Use of Reinforcement Learning and Simulation to Optimize Wheat Crop Technical Management

Frédérick Garcia
Unité de Biométrie et Intelligence Artificielle
INRA, BP27, Auzeville,
31326 Castanet Tolosan cx,
France
fgarcia@toulouse.inra.fr, www-bia.inra.fr/T/garcia

Abstract Considering the example of winter wheat crop production in France, a Markov decision problem (MDP) model was built to characterize optimal state dependent technical strategies for crop management. Stochastic dynamic programming algorithms could theoretically solve the problem of automatically generating optimal strategies. Unfortunately, the important size of state and decision spaces precludes this option. In this paper we present an alternative methodology called reinforcement learning. Reinforcement learning is one of the major approaches to solve Markov decision problems with unknown transition probabilities. R-learning, one of the most studied reinforcement learning algorithms, maintains estimates of the average reward ρ and of the relative value functions $R(s,d)$ of choosing decision d in state s, from which an optimal strategy can be derived. Application of R-learning to the crop management problem leads to a regular updating of the ρ and R parameters after each trial of the crop simulation, for different one-year weather series. The XITEK software implements this algorithm on the basis of a winter wheat crop management simulation tool called DÉCIBLÉ and a stochastic weather simulation model. Applied to the problem of maximizing yield under a strong constraint on the soil nitrogen pollution, interesting new strategies have been obtained and compared with more classical crop management decision rules defined by agronomy and farming system experts.

1. INTRODUCTION

Considering the new economical context and the growing environmental preoccupation in European agriculture, it is a common opinion that some new production models have to be defined. For the specific question of technical crop management like wheat, maize, etc., simulation tools based on crop models have been considered in increasing numbers over the last two decades, and proved themselves to be very efficient for estimating the economic and environmental effects of particular technical decisions concerning the selection of the variety, seed rate, sowing date, fertilization, etc., hence helping in the manual design of new crop management strategies (Attonaty et al. [1999]).

The main objective of the work presented in this article is to apply powerful optimization algorithms to this simulation context, in order to support agronomist engineers and researchers in automatically designing technical crop management strategies. The new methodology we develop and apply to this problem is called reinforcement learning. It is a kind of stochastic dynamic programming approach based on simulation, very popular in the field of machine learning and artificial intelligence, that has been successfully applied to complex domains like autonomous robotics, manufacturing production, etc. The present paper is an introduction to this reinforcement learning methodology we applied to the DÉCIBLÉ simulation tool for winter wheat crop management, that has resulted in a software called XITEK.

2. DÉCIBLÉ: A WHEAT CROP MANAGEMENT SIMULATOR

The DÉCIBLÉ software has been developed by INRA (the French national institute on agronomic researches) and ITCF (the industry federation for cereals and fodder crops) as a decision support model for designing winter wheat crop management operational strategies by means of simulation. It is built on 3 main models that day after day simulate the crop, weather and decision processes.

The weather model specifies for each day the temperature, the solar radiation, the potential evapotranspiration, etc. These data either correspond to some historical climatic scenario, or are randomly generated using the weather generator proposed by Racsko et al. [1994].

The crop model consists of a set of empirical modules that simulate the plant development and yield build-up. It can model the agronomic consequences of a large range of crop husbandry

operations. Inputs of this crop model are the weather data, some soil variables defining the soil context, like texture, density, and cropping history, and precise husbandry operations produced by the decision model (Meynard [1997]).

The decision model determines the daily operations from the observation of some indicators concerning climate, crop state, previous operations, etc. The input of this decision model is a crop management strategy expressed as a set of formal decision rules IF THEN ELSE.

3. THE OPTIMIZATION PROBLEM

The problem we have chosen to tackle in order to illustrate the reinforcement learning approach applied to the DÉCIBLÉ crop management simulation tool consists in finding strategies that satisfy two contradictory goals such as:

- to maintain under the prescribed limit the nitrogen content of drinking water going out of the parcel;
- to obtain a satisfying production level.

That problem, defined with the help of agronomy and farming system experts, has been chosen since no a priori corresponding optimal strategies are known, making well-justified the use of a simulation tool like DÉCIBLÉ to explore different potential solutions.

