Learning to Control Forest Fires

Marco Wiering[1] and Marco Dorigo[2]

Abstract

Forest fires are an important environmental problem. This paper describes a methodology for constructing an intelligent system which aims to support the human expert's decision making in fire control. The idea is based on first implementing a fire spread simulator and on searching for good decision policies by reinforcement learning (RL). RL algorithms optimize policies by letting the agents interact with the simulator and learn from their experiences. Finally, we observe different problems and propose solutions for solving them. Among these problems are storing policies for huge state spaces and coping with multiple agents which need to learn cooperative strategies.

1. Introduction

Forest fires. Forests play a crucial role for sustaining the human environment and because forest fires are among the largest dangers for forest preservation, it is not a surprise to see increasing state expenditures for forest fire control. Despite of this, annually millions of hectares of forests are still destroyed by fires. That this number has not declined implies that controlling fires is a complex task, and indeed forest fires can become as large as 600 km² within 9 days and cost millions of dollars to extinguish (The Petawawa NF Institute 1982). Although most fires are extinguished quickly, a few forest (or bush) fires become uncontrollable for human intervention after which they cause huge damages to the environment and endanger human lives. E.g., Ash Wednesday, the SA & Victoria (Australia) fire disaster in 1983, burned 392 000 ha of (grass) land and killed 75 people (Moore/Trevitt 1991).

Environmental decision support. Environmental protection receives growing interest from the computer science community. Research ranges from using information technology (IT) for efficiently recording, processing and distributing environmental data to modelling and simulation of environmental processes. Environmental decision support systems (EDSSs) are systems which integrate databases and models and which support a user in her decision making regarding environmental problems (Rizzoli/Young 1997). Environmental problems involve many interacting

[1] IDSIA, Corso Elvezia 36, 6900-CH Lugano
email: marco@idsia.ch, Internet: http://www.idsia.ch/~marco
[2] IRIDIA, Université Libre de Bruxelles, Avenue Franklin Roosevelt 50, 1050 Bruxelles
email: mdorigo@ulb.ac.be, Internet: http://iridia.ulb.ac.be/dorigo/dorigo.html

subprocesses, which makes choosing good decisions in real time very difficult for human experts. Therefore, some EDSSs have been developed which contain intelligent subsystems for advising the user. Examples of intelligent subsystems are expert systems, planning tools, and optimization methods. In this paper we will consider the latter.

Complex problem solving. Complex problem solving refers to a broad class of methods which may be used to compute solutions for problems which are hard to solve by human experts. Well known examples are the problems in combinatorial optimization such as the traveling salesman problem (TSP). A wide variety of algorithms for finding solutions to such problems have emerged from operations research (OR) and artificial intelligence (AI) during the last decades, examples are genetic algorithms (Holland 1975), tabu search (Glover/Laguna 1997) and the ant colony system (Dorigo et al. 1996, Dorigo/Gambardella 1997). These algorithms are currently being applied to real world problems such as scheduling containers (Gambardella et al. 1998), and when wisely applied result in both economical as ecological gains.

Steps in forest fire control. Forest fires depend on three things: fuel, heat and oxygen. Taking away one of them will put out the fire. Methods (resources) for controlling the fire can be divided into airborne agents and ground agents. Airborne agents such as airplanes and helicopters drop water or chemical retardants in front of the fire and take away heat or oxygen. Ground agents such as trucks or land rovers are equipped with water tanks for directly attacking the fire. Other ground agents are bulldozers, tractors or people equipped with e.g. chain saws. These agents cut firelines, which is an effective means for removing fuel. The choice of methods and equipment which are actually used depends on the country and kind of environment (Calabri 1982).

Once a fire outbreak is signalled, the fire manager evaluates the situation and makes an initial attack plan to stop the spread of the fire. This plan consists of a number of firelines (subplans) which break the fire-propagation. Then she allocates resources from neighboring resource bases to fulfill all subplans. Once the resources have started the fight, the fieldcommander is in control. He coordinates the teams in the field and gets a stream of online information which enables him to reevaluate plans constantly, e.g. if the situation gets too dangerous, he can choose to retreat.

Fire management decision support systems. Currently, a few EDSSs have been constructed to support fire managers in their decision making (Beer 1990, Avesani et al. 1993, Kourtz 1994). The CHARADE project (Ricci et al. 1994) is a software platform for the development of intelligent EDSSs, and has e.g. been used to construct a working EDSS for managing first intervention in forest fires. The planning system integrates case-based reasoning and constraint reasoning (Avesani et al. 1993) and is integrated with a Geographic information system (GIS) for displaying spatial data and a model for simulating forest fires. Kourtz (1994) describes various applications of expert systems and AI algorithms to support managing Canadian

forest fires. Example applications are improving fire detection, computing cost-effective resource allocation and predicting forest fire occurrence.

