

USING EXTENDED MACHINE LEARNING AND SIMULATION TECHNIQS TO DESIGN CROP MANAGEMENT STRATEGIES

**J.-M. Attonaty¹, M.-H. Chatelin¹,
F. Garcia², S. Ndiaye²**

*¹ Station d'Economie et Sociologie Rurales
INRA 78850 Grignon, France*

{attonaty,chatelin}@grignon.inra.fr

*² Station de Biométrie et d'Intelligence Artificielle
INRA 31326 Castanet Tolosan, BP27 Auzeville, France
{fgarcia,ndiaye}@toulouse.inra.fr, www-bia.inra.fr*

Abstract: An analysis of the different applications of a simulation tool for winter wheat crop management previously developed has outlined the difficulties encountered by crop production plan engineers in designing strategies that diverge from those currently used. We present in this paper an Exploration Support System we are currently developing that is based on reinforcement learning and genetic algorithm technics. It defines interesting strategies in an incremental manner, and appears to be a promising approach to assist users to explore a wider range of solutions.

Keywords: Crop Mangement Strategies, Machine Learning, Simulation Techniques

1. Introduction

Today's agriculture has developed in a different context than that proposed by the CAP reform. The currently used crop production models reflect an out-of-date unique criterion of maximum production. Therefore, the redefinition of new options for crop management is crucial. Classical experimental methods can be used but these are very expensive and time consuming. For these reasons, in the context of the winter wheat crop production, we have developed for the last years a simulation tool (DECIBLE) that allows virtual experimentations of new crop management strategies and their implementation in practical situations [ATT96]. An important part of this project has consisted in identifying and representing crop productior strategies, and the formalism we have retained is based on decision rules to be applied by the farmer during the vegetation period considering e.g. the sowing, nitrogen supplies etc... This simulation tool is based on two main components:

- a crop model for the elaboration of the yield strongly related with the effects of technics and environmental conditions (soil, climate) on various criteria (quantity and quality of the yield, risk of diseases and lodging, wastage of inputs etc.).
- a decision model representing the management rules using a specific descriptive language suitable for this type of decision problems. The syntax is based on IF-THEN rules formalism linking the state indicators of the system (crop growth, climate, soil etc.) and the executable actions. In fact, this model plays the role of a pilot by simulating the technical decisions during the crop production period.

The use of this system by crop production plan engineers (researchers and advisers) have made clear that it is not so trivial for them to explore quite different solutions than the actual ones, used in practice. To help users to explore a wide range of solutions, and thus to enable them to select

some different strategies more suitable for the new encountered production contexts, one needs a system able to generate automatically some *a priori* interesting strategies, given constraints and criteria to optimize. In this perspective we are implementing some of Machine Learning technics in a tool conceived as an Exploration Support System.

2. Towards an exploration support system

2.1 Automatic generation of strategies

The problem of automatically generating a set of decision rules for the winter wheat crop production can be theoretically considered as a Markov decision problem [PUT94] with the following important characteristics:

- our problem is a finite-horizon one, with particular state-spaces and decision-spaces at each stage;
- state and decision variables can have continuous domains;
- the time (decision date, observation date) must be explicitly taken into account;
- the stochastic evolution of the crop process due to weather uncertainty is an essential feature of the problem;
- we do not have a markovian formal model of the crop growth process and we have to deal with delayed rewards.

Unfortunately, most of these characteristics preclude us from using classical dynamic programming algorithms to generate optimal strategies. Among these reasons, the principal one is the complexity and time consuming property of the resulting PD algorithms. Indeed, these classical resolution methods need an explicit and flat discrete representation of state and decision spaces, which would necessarily have an huge size in the case of our realistic decision problem. Furthermore, the resolution principle of these algorithms is based on a perfect knowledge about the stochastic dynamics of the controlled system, that is the probability distribution governing the next state of the system given the current state and the current action, which is usually represented as a set of transition matrices. In our case these matrices are not available, since we do not have an explicit formal model of the crop growth, but instead our knowledge about the biophysical processes involved is embodied in the DECIBLE simulator. A solution would consist in estimating these probabilities by simulating a large number of different sequences of actions with DECIBLE, but that would be a second source of computational complexity. Finally, a last objection to the dynamic programming approach concerns the choice of the decision rule representation of a strategy. To exhibit such decision rules from the resulting map produced by a dynamic programming algorithm would require the use of classification and rule learning tools, and that would be a third source of algorithmic complexity.