3.1 Introduction to Markov Decision Processes

In order to implement a reinforcement learning algorithm for solving this optimization problem, the generality of the formal language used in DÉCIBLÉ for expressing crop management strategies has been circumscribed and the decision problem we consider has been modeled as a Markov Decision Problem (see Kennedy [1988], Puterman [1994]).
Within the MDP representation, winter wheat crop management can be divided into a sequence of N decision steps concerning sowing, fertilizing, etc. until harvest. Each step 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 randomly 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 concerns 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 $P_i(s' \mid s, d)$. Concerning the objective function, we assume that the criterion to be maximized can be represented as the expected value of an additive objective function $V = E(r_1+...+r_N)$ 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, that criterion can also only depend on the final state (for instance the yield). Furthermore, this criterion can be the result of a number of calculations (veto, weighted sum, etc.) and thus can aggregate different performances.

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 $\{\Pi_1,...,\Pi_N\}$. 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 Markov decision process and an objective function, the question is then to define and generate an optimal policy Π that maximizes $V_\Pi = E(r_1+...+r_N \mid \Pi)$

3.2 Modeling the Optimization Problem

For the optimization problem considered in this article, we have retained $N=3$ decision steps, respectively sowing, first and second nitrogen supply. 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. The state variable of the sowing step dS is introduced as an external constraint to the problem, since this sowing time is modeled as a random variable. Decision variables have been fixed after agronomists confirmed that they were the main influent variables modeled in DÉCIBLÉ for the considered objective.

The variable domains of the table correspond to the optimization cases we will present in section 5. One can note the relative definition of the $dN1$ and $dN2$ domains, meaning that a policy does not specify an absolute date, but rather a relative date before or after the appearance of tillering or stem elongation.

	Sowing	1st nitrogen application	2nd nitrogen application
State Variables	- sowing time dS $\in$ [01/10,15/12]	- tillering dT $\in$ [15/11,01/04] - number of plants NP $\in$ [0,200]	- residual soil nitrogen $Ns \in$ [0,100]kg/ha - start of stem elongation $d1cm$ $\in$ [15/02,15/05] - aerial biomass $ba1cm \in$ [0,200]g/m^2
Decision Variables	- seed rate qS $\in$ [100,200]g/m^2 - wheat variety vS $\in$ {soissons, artaban, ..}	- date $dN1$ $\in$ [dT-5,dT+20] - quantity $qN1$ $\in$ [0,100]kg/ha	- date $dN2$ $\in$ [$d1cm$-5,$d1cm$+20] - quantity $qN2$ $\in$ [0,200]kg/ha

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

In order to model the optimization problem defined at the beginning of this section, our choice has been to consider the following objective function $V_{\Pi} = E(Y \mid \Pi)$ where the variable Y stands for the annual yield when the value of the post-harvest-nitrogen-in-soil variable PHN is less than 30kg/ha, and is equal to 0 when PHN is greater than 30kg/ha:

$$Y = \begin{cases} yield \text{ if } PHN < 30\text{kg/ha}; & (1) \\ 0 \text{ elsewhere.} \end{cases}$$

This sharp criteria has been intentionally chosen in order *(i)* to correctly model the 30kg/ha limit as a strong constraint on the crop management, and *(ii)* to facilitate the comparison with crop management strategies hand-coded by experts.

4. A REINFORCEMENT LEARNING APPROACH

4.1 A Dynamic Programming Algorithm

The problem of automatically generating optimal strategies maximizing V_{Π} could theoretically be solved by using a finite-horizon dynamic programming algorithm (Putterman [1994]). This algorithm is based on the classic Bellman's optimality equations on value functions V_i mapping S_i to IR.

$$\forall i < N, \forall s \in S_i$$
$$V_i(s) = \max_{d \in D_i} \sum_{s'} P(s' \mid s,d)\left(r_i(s,d,s') + V_{i+1}(s')\right) \quad (2)$$

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 (2), the optimal policy $\Pi = \{\Pi_1,...,\Pi_N\}$ is then determined by

$$\forall i < N, \forall s \in S_i \quad (3)$$
$$\Pi_i(s) = \arg\max_{d \in D_i} \sum_{s'} P(s' \mid s,d)\left(r_i(s,d,s') + V_{i+1}(s')\right)$$

Unfortunately, two characteristics of the problem prevent us from directly applying this algorithm:

- Most of the state and decision variables have continuous domains, forbidding the use of a discrete representation of the V_i value functions.
- we do not have a formal markovian model of the stochastic crop growth process with its transition probabilities, and we can only use the DÉCIBLÉ simulation tool.