Learning to control forest fires. We are interested in supporting the decisions during a fire fight. Since the number of possible situations is very large, it is practically impossible to design and implement good decision policies for all agents by hand. Instead, we will use reinforcement learning (RL) to *learn* reactive policies which map agent's inputs to appropriate actions. The RL methods we will use are based on learning a mapping from inputs to evaluations of actions by trial and error. These evaluations correspond to the expected cumulative cost of particular fire situations and enable selecting the actions which minimize expected cost. RL methods have been successfully applied to learning to play backgammon at human worldclass level (Tesauro 1992) and to learning to control an ensemble of elevators in a simulated building (Crites/Barto 1996).

Outline of this paper. In Section 2, we describe the forest fire simulator. In Section 3, we describe the fire fighting team. In Section 4, we describe how we are going to apply reinforcement learning. In Section 5, we conclude with a short discussion.

2. The Forest Fire Simulator

The forest fire simulator is the heart of the system, since the agents will learn their decision policies while interacting with the simulator. The simulator determines how the environmental state will change during each time step. It consists of an initial environmental state, a set of possible environmental states, and a model for fire spreading.

The environmental model. The environment is represented as a grid-based model. We have global variables and local variables. There are many global variables which need to be considered for modelling the spread of fire. We first consider only the most important global variables: *wind speed* and *wind direction*. When the wind is strong, the fire spreads much faster and we have to take this into account. The wind direction influences in which direction the fire is propagated. The local variables describe the state of each gridcell by the following attributes:

1) *Terrain* with the possible set of states: {*trees, grass, empty, water, asset*}. Here assets are special states which have to be protected at high cost.
2) *Fire activation*, a continuous variable which denotes how much the fire intensity is at a particular gridcell. It determines when a gridcell starts burning and how much heat it transfers to neighboring cells.

Fire spreading models. Developing fire spreading models is a difficult task. Data from field experiments can be used to design, calibrate and extend models. Most

approaches (Rothermel 1972, Beer 1991) for modelling the propagation of forest fires use a set of partial differential equations. The problem with these models is that they often do not take all local characteristics of the environment into account. Another modelling approach is to use cellular automata, where the dynamics are described by a set of transition functions which map the state of a local neighborhood to the change of the contents of some gridcell. We will use this latter approach.

The forest fire dynamics. We have a set of transition functions which determine: (1) Ignition levels of the different fuel-types, (2) How much activation is propagated from one state to another state given the wind speed, and (3) How the fire activation changes due the burning of the fuel. These functions need to be calibrated and may contain noise to allow for a large variety of outcomes. Due to its simplicity, the model also allows for changing wind speeds and directions during a simulation.

At time step 0, a fire starts at some gridcell. This determines the initial environmental state. This fire spreads according to the given transition functions mentioned above, and after some time (the time needed to signal the fire) the fire team is notified. From this moment on the dynamics of the forest fire is influenced by agent actions.

Incremental design. Like all modelbuilders, we have to face the trade-off between accuracy and efficiency of the model. If we want to model the forest fire dynamics very accurately, detailed knowledge about the environment should be given and the computational demands for simulating the fire would easily become overwhelming (since the states are very small). On the other hand, it may be argued that if we make the model too simple, we risk that our algorithms will not learn to control realistic fires. Therefore, we propose to incrementally increase the complexity of the model. We do not immediately include global variables such as temperature, drought, and weather conditions (rainfall) in our model. We also do not use local variables such as moisture, slope and slope direction (for mountains), and the height of flames. The simplistic model allows us to study learning fire fighting strategies without getting confused by all the details. After we have constructed algorithms which work well for the first model, we add different fuel types, forest fire types, weather types etc.

3. The Fire Fighting Team

The goal of our fire fighting team is to minimize the total cost due to the fire. This means that the fire should be put out in the most cost effective way.

The plan. Methods to control forest fires follow the following plan:

1) Create firelines which stop the fire from spreading across particular boundaries. Since the size of the area enclosed by the firelines is lost, it should be minimized.

2) Decrease the speed with which the fire spreads by dropping waterbombs in order to make plan (1) more successful.