2.2 An iterative evaluation/modification approach

Hence using classical dynamic programming technics is obviously not adapted to define a system that generates a set of decision rules for the crop production problem. The opposite approach we have chosen consists in defining an Exploration Support System which iteratively generates, transforms and tests different strategies directly expressed as sets of decision rules, until it provides a high quality solution.

The system directly manipulates strategies on an interactive way, via a syntactic reformulation into a well-suited language for algorithmic computations. At each step, the more or less local strategy modifications depend on the result of the strategy evaluations. The system stops when the current-strategy's quality is above a threshold, or when this quality does not improve anymore. Our expectation is that this general approach, since it defines interesting strategies in an incremental way, is really adapted to the present exploration context.

Among the different methods coming within this Machine Learning approach, the reinforcement learning and the genetic based learning technics are probably the most interesting ones. We have chosen to use them jointly in the definition of our exploration support system.

3. Reinforcement learning, genetic algorithms and crop management

3.1 Machine learning algorithms

Reinforcement learning (RL) is one of the major approaches to solve Markov decision problems with unknown transition probabilities. It consists in the learning of a mapping from situations to actions (control rules) so as to maximize an expected scalar reward. Q-learning [WAT92], one of the most studied reinforcement learning methods, is a direct adaptive method since it does not rely on an explicit model of the Markov process. It only maintains estimates Q_n of utilities Q^* for each state-action pair (s,a) , an optimal action for state s being any action a that maximizes $Q^*(s,a)$.

$$\pi^*(s) = \arg \max_a Q^*(s,a)$$

In the case of a finite-stage process like in a crop management problem, the estimate Q_n of the function Q^* is regularly updated after each trial of the crop simulation, according to the selected actions and to the observed final reward like yield for instance. In our case, the functions Q correspond to the strategy, where the states s have to be derived from the crop indicators and the actions a are the technical acts like seeding or fertilizing. The principal advantage of reinforcement algorithms like Q-learning relies in the fact they do not necessitate a probabilistic model of the controlled system, since they just need to observe the transitions and the associated rewards in the course of the process evolution. Nevertheless, like for dynamic programming algorithms, it is difficult to adapt them to more compact strategy representations like decision rules.

The Genetic Algorithms (GA) are another approach to determine a set of decision rules of maximal quality [HOL75]. Indeed, the general aim of GA is to optimize a function f (the *fitness*) on a space E of very large dimension, where elements $x \in E$ can be represented as an ordered sequence of valued attributes. The principle of GA is simple and is inspired from the Darwinian selection and evolution processes with Mandelian representation. It consists in generating randomly or from expert knowledge an initial population $X \subset E$, and then successively at each iteration evaluating each element of the current population X , and transforming the population X into a new population X' , the average-fitness of which is improved. The transformation of X consists first in reproducing on the basis of their relative fitness the elements of the population. Then the selected elements are recombined by idealized crossing-over or mutation operators to form a new population X' .

In our case, the most promising approach to adapt GA to the sequential decision rule problem we encounter seems to be the one developed in the SAMUEL system [GRE90], where elements x of E are sets of decision rules explicitly represented. We propose to adapt this approach to our problem with an extension of the credit assignment procedure used in SAMUEL, by using reinforcement learning algorithms to adjust the rule strengths within a strategy.

3.2 Rule and strategy representation

For the purpose of implementing this exploration support system, decision rules of each decision steps have been simplified in regards of the DECIBLE representation. A strategy is a set of sub-strategies $\{SS1, \dots, SSn\}$, and each sub-strategy is defined as a set of decision rules $\{R1, \dots, Rp\}$. The number n of decision steps is a constant of the model, which is not the case for the number of rules.

At each step i is associated a state space S_i and a decision space D_i . A decision rule R maps each state s of S_i to a decision d of D_i . If $S_i = \{s = (s1, \dots, sp)\}$ and $D_i = \{d = (d1, \dots, dq)\}$, then the decision rules of the stage i are formally represented as follows:

R : weight ω ,
 IF $s1 \in [s1_{begin}; s1_{end}] \wedge \dots \wedge sp \in [sp_{begin}; sp_{end}]$
 THEN $d1 \in [d1_{begin}; d1_{end}] \wedge \dots \wedge dp \in [dq_{begin}; dq_{end}]$

where ω is a real value characterizing the strength of the rule R. We note $R = (SR, DR, \omega)$, with $SR = [s1_{begin}; e1_{end}] \times \dots \times [ep_{begin}; ep_{end}]$, $DR = [d1_{begin}; d1_{end}] \times \dots \times [dq_{begin}; dq_{end}]$.