4.2 The Reinforcement Learning Approach

The reinforcement learning approach consists in learning optimal policies by repeatedly modifying a value function on the basis of a repeated experimental evaluation of the policy determined by the current value function (Sutton et al. [1998]). Today reinforcement learning is one the major approach to solve Markov decision problems with unknown transition probabilities and/or with large state variable domains (Bertsekas et al. [1996]).

The most studied reinforcement learning algorithms like Q-learning and R-learning (see Kaelbling [1996]) were developed for stationary infinite-horizon Markov decision processes. In order to apply these algorithms to the crop management problem we consider, we proposed an adaptation of Q-learning and R-learning to the case of non-stationary finite-horizon MDPs (Garcia et al [1998]). In this paper we only present the adapted R-learning approach, since that was the method that led to the best experimental results.

R-learning, like Q-learning, is often called a direct adaptive method since it does not rely on an explicit model of the markovian process. The principle of R-learning in finite-horizon is to learn estimates of the average value

$$\rho = E\left(\frac{1}{N}\sum_{i=1}^{N} r_i\right) \quad (4)$$

and of the relative value functions

$$R_i(s,d) = E\Big(\sum_{j=i}^{N} r_j - \rho \mid s_i = s,\ a_i = d\Big) \qquad (5)$$

for an optimal policy. When these estimates have been learned, the optimal policy can be obtained through (6):

$$\forall i, \forall s \in S_i\ \Pi_i(s) = \arg\max_{d \in D_i} R_i(s,d) \quad (6)$$

In the case of a finite-stage process like the crop management problem, these estimates R_i and ρ are regularly updated after each trial of the crop simulation. At each iteration n the decisions taken at each stage are determined either from the current optimal policy (*greedy-policy*) calculated from the current values R_i^n with the same formulae than (6), or are chosen randomly in order to explore the domain.

If we assume for the moment a discrete state and action representation, the R-learning algorithm we implemented is the following, where α_n and β_n are small learning rates decaying over time:

Initialize ρ and R_i to 0
For n = 1 to n_{max} Do:
- Choose s_1 in S_1;
- Generate with DÉCIBLÉ a one-year random trajectory from s_1, following the greedy or a random policy;
- For i = 3 to 1 Do:

$$R_i(s_i,d_i) \leftarrow R_i(s_i,d_i) + \alpha_n (r_i - \rho + \max_{d'} R_{i+1}(s_{i+1},d') - R_i(s_i,d_i));$$

If d_i *is greedy,*

$$\rho \leftarrow \rho + \beta_n (r_i - \rho + \max_{d'} R_{i+1}(s_{i+1},d') - R_i(s_i,d_i)),$$

where $r_1 = r_2 = 0$, $r_3 = Y^n$ and $R_4^n() = 0$

Return ρ and R_i.

Figure 2 : R-learning algorithm : the tabular case.

4.3 Parameterized Value Functions

One of the main interests of reinforcement learning algorithms like R-learning stems on their capacity to be adapted to continuous or large-scale state and decision variable domains. When the tabular representation is not possible, the value functions R_i that determine the optimal policy Π_i at each decision step can be parameterized in many different ways from neural networks to weighted decision rules, for which reinforcement learning update rules still can be defined. Among these representations, the linear architectures are, with no doubt, the simplest ones and though are often very efficient. They consist in modeling the value functions R_i as a linear combination of a small number of features ϕ^k_i that describe states and decisions:

$$R_i(s,d) = \sum_{k=1}^{p} \omega^k_i\, \phi^k_i(s,d) \qquad (7)$$

In that case the adaptation of the R-learning algorithm is direct, the $R_i(s,d)$ terms being replaced by the weights ω^k_i and the error terms being multiplied by the gradient terms $\phi^k_i(s_i, d_i)$. In our application these features have been chosen following a method called CMAC (see Sutton [1998]). The CMAC representation consists of a number of 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 that there are cells in the partitions. This representation is classically used for estimating functions in a continuous space, particularly in reinforcement learning.

4.4 Automatic Rule Extraction

Parameterized representations like CMAC do not allow neither a practical comprehension of the generated policies nor their direct exploitation for crop management by farmers or agronomists. Thus our next objective has been to automatically extract simple decision rules from these CMAC policies.

The first method we have implemented was proposed by Craven [1996]. Binary regression trees mapping S_i to D_i are directly built from the optimal policies Π_i that have been learned. We have used the C code of the Splus *rpart* package of Therneau et al. [1997]. The tree is classically built as follows: a first state variable is found that best splits the initial node S_i into two groups, 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 $\overline{\Pi_i}$, and the error of a node is determined as its variance $\sum_s \left\| \overline{\Pi_i} - \Pi_i(s) \right\|^2$.