The fire manager. The fire manager is a special kind of agent which is responsible for making and coordinating the plans of the agents. A fire manager is responsible for the first attack and for allocating resources. The first attack plays an important role, since if it is successful, then the fire can be handled easily. If it is not successful, then the fire may become uncontrollable, although the fire can still be managed (e.g. assets may be protected). Once a fire is signalled, the following steps are made:

(1) The behavior of the fire is simulated.
(2) The fire manager evaluates the situation and designs a number of firelines (usually 1 to 3) which stop the fire front from advancing.
(3) The fire manager selects and allocates resources (agent teams/equipment) for solving each subplan.
(4) The resources are brought to their starting place and execute their plans.

Ground agents. Although in reality there are various types of ground agents, we start with a single ground agent type which can only cut firelines. The ground agents can be considered as a team of five persons equipped with chainsaws etc. They are sent to their starting position by air (helicopters) from a specific resource base. At each time step, ground agents select one out of ten possible actions: an action consists of a direction for moving (including standby) and the decision whether the agent wants to cut a fireline or not. The ground agents are characterized by their traveling speed and cutting speed. The cost of using groundagents is calculated by the time they have worked and the distance the helicopter has to travel to bring them and pick them up. Finally, there is a large penalty for cases in which ground agents are caught by the fire and accidents take place.

Airborne agents. The team consists furthermore of a number of airplanes which can move to each of the four directions at high speed and drop a waterbomb. The positions on which water is distributed after dropping a waterbomb depends on the direction and speed of the airplane. The most effective place to drop water is in front of the fire. The effect is that waterbombed cells will have their fire activation decreased. Once the agents' water reservoirs are empty, they have to return to refill which takes some time. These agents are characterized by their flying speed, landing time, taking off time, refilling time, and the size of the waterreservoir. The cost of putting them into operation depends on the total distance traveled and the number of landings.

4. Reinforcement Learning

Planning is one approach to solve the problem. Given a simulated fire a plan can be made using information about the expected time needed by a group of agents to cut a fireline and the expected time the fire takes to arrive at these lines. However, planning in stochastic dynamic environments is very difficult. It may happen that after designing a plan, the environment changes in an unexpected way so that the plan cannot be executed. Therefore, for such environments repeated planning is necessary as new information arrives, which can be computationally very costly.

Instead of planning, we can also learn reactive policies. Such policies map inputs to actions and react immediately to changes in the environment. To search for policies we will use reinforcement learning (RL) algorithms (Watkins 1989, Bertsekas/Tsitsiklis 1996, Kaelbling et al. 1996). RL methods learn from trial and error to predict the total expected cost of particular situations. The policy uses these predictions to select actions which are expected to lead to the least expected future cost.

The decision cycle. Reinforcement learners are online decision makers. Thus, instead of using a prior simulation of the fire and basing the decisions on this simulation, decisions are made as the simulation is running. This makes it possible to immediately react to changes of the environment (including new data of the real fire) without the need for any replanning. The following decisions are made/learned during a forest fire simulation:

1) Fire severity assessment. The fire danger is rated according to the expected difficulty to control the fire and the expected speed of the fire front.
2) Choose resources based on global features and the fire severity rating.
3) Select an initial target for all resources.
4) Execute the agent policies. These policies enable the agents to react to changes in the environment.

Here (1-3) are global decisions for the fire manager which set the stage for the fight, and (4) consists of sequences of decisions of multiple agents (we execute a complete behavior). Note that we do not (yet) allow for adding resources after the fire has started.

Cost function. The performance of the team is evaluated by using a cost function. This cost function assigns costs to burned-trees areas, to burned assets, and to all actions executed by the teams, helicopters and airplanes. By using the cost function, each change (time step) of the environment can be evaluated, and summing over all steps makes a long term evaluation of the team's performance possible. We have to note that we cannot use hillclimbing on the cost function for decision making. In RL the goal is to minimize the *long term* cost. If we would greedily act to

save costs in the next steps, we may let the fire escape so that it causes much larger costs later on.

The input. Given some state of the environment, consisting of the state of all cells and of all agent positions, we construct an agent's input. It is infeasible to give agents complete global information of the state of the environment - there are too many dynamic features, which would make the input space explode and learning terribly slow. Furthermore, most decisions are not sensitive to many of the possible inputs. It is clear that for the different decisions in our decision cycle, we have to use different inputs. For the fire manager decisions (1-3), we need to focus on the fire, the wind speed, and the area in front of the fire front to find out where and how fast the fire is going to spread. For the behavior of an agent during the simulation, we use the state of the region around the agent as input for her decision making. This information contains fire activation levels and the terrain types of the areas around which is very helpful for local navigation. To make learning global goals easier, we will also include inputs which tell how far and in which direction the closest burning gridcell is, how large the fire is, where the fire front is etc.

