Pyrosim Agent Platform and of the Firefighting Agents Architecture.

References

[1] Divisão da Defesa da Floresta Contra Incêndios. Incêndios florestais – relatório de 2005. Technical report, Ministério da Agricultura do Desenvolvimento Rural e das Pescas, Portugal, January 2006. http://www.dgrf.min-agricultura.pt/v4/dgf/pub.php?ndx=2271.

[2] Daniel Moura. Coordinating a team of agents in the forest firefighting domain. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2006. http://dcm.web.simplesnet.pt/publications/MSc_dcm_0 6.pdf.

[3] Marc Nicolas and Grant Beebe. The training of forest firefighters in indonesia. Technical report, German Agency for Technical Cooperation, European Union, and Government of Indonesia, 1999.

[4] Luís Paulo Reis. *Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, June 2003.

[5] Luís Paulo Reis and Nuno Lau. Fc portugal team description: Robocup 2000 simulation league champion. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors, *RoboCup 2000: Robot Soccer World Cup IV*, pages 29–40, London, UK, 2001. Springer-Verlag.

[6] Luís Paulo Reis, Nuno Lau, and Eugenio Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In Markus Hannebauer, Jan Wendler, and Enrico Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions)*, pages 175–197, London, UK, 2001. Springer-Verlag.

[7] Luís Sarmento. An emotion-based agent architecture. Master's thesis, Faculdade de Ciências da Universidade do Porto, 2004.

[8] Peter Stone. *Layered Learning in Multi-Agent System*. PhD thesis, School of Computer Science, Carnegie Mellon University, December 1998.

[9] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[10] Marco Wiering and Marco Dorigo. Learning to control forest fires. In H. Haasis and K. Ranze, editors, *Proceedings of the 12th international Symposium on 'Computer Science for Environmental Protection'*, pages 378–388, 1998.

[11] Marco Wiering, Fillipo Mignogna, and Bernard Maassen. Evolving neural networks for forest fire control. In M. van Otterlo, M. Poel, and A. Nijholt, editors, *Benelearn '05: Proceedings of the 14th Belgian-Dutch Conference on Machine Learning*, pages 113–120, 2005.