The regression tree that are built through that method are very dependent on the norm on D_i that is considered in the node error, and on the Π_i(s) values. As these values have been obtained by minimization of the $R_i(s,.)$ functions, they are most of the time altered by some more or less important noise resulting from learning. Applied to our optimization problem, that method resulted, as we expected, unsatisfying decision rules when the number of rules was constrained to be small.

In order to overcome that problem, the second approach we have developed consists in reversing the maximization process by (i) building binary regression trees T_i from the R_i CMAC value functions mapping $S_i \times D_i$ to IR and (ii) extracting from these T_i trees new binary decision trees $MAXT_i$. The main interest of the first step is that now the whole information included in the learned functions R_i is exploited within the regression process, that does not depend anymore on a specific norm on D_i : the node error is classically defined as $\sum_{s,d} \left(\overline{R_i} - R_i(s,d)\right)^2$.

An original algorithm has been developed to compute efficiently $MAXT_i$ from T_i (Garcia [1999]). The important point to note is that the outputs of the binary trees $MAXT_i$ are not specific values but rather intervals for each variable of D_i. These binary trees $MAXT_i$ lead directly to policies represented as a set of exclusive decision rules like:

$$IF\ sx \in [sx_{min};\ sx_{max}] \wedge sy \in [sy_{min};\ sy_{max}] \wedge \ldots$$
$$THEN\ dx \in [dx_{min};\ dx_{max}] \wedge dy \in \ldots \qquad (8)$$

where *sx, sy, ...* and *dx, dy, ...* are respectively the state and decision variables of one of the *N* stages. The interpretation of this kind of rule is the following: when the previous rule is active, any decision *dx, dy, ...* in the intervals $[dx_{min};\ dx_{max}]$, $[dy_{min};\ dy_{max}]$, ... can be chosen with approximately the same optimal effect on the final objective *Y*.

5. EXPERIMENTAL RESULTS

5.1 Four Considered Cases

The XITEK software is an C implementation of the previous algorithms. It was tested on four different problem configurations, and the generated policies were compared with four crop management strategies represented with DÉCIBLÉ decision rules, designed by agronomy and farming systems experts. In this four cases, the decisions of the first sowing stage were completely predefined in order to simplify the result analysis and the comparison with experts.

In agreement with the experts, the four cases were selected for the soil and climate context of the Paris Basin. These cases were defined by considering two distinct winter wheat varieties (the decision variable *vS*), SOISSONS and ARTABAN, and two field situations with different soil nitrogen reserves available to the crop at the end of winter, corresponding to "POOR soil" (N_{soil} = 50 kg/ha) or "RICH soils" (N_{soil} = 150kg/ha). The seed rate *qS* was fixed to 200 kg/ha.

5.2 Learning Parameters

The CMAC value functions for the first and second nitrogen supplies were defined using 5 shifted grids of 4x7x3x3 cells for R_2 and 5x5x5x3x5 cells for R_3. Thus the 2 functions R_2 and R_3 were completely specified given the values of 252+1875=2127 ω^k parameters.

The learning parameters were fixed to

$$\alpha_n(s,a) = \frac{0.05}{1+\log N(s,a)}, \quad \beta_n = \frac{0.01}{1+\log n} \qquad (9)$$

where *N(s,a)* is the number of time where pair *(s,a)* was visited at time *n*.
The greedy policy was chosen in 85% of the trajectories, The initial values of the R_i parameters and of ρ were fixed randomly in [0,1]. After different trials, we retained a learning length of 800,000 one-year trajectories (about 10 hours of computation time on a Linux Pentium PC).
The $MAXT_i$ trees were built so that they define less than 10 decision rules at each stage.

5.3 Analysis of the Learned Strategies

In order to evaluate the learned policies, we simulated them on 5,000 random trajectories. For the binary decision trees, we always chose the mid-action between $[dx_{min};\ dx_{max}]$ for each *dx* decision variable.
The expected value of the output variable *Y* and the percentage of simulated years where *PHN* was over 30kg/ha are presented on tables 2 and 3. As we can see all the four cases were satisfactorily solved concerning the *PHN* constraint.