We say a rule is *activable* in the current state s if $s \in SR$. An activable rule recommend any decision $d = (d1, \dots, dq)$ such that $d \in DR$. Thus these rules are non-deterministic and smaller are the sizes of the intervals $[dj_{begin}; dj_{end}]$, less important is this non-determinism. Similarly, the state-space representation will be as much accurate as the size of the intervals $[ej_{begin}; ej_{end}]$ will be small. But in these cases, the minimal number of rules necessary to cover the complete state space S_i will be more important. This trade off between rule number and rule domain sizes is a critical point.

For algorithmic reasons, the domains SR_j of the rules do not form a partition of S_i at step i : some overlap is possible. In this case we need to choose between different activable rules in a same state s in S_i , and we retain the one which have the greater weight ω . Hence, the non-deterministic decision function at step i is:

IF $\omega_j = \text{argmax}\{\omega_k / R_k = (SR_k, DR_k, \omega_k) \text{ activable in } s\}$
 THEN d is drawn randomly in DR_j

We say a rule R is *active* if it is activable ($s \in SR$), and if the chosen decision $d \in DR$. Several different rules can be simultaneously active.

3.3 Strategy evaluation

The learning algorithms we employ require numerical evaluations of the strategies which are tested. The evaluation of a strategy is conducted with the DECIBLE simulator. For a given one year weather time serie, the execution of the strategy is simulated decision stage after decision stage, from the initial conditions before seeding until the harvest. At this level, non-determinism results only from the non-deterministic rule-based decision procedure. In order to select interesting strategies, a numerical criterion is used to grade this strategy execution. Usually this criterion can be decomposed into a sum of local rewards at each steps, but it is not essential, and it can only depend of the final situation (for instance the yield). Furthermore, this criterion can be the result of numerous calculations (veto, weighted sum, etc.) aggregating different performances. To take into account the fact that users do not know the weather in advance, and that for a given strategy the rule representation is non-deterministic, we finally have to consider the expected value of this criterion. Thus the global criterion we try to optimize has the following form:

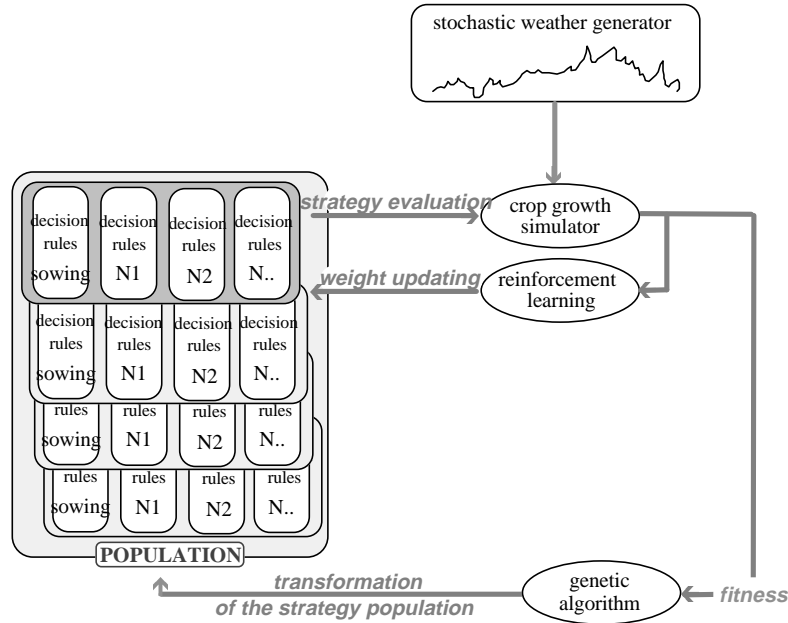
$$V(S) = E\left(\sum_{i=1}^n r_i\right)$$

This expected value cannot be of course calculated analytically, and we can only estimate it, by simulating the strategy S with a large number of different weather sequences. Unfortunately, the available time series of the weather parameters used in DECIBLE were not numerous enough for a correct statistical evaluation of $V(S)$. Furthermore, as we will see, this evaluation is necessary for a lot of different strategies S. Thus we have decided to use a stochastic weather generator, that can provide as many different sequences as the evaluation module needs. The weather generator we use is an extension of the one developed by Racsko and Semenov¹ [RAC91].

¹We thank A. Harnos and M. Semenov for the C code of the weather generator, and D. Lenhart for helpful discussions.

