

GENERATING DECISION RULES BY REINFORCEMENT LEARNING FOR A CLASS OF CROP MANAGEMENT PROBLEMS

F. GARCIA, R. MARTIN-CLOUAIRE

Unité de Biométrie et Intelligence Artificielle, INRA, Toulouse, France

E-mail: {fgarcia, rmc}@toulouse.inra.fr

G. L. NGUYEN

Faculty of Information Technology, Hanoi Univ. of Technology, Hanoi, Vietnam

E-mail: giangnl@it-hut.edu.vn

ABSTRACT

This paper addresses the questions of generating near optimal management strategies for a class of crop management problems. Two approaches are presented. They provide strategies under the form of a timely-structured set of decision rules. Both rely on a formalization of the problem as a finite-horizon Markov decision process and use stochastic optimization techniques that belong to the reinforcement learning family of algorithms. The basic idea consists in combining the dynamic programming principle with an iterative evaluation mechanism that exploits a simulator of the crop production system. The two approaches enable to deal efficiently with the large size and continuous nature of the state and decision spaces. The first method uses a CMAC discretization methods of these spaces to represent compactly the worth of state-decision pairs and ultimately extracts the decision rules constituting the strategy. The second method generates directly decision rules having fuzzy (flexible) antecedents.

1. INTRODUCTION

Crop production systems are natural systems that respond to both uncontrollable factors and technical operations that are decided by a human agent at particular times and in relation to the current state of the system. In order to support farmers in making a rational choice of actions at execution time, it is convenient to specify the decision-making behavior over the whole production period through a management strategy elaborated beforehand. The external form of a strategy has to be simple and compact for several practical reasons:

- making them intelligible (for easing their communication);
- enabling their application without the support of any computational device (which is the usual setting in farms).

Constraining strategy to take a simple form may induce some loss of information (and thus sub-optimality in theory) but it is still preferable given the importance of the above requirements. Simplicity is, in a way, a means to avoid over precision in the definition of strategies, acknowledging the imperfection and incompleteness of the knowledge used to build them.

In this paper, the sub-class of problems exemplified by the winter wheat management problem is considered. In this case a strategy can be defined as a temporally structured set of rules of the form *if situation then action* and we address the problem of generating them automatically. We assume that a dynamical model of the biophysical system is given. The model is

implemented in a simulator that is able to respond to time varying climatic factors and to a sequence of technical operations over a given temporal horizon. The problem is to build a management strategy that would enable optimal decisions about these operations over the horizon considered. The optimality of the rule-based strategy is defined in terms of a value function that we want to maximize in expectation in order to take into account the uncertainty in the climatic factors.

Two strategy construction methods are presented here. They rely on a formalization of the decision process as a finite-horizon Markov decision process (MDP). Both use stochastic optimization techniques belonging to the reinforcement learning family of algorithms. The basic idea consists in combining dynamic programming approaches with an iterative evaluation mechanism that exploits the simulator of the biophysical system. Besides reinforcement learning algorithms the first method uses a discretization method of the decision and state spaces to represent compactly the worth of state-decision pairs and ultimately extracts the decision rules constituting the strategy. The second method uses reinforcement learning to generate directly decision rules having fuzzy (flexible) antecedents. The learning technique is combined with a refinement process that progressively modifies the granularity of the rules by splitting those that are not specific enough.

2. INTRODUCTION TO REINFORCEMENT LEARNING

Markov Decision Problems (MDP)

We consider sequential decision problems that can be modeled as Markov Decision Problems (Kennedy,1988) in a finite-horizon setting. Within this representation, the decision process can be divided into a sequence of N decision stages. Each stage i has an associated state space S_i and decision space D_i , and these spaces S_i and D_i are respectively characterized by a set of state and decision variables. A trajectory of this decision process is the result of choosing an initial state s in S_1 and applying a decision d from s to s' in S_2 , and so on until S_N .

Two important characteristics of MDP models concern the Markov property of the uncertain dynamics and the objective function. The Markov property requires that the stochastic transition from s in S_i to s' in S_{i+1} given the decision d in D_i is completely determined by the probability $P_i(s' / s, d)$. Concerning the objective function, we assume that the criterion to be maximized can be represented as the expected value of an additive objective function $V = E(r_1 + \dots + r_N)$ where the r_i terms are local rewards associated to each transition (s, d, s') from S_i to S_{i+1} along the trajectory. Of course, this criterion may depend only on the final state (for instance the yield). Furthermore, it can be the result of a number of calculations (veto, weighted sum, etc.) and thus can aggregate different performance measures.