E(Y) ; %PHN>30kg	POOR soil	RICH soil
ARTABAN	71.75 ; 1.3%	78.40 ; 5.4%
SOISSONS	65,05 ; 1,2%	91,20 ; 0,7%

Table 2: CMAC policy evaluation.

E(Y) ; %PHN>30kg	POOR soil	RICH soil
ARTABAN	65.29 ; 0,1%	80.45 ; 3,8%
SOISSONS	66,27 ; 1,2%	89,84 ; 0,2%

Table 3: Decision tree policy evaluation.

In some cases, *Y* or *PHN* results are better with the decision tree policies than with the original CMAC policies. That can be explained by the fact that regression trees are a way of smoothing noisy data, and thus can improve CMAC policies by extracting continuous trends in the R_i value functions.

The performance of these policies concerning the *Y* final objective were established by asking to the same experts to design manually DÉCIBLÉ decision rules that optimize the yield output under the *PHN* constraint. These experts made during several days an intensive use of the DÉCIBLÉ simulator and this manual optimization led to some simple DÉCIBLÉ decision rules the performances of which are shown on table 4.

E(Y) ; %PHN>30kg	POOR soil	RICH soil
ARTABAN	74.47 ; 0%	86.93 ; 6,45%
SOISSONS	68,35 ; 0,4%	89,69 ; 6,9%

Table 4: Expert decision rules evaluation.

As one can note, the expected yields *Y* obtained with the CMAC and decision tree policies are very close to the results of the expert strategies. Except for the SOISSONS-RICH case, these expert strategies seem to be better than the learned strategies. That is not so surprising since (i) the learning process was not necessary conducted until its complete convergence ; (ii) experts knew very well the behavior of the DÉCIBLÉ model, and (iii) they had access to the CMAC policies learned with XITEK. It is interesting to see that nevertheless, they didn't manage to design a *PHN* satisfactory strategy with a high expected annual yield for the SOISSONS-RICH case, but that XITEK did it.

6. CONCLUSIONS

We have presented in this paper an application of the reinforcement learning methodology to the problem of optimizing by simulation sequential decision processes for crop management in agricultural production systems. Experimental results on 4 cases specified by agronomy and farming system scientists have shown that the approach is reliable, generating satisfactory crop management decision rules.

Nevertheless, the XITEK software didn't manage to generate a strategy that performs really better than the decision rules designed by experts. We explain that by the small number of decision steps of the problem, and we are currently implementing the same reinforcement learning approach on a more complex sequential decision problem concerning irrigation scheduling of maize crops (Bergez et al [1999]).

7. REFERENCES

Attonaty, J.-M., Chatelin, M.-H., and Garcia, F., Interactive simulation modelling in farm decision making, *J. of Computers and Electronics in Agriculture*, accepted for publication, 1999.

Bergez, J.-E. and Garcia, F., Methods for improving decision rules: application to irrigation scheduling of maize crops, Int. Symp. On Modelling Cropping Systems, Lleida, 1999.

Bertsekas, D.P. and Tsitsiklis, J.N., *Neuro-Dynamic Programming*, Athena Scientific, Belmont MA, 1996.

Craven, M.W., Extracting comprehensible models from trained neural networks, Ph.D. thesis, C.S. Dpt, University of Wisconsin, Madison, 1996.

Garcia, F., A learning rate analysis of reinforcement learning algorithms in finite horizon, *proc. of Int. Conf. on Machine Learning*, 1993.

Garcia, F., Extracting decision trees from approximated value functions, *technical report*, INRA BIA, Toulouse, 1999.

Kaelbling, L.P., Littman, M.L. and Moore, A.W. Reinforcement Learning: a survey, *Journal of Artificial Intelligence Research*, 4, 237-285, 1996.

Kennedy, J.O., *Dynamic Programming: application to agricultural and natural resources*, Elsevier Applied Science, London, 1986.

Meynard, J.M., Which crop models for decision support in crop management ? Example of the DECIBLE system, *Proc. of the INRA-KCW workshop on DSS*, Laon, 1997.

Puterman, M.L., *Markov Decision Processes*, Wiley, New-York, 1994.

Racsko, P., Szeidl, L. and Semenov, M., A serial approach to local stochastic weather models. *Ecological Modelling*, 57, 1991.

Sutton, R.S. and Barto, A.G., *Reinforcement Learning: an introduction*, MIT Press, Cambridge, 1998.

Therneau, T.M. and Atkinson, E.J., An introduction to recursive partitioning using the RPART routines, *technical report*, Mayo Foundation, 1997.