3.3 Learning the rules

For the representation we have chosen, learning a strategy consists in learning at each decision step the associated sub-strategy, that is the number of rules, the rule domains (SR, DR) and the weights ###. We clearly distinguish the generation of the rule structure (number and domains) from the learning of the weights. Indeed, the former is a symbolic learning task, and the latter a numerical learning task. Hopefully, genetic algorithms and reinforcement learning can be combined to achieve the learning of global strategies, as this is represented in the following figure:



Let us assume fixed the structure of the rules. The weight learning has for purpose to remove potential conflicts between different simultaneously activable rules. We achieve this learning task with reinforcement learning technics. They lead to the definition of algorithms that iteratively update the weight values: at step i , if we observe the transition $(s_{\text{current}}, d_{\text{current}}, s_{\text{new}}, r_{\text{current}})$, then we update the weights of all the active rules relatively to $(s_{\text{current}}, d_{\text{current}})$:

$$w_{n+1} = (1 - \alpha_n)w_n + \alpha_n \{r_n + \Psi(s_{\text{new}})\}$$

where α_n is a small learning rate decaying over time and $\Psi(s_{\text{new}})$ is an estimation of the value of the rest of the best trajectory starting from s_{new} . We have analysed and compared different reinforcement learning algorithms, like Q-learning [WAT92] and R-learning [SCH93], that correspond to different functions $\Psi()$. One critical point of our approach is to adapt them to the finite horizon assumption and the decision rule representation.

The second major problem is then to learn the structures of the sub-strategy rules. We use for that a genetic algorithm which is characterized by the followings points:

- each individual in the population represents a strategy, i.e. sets of decision rules;
- the *fitness* of an individual S is the evaluation of the value $V(S)$;
- *crossing-over* between two strategies, within a given decision step, consists either in the exchange of two complete rules, or in the creation a new rule which is obtained as a barycenter of the same two rules.
- *mutation* of a strategy consists in one part in the random modification of one or more rules, in one or more decision steps, i.e. in the local modification of some domains SR and DR. A second kind

of mutation corresponds to the elimination of rules with small domains, to the partitioning of rules large domains, and to the merging of rules the domains of which are close.

The two learning algorithms are used jointly. At each iteration, a population of N strategies is maintained. Each strategy is evaluated with the DECIBLE simulator, and during these simulations, the weights of the strategy rules are updated. Then the fitness of the resulting strategies are calculated and the current population is transformed into a new population. The general structure of our approach is represented on the following figure:

4. Conclusion

After numerous experiments with the DECIBLE winter wheat crop management simulation tool that has been developed for the last years, we are convinced that crop production plan engineers need help to imagine different solutions than the ones used in practice. The Exploration Support System we have presented in this paper is a first answer to this problem. It combines simulation and learning technics to help users to explore a wide range of possible strategies, that have been pre-selected by the system on the basis of an evaluation module. This system is still under development, but the current results already suggest two main topics that require further investigations:

- from the Modelling point of view: how do the reinforcement learning and genetic algorithm technics enable the automatic generation of interesting crop management strategies, given production constraints and complex evaluation criteria ? Assuming a positive answer for the winter wheat domain, will it be possible to generalize this tool to different biophysical systems ? under which assumptions concerning the biophysical models ?
- from the Decision point of view, can these simulation-based learning approaches enhance the facility of using decision support systems by researchers, advisers or even farmers ? Do the strategy formalism we have retained is sufficient for representing actual crop management ? Can this exploration support system really generate new peculiar strategies and so develop new knowledge ? Any answer to this last question will be an important progress for the DSS community.

5. References

- [ATT96] J.M. Attonaty, M.H. Chatelin, F. Garcia *Interactive simulation modelling in farm decision making*. In Proc. of Int. Congress for Computer Technology in Agriculture, ICCTA'96 (1996).
- [GRE90] J.J. Grefenstette, C.L. Ramsey and A.C. Schultz *Learning sequential decision rules: Using simulation models and competition*. Machine Learning, 5, 4 (1990).
- [HOL75] J.H. Holland *Adaptation in natural and artificial system*. Ann Arbor: University Michigan Press (1975).
- [PUT94] M.L. Puterman *Markov Decision Processes*. Wiley, New-York (1994).
- [RAC91] P. Racsko, L. Szeidl and M. Semenov *A serial approach to local stochastic weather models*. Ecological Modelling, 57 (1991).
- [SCH93] A. Schwartz *A reinforcement learning method for maximizing undiscounted rewards*. In Proc. of Int. Conf. on Machine Learning, ML'93 (1993).
- [WAT92] C.J. Watkins and P. Dayan *Technical note: Q-learning*. Machine Learning, 8, 3-4 (1992).