The policy and evaluation function. An agent's policy maps inputs to actions (decisions). This is done by using evaluation functions which map input/action pairs to future cumulative expected cost. After having evaluated all input/action pairs, the most promising action is selected. Since the input space is very large (the number of input dimensions may be over 50), we cannot use tabular representations in which each input has a special entry. Instead, we have to use a function approximator. Combining function approximators with RL is an active topic of research, and in the course of years many different function approximators have been used, including neural networks (Tesauro 1992, Bertsekas/Tsitsiklis 1996) and CMACs (Albus 1975, Sutton 1996). We will in particular focus on model-based CMACs, which have been used successfully for learning to play soccer strategies (Wiering et al. 1998a). CMACs use multiple filters which evaluate different partial descriptions of the state and then combine these evaluations.

H**ierarchical evaluation functions**. Since the wind speed and wind direction play a crucial role in evaluating actions, we will use them to combine expert evaluation functions (EFs). Different expert EFs are used for specific wind speeds and directions. For combining the EFs for a particular wind situation, we locally interpolate between them. This setup makes coping with changing winds a simple task.

The hierarchical EFs also allow to search for cooperative (team) solutions. The fire manager selects a global attack plan by selecting initial targets for the agents. These subplans could also be used to select a specific evaluation function for each agent. By learning which evaluation functions can be combined in a fruitful way, the fire manager searches for a set of cooperative agent strategies.

Learning the policy. The policy is learned by trial and error: we start a simulation with some initial fire, let the fire manager select the fire fighting team and

let the agents execute their reactive behaviors. We continue the simulation until the fire has been extinguished or some time limit has been reached, and then compute the total cost of the fire.

To learn manager decisions, we simply keep track of the average total cost of simulations in which a particular decision was made. The decision which is expected to lead to the minimal future cost in a particular environmental state is considered as the best decision, although it is necessary to repeatedly make other decisions (explore) to learn which one is optimal.

For learning the agent behaviors, we adapt the evaluation of all situation/action pairs which have occurred during a simulation. To adapt these evaluations, we do not just adapt the evaluations towards the cumulative future cost of the simulation. The cumulative cost has very large variance and when actions only differ slightly, it is very difficult to find out which action is better. Instead, RL methods learn the evaluation by using the immediate cost and the expected cost of the successor state(s). There exist a number of RL methods such as Q-learning (Watkins 1989), temporal difference learning (Sutton 1988, Tesauro 1992) and model-based RL (Moore/Atkeson 1993, Wiering/Schmidhuber 1998b). We will focus on the latter which learns a model of state transitions and estimated costs of these transitions. Given this model, dynamic programming (DP) like algorithms (Bellman 1961, Moore 1993) are used to compute the evaluation function and a new policy. Model-based RL is very effective for problems in which we can store all state transitions (Moore 1993). We can combine model based RL with CMACs by learning multiple models, where each model is used for estimating the dynamics of each partial state description.

Learning team policies. Since the evolution of the environment depends on the complete team behavior, it is difficult to evaluate the behavior of a single agent. It may be possible that some agent's behavior is very good, whereas another agent does not help to solve the problem. In such cases, we may punish the first agent's behavior, since the team solution was bad. Only if the team is successful, the behaviors will be reinforced by the learning algorithm. However, the probability of having multiple good behaviors together decays exponentially with the number of agents. To solve this problem, we will construct some simple behaviors, which provide the learning system with reasonable starting points for learning. After this, the system may learn from the experiences generated by executing these behaviors to shape them (Dorigo/Colombetti 1998).

5. Discussion

We presented a framework for building an intelligent system which can learn to control forest fires, a complex environmental problem which has received increasing interest during the last decades. It is important to include intelligent subsystems for

supporting human decision making for this complex task, since the problem involves many interacting subprocesses (multiple forest fires can happen at the same time) which make long term cost estimation of different plans very difficult for human experts.

Our methodology relies on learning to control a team of forest fire agents by reinforcement learning. Reinforcement learning is a promising method to learn reactive agent behaviors from trial and error. Agents are put in the environment and learn from the experiences generated by their interaction with the environment. RL is usually used for learning single-agent policies and for solving Markov decision processes (Puterman 1994). The current problem features multiple agents which act in parallel, where actions take different amounts of time and where the true environmental state is partially observable. This makes the problem very challenging and finding good methods for coping with these topics would make RL more widely applicable to solve real world problems.

Due to the complexity of the problem, we argue that an incremental approach should be used for developing the fire spread model. The initial model describes the abstract mathematical problem of constructing a set of lines to minimize the size of an expanding stochastic process. It should be noted that this model also captures essential features of related problems such as the spreading of infectious diseases, flooding, fire outbreaks inside cities, and the spreading of violence or panic.