A policy is a function that maps states to decisions. For finite-horizon problems, such a policy Π can be represented as a set of sub-policies $\{\Pi_1, \dots, \Pi_N\}$. Each sub-policy Π_i is defined as a function which maps state s in S_i to decision d in D_i . Hence, given a policy and an initial state s in S_1 we are able to determine step after step until harvest, what are the decisions to apply to the crop, depending on the current state that depends itself on the initial state, on previous decisions and on the uncertain weather. Considering a MDP and an objective function, the question is then to define and generate an optimal policy Π that, for any initial state s in S_1 maximizes $V_{\Pi} = E(r_1 + \dots + r_N / \Pi)$.

Winter wheat management as a MDP

To give a concrete example of agricultural MDP let us consider the case of a winter wheat crop production on a field (Garcia, 1999). Within the MDP representation, winter wheat crop management can be divided into a sequence of $N = 4$ decision steps: sowing, first and second nitrogen supply and harvest. The corresponding state and decision spaces are defined by the state and decision variables presented in Table 1. The state variables for the 2 nitrogen supplies are retained for their capacity to summarize the past trajectory of the process at the current steps, and thus to approach the Markov property as close as possible.

	<i>Sowing</i>	<i>1st N application</i>	<i>2nd N application</i>	<i>Harvest</i>
State Variables	- sowing time dS	- tillering dT - nb of plants NP	- residual N in soil Ns - start of stem elongation $d1cm$ - aerial biomass $ba1cm$	- post harvest N in soil PHN - yield
Decision Variables	- seed rate qS - wheat cultivar vS	- date $dN1$ - quantity $qN1$	- date $dN2$ - quantity $qN2$	

TABLE 1: State spaces and decision spaces of the MDP model.

It is assumed that the farmer's goal is to maximize yield, minimize production costs (inputs) and to keep nitrogen leftover after harvest (PHN) below a given threshold value. The costs are represented by 3 variables taking negative values: r_{Sowing} , r_{N1} and r_{N2} which are functions of qS , $qN1$ and $qN2$ respectively. The yield and environmental constraints are combined into a gain $R_{Harvest}$ that is equal to a function of the yield when the value of the post-harvest-nitrogen-in-soil variable PHN is less than 30kg/ha and is equal to 0 otherwise. In this MDP, the criterion to maximize is given by $V_{\Pi} = E(r_{Sowing} + r_{N1} + r_{N2} + R_{Harvest} / \Pi)$.

Finite-horizon Dynamic Programming and reinforcement Learning

The automatic generation of optimal strategies maximizing V_{Π} could theoretically be done by using a finite-horizon dynamic programming algorithm. This algorithm is based on the classical Bellman's optimality equations on value functions V_i mapping S_i to \mathbb{R} .

$$\forall i < N, \forall s \in S_i \quad V_i(s) = \max_{d \in D_i} \sum_{s'} P(s' | s, d) (r_i(s, d, s') + V_{i+1}(s')) \quad (1)$$

where V_N is the terminal value function defined on S_N . From the optimal value function $V_{\Pi} = \{V_1, ..V_N\}$ solution of (1), the optimal policy $\Pi = \{\Pi_1, .., \Pi_N\}$ is then determined by

$$\forall i < N, \forall s \in S_i \quad \Pi_i(s) = \arg \max_{d \in D_i} \sum_{s'} P(s' | s, d) (r_i(s, d, s') + V_{i+1}(s')) \quad (2)$$

Unfortunately, this algorithm cannot be applied directly in some problems where:

- most of the state and decision variables have continuous domains, preventing the use of a discrete representation of the V_i value functions;
- the transition probabilities of the stochastic biophysical process are not known and difficult to estimate.

The reinforcement learning approach overcomes these two difficulties. It consists in learning optimal policies by modifying iteratively a value function on the basis of a simulation-based evaluation of the policy determined by the current value function. Today, reinforcement learning is one of the major approaches to solve Markov decision problems with unknown transition probabilities and/or with large state variable domains (Sutton and Barto, 1998).

The most studied reinforcement learning algorithm is Q-learning, that is a direct adaptive method since it does not require an explicit model of the Markovian process. The principle of Q-learning in finite-horizon is to learn for each stage i an estimate of the Q-value

$$Q_i(s, d) = E\left(\sum_{j=i}^N r_j \mid s_i = s, d_i = d\right), \quad (3)$$

assuming that for the subsequent stages $j > i$ an optimal policy is followed. Once these estimates have been learned, the optimal policy can be obtained through (4):

$$\forall i, \forall s \in S_i \quad \Pi_i(s) = \arg \max_{d \in D_i} Q_i(s, d). \quad (4)$$

In the finite-horizon setting, the estimates Q_i are regularly updated after each simulated trajectory. The trajectories are obtained by picking randomly an initial state in S_i and then choosing at each stage i either the current optimal decision according to (4) or a random decision in S_i . In the case of discrete spaces S_i and D_i , the update rule for Q_i is:

$$Q_i(s_i, d_i) \leftarrow Q_i(s_i, d_i) + \varepsilon \cdot (r_i + \max_{d'} Q_{i+1}(s_{i+1}, d') - Q_i(s_i, d_i)). \quad (5)$$

In this rule, (s_i, d_i, s_{i+1}) is the observed transition from stage i to stage $i+1$, and r_i the associated reward. The learning rate ε decays toward 0. The factor on the right hand side of ε is the update factor, also called the temporal difference error, and denoted ΔQ_i .

When the S_i and D_i domains are continuous or are very large, the approach cannot be applied directly. Nevertheless the general Q-learning mechanism can be exploited in adapted approaches that use compact representation of Q_i and Π_i and specific update rules based on this mechanism. Two such approaches are presented in the next section.

3. TWO APPROACHES FOR GENERATING RULES

CMAC Representation and Automatic Rule Extraction

Linear architectures consist in representing the Q_i value functions as a linear combination of a small number of features ϕ_i^k that describe states and decisions:

$$Q_i(s, d) = \sum_{k=1}^p \omega_i^k \phi_i^k(s, d). \quad (6)$$

The CMAC representation (Sutton and Barto, 1998) is a simple linear architecture with binary valued features. It is defined by uniform partitions of the state and decision domains which are shifted with respect to each other, in order to define as many binary valued features as there are cells in the partitions. This representation is classically used for estimating functions in a continuous space. In that case the adaptation of the Q-learning algorithm is direct, the $Q_i(s, d)$ terms being replaced by the weights ω_i^k and the error terms being multiplied by the gradient terms $\phi_i^k(s_i, d_i)$.

Parameterized representations like CMAC are not suitable with respect to intelligibility and ease of applicability requirements. To remedy this we have developed an efficient procedure to automatically extract simple decision rules from the CMAC Q-value functions.

This method can be decomposed in two steps. First, we use the RPART routines (Therneau and Atkinson, 1997) for building binary regression trees T_i from the Q_i CMAC value functions mapping $S_i \times D_i$ to \mathbb{R} . Each tree is classically built as follows: a state or decision

variable that best splits the initial node $S_i \times D_i$ into two groups is found, and then this process is recursively applied to each sub-node, until a minimum size is achieved or until no improvement can be made. The fitted value of a node is its mean value, and the error of a node is determined as its variance. Second, we extract from these T_i trees new binary decision trees $MAXT_i$ defined by $MAXT_i(s) = \operatorname{argmax}_d T_i(s,d)$. Since the outputs of T_i are the same for all points that belong to the same node, the outputs of the binary trees $MAXT_i$ are not specific values but rather intervals for each variable of D_i . These binary trees $MAXT_i$ thus lead directly to policies represented as a set of exclusive decision rules like:

$$\text{Rule: if } s \in S \text{ then } d \in D \quad (7)$$

where S is a subset of S_i defined as a product of intervals for the different state variables of S_i , and where D is similarly defined as the product of intervals for the decision variables of D_i . The interpretation of such a rule is the following: when $s \in S$ the rule is active and any decision d in D can be chosen with approximately the same optimal effect. A typical rule might be *if $dT \in [Nov1, Jan1]$ and $NP \in [140,170]$ then $dN1 \in [dT, dT+7]$ and $qN1 \in [20,40]$.*

Learning fuzzy rules

The other method presented here aims at directly learning strategies represented, for each decision stage, by a set of fuzzy rules. Each rule has the form:

$$\text{Rule}_r: \text{if } s \text{ is in } C_r \text{ then } d \text{ should be } d_r \quad (8)$$

where C_r is a fuzzy set, that is a subset of the state space in which some elements have only partial membership, and d_r is a value in the decision space. Such use of a fuzzy set is a convenient way to express a flexible restriction on the set of states for which d_r is an acceptable decision. At any stage, applying the strategy means inferring the decision d^s defined by:

$$d^s = \alpha^{-1} \cdot \sum_{r=1}^p \alpha_r(s) \cdot d_r \quad (9)$$

where p is the number of rules for the current stage, $\alpha = \sum_{r=1}^p \alpha_r(s)$, α_r is the membership function of C_r and $\alpha_r(s)$ is the degree of compatibility between the condition C_r and the state s (which may be multidimensional). A typical fuzzy rule might be:

if dT is around Dec1 and Np is around 180 then $dN1$ should be Dec15 and $qN1$ should be 15.