Once we are able to find good team strategies for controlling forest fires in our initial model, we make the model more complex by adding features which make the model more realistic. This will enable us to face the difficulties of realistic forest fires and to search for efficient ways for controlling them.

References

Albus, J.S. (1975): A new approach to manipulator control: The cerebellar model articulation controller (CMAC). Dynamic Systems, Measurement and Control, 97:220-227.

Avesani, P., Perini, A., Ricci, F. (1993): Combining CBR and constraint reasoning in planning forest fire fighting. In Proceedings of the first European workshop on Case-Based reasoning, pages 235-239.

Beer, T. (1990): The Australian national bushfire model project. Mathematical and Computer Modelling, 13, (12):49-56.

Beer, T. (1991): The interaction of wind and fire. Boundary-Layer Meteorology, 54:287-308.

Bellman, R. (1961): Adaptive Control Processes. Princeton University Press.

Bertsekas, D.P., Tsitsiklis, J.N. (1996): Neuro-dynamic Programming. Athena Scientific, Belmont, MA.

Calabri, G. (1982): Recent evolution and prospects for the Mediterranean region. In van Nao, T., editor, Forest fire prevention and control: Proceedings of an international seminar. Martinus Nijhoff and Dr W. Junk.

Crites, R., Barto, A. (1996): Improving elevator performance using reinforcement learning. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, Advances in Neural Information Processing Systems 8, pages 1017-1023, Cambridge MA. MIT Press.

Dorigo, M., Maniezzo, V., Colorni, A. (1996): The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41.

Dorigo, M., Colombetti M. (1998): Robot Shaping: An Experiment in Behavior Engineering. MIT Press/Bradford Books.

Dorigo, M., Gambardella, L.M. (1997): Ant colony system: A cooperative learning approach to the traveling salesman problem. Evolutionary Computation, 1(1):53-66.

Gambardella, L.M., Rizzoli, A., Zaffalon, M. (1998): Simulation and optimization for intermodal terminal ship planning and shipyard layout. Special Issue of Simulation on Harbour and Maritime. To appear.

Glover, F., Laguna, M. (1997): Tabu Search. Kluwer Academic Publishers.

Holland, J.H. (1975): Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Kaelbling, L.P., Littman, M., Moore, A. (1996): Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4:237-285.

Kourtz, P. (1994) Advanced information systems in Canadian Forest Fire Control, AFAC Conference, Australia.

Moore, A., Atkeson, C.G. (1993): Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning, 13:103-130.

Moore, P.F., Trevitt, A.C.F. (1991): Computers in fire management: Limitations of the mechanistic apporach. In Andrews, P.L. and Potts, D.F., editors, Proceedings of the 11th Conference on Fire and Forest Meteorology, pages 98-108. Society of American Foresters.

Puterman, M.L. (1994): Markov Decision Problem - Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc, New York.

Ricci, F., Mam, S., Marti, P., Normand, P., Olmo, P. (1994): CHARADE: a platform for emergencies management systems. Technical Report 9404-07, IRST, Trento.

Rizzoli, A.E., Young, W.J. (1997): Delivering environmental decision support systems: software tools and techniques. Environmental Modelling and Software, 12:237-249.

Rothermel, R.C. (1972): A mathematical model for predicting fire spread in wildland fuels. Technical Report INT-115, USDA Forest Service, Ogden Utah.

Sutton, R.S. (1988): Learning to predict by the methods of temporal differences. Machine Learning, 3:9-44.

Sutton, R.S. (1996): Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E., editors, Advances in Neural Information Processing Systems 8, pages 1038-1045. MIT Press, Cambridge MA.

Tesauro, G. (1992): Practical issues in temporal difference learning. In Lippman, D.S., Moody, J.E., and Touretzky, D.S., editors, Advances in Neural Information Processing Systems 4, pages 259-266. San Mateo, CA: Morgan Kaufmann.

The Petawawa National Forestry Institute (1982): Forest fires in North America. In van Nao, T., editor, Forest fire prevention and control: Proceedings of an international seminar. Martinus Nijhoff and Dr W. Junk.

Watkins, C. J. C. H. (1989): Learning from Delayed Rewards. PhD thesis, King's College, Cambridge, England.

Wiering, M., Salustowicz, R., Schmidhuber, J. (1998a): CMAC models learn to play soccer. In Proceedings of the Eighth International Conference on Artificial Neural Networks. To appear.

Wiering, M., Schmidhuber, J. (1998b): Efficient model-based exploration. In Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior: From Animals to Animats. To appear.