The learning process operates on a set of rules that are slightly different. They have multiple consequents that, for any rule r , have the form: *d should be d_r^j with pertinence $q[r,j]$* . The degree $q[r,j]$ represents the current estimate of the worth of choosing d_r^j when the state s matches the condition part of the rule r . These degrees are learned by using the basic iterative mechanism at work in Q-learning. For each simulated trajectory, $q[r,j]$ is updated by adding an update factor $\Delta q[r,j]$ defined as follows (Jouffe, 1998):

$$\Delta q[r,j] = \varepsilon \cdot \Delta Q_i \cdot \alpha_r(s) / \sum_{m=1}^p \alpha_m(s) \quad (10)$$

where ε and ΔQ_i are the rate and the temporal difference error involved in formula (5):

$$\Delta Q_i = r_i + \max_{d'} Q_{i+1}(s_{i+1}, d') - Q_i(s_i, d_i).$$

The simulated decision d_i at stage i is derived by interpolation as in (9), by choosing for each rule r a consequent d_r^j . The term $Q_i(s_i, d_i)$ is computed similarly as the weighted sum over the p rules of the pertinence degrees of the selected consequents:

$$Q_i(s_i, d_i) = \alpha^{-1} \cdot \sum_{r=1}^p \alpha_r(s_i) \cdot q[r, j] \quad (11)$$

where $\alpha = \sum_{r=1, p} \alpha_r(s_i)$. The term $\max_{d'} Q_{i+1}(s_{i+1}, d')$ is computed as in (11) but for s_{i+1} in place of s_i and $\max_j q[r, j]$ in place of $q[r, j]$.

For any stage the conditions of the rules constitute a fuzzy partition of the state space. Initially the partition is coarse and the learning process is equipped with a mechanism that:

- identify periodically the region in which the set of recent ΔQ_i values is most scattered;
- modify the partition so as to have a better covering in the region identified;
- modify existing rule, create new ones and initialize them.

The learning process stops when no more regions need to be refined according to a numerical criterion. Ultimately, in each rule only the most pertinent consequent is kept, thus the learning process returns rules of the form (8).

4. CONCLUDING REMARKS

We have presented in this paper two reinforcement learning methods to derive rule-based optimal strategies of crop management by using a simulator of the crop production process. Empirical studies conducted in the winter wheat production case have shown that the methods are reliable and can generate satisfactory crop management decision rules. See (Garcia, 1999) for a detailed report of experimental results with the first method. The second method which is of more recent conception need to be further explored and compared with the first one.

The methods that have been presented are appropriate only if simple decision rules are sufficient to express management strategies as in the winter wheat problem. This may likely not be the case with other management problems. More hierarchical representations that distinguish between overall planning of activities and operational realization of these may be needed. Investigation in this direction is the subject of future work in our research team.

REFERENCES

- Attonaty, J.-M., Chatelin, M.-H., Garcia, F. and Ndiaye, S. (1997) Using extended machine learning and simulation technics to design crop management strategies, Proc. of EFITA97, Copenhagen, DK.
- Garcia, F. (1999) Use of reinforcement learning and simulation to optimize wheat crop technical management, Proc. of MODSIM99, Hamilton, NZ.
- Jouffe, L. (1998) Fuzzy Inference System Learning by Reinforcement Methods, IEEE Transactions on Systems, Man, and Cybernetics, part C, vol. 28, n° 3, p 338-355.
- Kennedy, J.O. (1986) Dynamic Programming: application to agricultural and natural resources, Elsevier Applied Science, London.
- Racsko, P., Szeidl, L., Semenov, M. (1991) A serial approach to local stochastic weather models. Ecological Modelling, 57.
- Sutton, R.S., Barto, A.G. (1998) Reinforcement Learning: an introduction, MIT Press, Cambridge.
- Therneau, T.M., Atkinson, E.J. (1997) An introduction to recursive partitioning using the RPART routines, technical report, Mayo Foundation.