Neuroevolutionary Inventory Control in Multi-Echelon Systems*

Steve D. Prestwich[1], S. Armagan Tarim[2], Roberto Rossi[3], and Brahim Hnich[4]

[1] Cork Constraint Computation Centre, Ireland
[2] Operations Management Division, Nottingham University Business School, Nottingham, UK
[3] Logistics, Decision and Information Sciences Group, Wageningen UR, The Netherlands
[4] Faculty of Computer Science, Izmir University of Economics, Turkey
s.prestwich@cs.ucc.ie, armtar@yahoo.com.tr,
roberto.rossi@wur.nl, brahim.hnich@ieu.edu.tr

Abstract. Stochastic inventory control in multi-echelon systems poses hard problems in optimisation under uncertainty. Stochastic programming can solve small instances optimally, and approximately solve large instances via scenario reduction techniques, but it cannot handle arbitrary nonlinear constraints or other non-standard features. Simulation optimisation is an alternative approach that has recently been applied to such problems, using policies that require only a few decision variables to be determined. However, to find optimal or near-optimal solutions we must consider exponentially large scenario trees with a corresponding number of decision variables. We propose a neuroevolutionary approach: using an artificial neural network to approximate the scenario tree, and training the network by a simulation-based evolutionary algorithm. We show experimentally that this method can quickly find good plans.

1 Introduction

In the area of optimisation under uncertainty, one of the most mature fields is inventory control. This field has achieved excellent theoretical and practical results using techniques such as dynamic programming, but some problems are too large or complex to be solved by classical methods. Particularly hard are those involving *multi-echelon systems*, in which multiple stocking points form a supply chain. In such cases we may resort to simulation-based methods. Simulation alone can only evaluate a plan, but when combined with an optimisation algorithm it can be used to find near-optimal solutions (or plans). This approach is called *simulation optimisation* (SO) and has a growing literature in many fields including production scheduling, network design, financial planning, hospital administration, manufacturing design, waste management and distribution. It is a practical approach to optimisation under uncertainty that can handle problems containing features that make them difficult to model and solve by other methods: for example non-linear constraints and objective function, and demands that are correlated or have unusual probability distributions.

* B. Hnich is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 05/IN/I886.

F. Rossi and A. Tsoukis (Eds.): ADT 2009, LNAI 5783, pp. 402–413, 2009.
© Springer-Verlag Berlin Heidelberg 2009

SO approaches to inventory control are typically based on policies known to be optimal in certain situations, involving a small number of reorder points and reorder quantities. For example in (s, S) policies whenever a stock level falls below s it is replenished up to S, while in (R, S) policies the stock level is checked at times specified by R, and if it falls below S then it is replenished up to S. SO can apply standard optimisation techniques such as genetic algorithms to these policies by assigning genes to reorder points and replenishment levels. In more complex situations involving constraints, multiple stocking points, etc, these policies may be suboptimal in terms of expected cost, though they can have other desirable properties such as improved planning stability. But a cost-optimal plan for a multi-stage problem with recourse must specify an order quantity in every possible scenario, so the plan must be represented via a *scenario tree*. The number of scenarios might be very large, or infinite in the case of continuous probability distributions, making the use of SO problematic. Scenario reduction techniques may be applied to approximate the scenario tree, but it might not always be possible to find a small representative set of scenarios.

An alternative form of approximation is to use an artificial neural network (ANN) to represent the policy. For example, the inputs to the ANN could be the current stock levels and time, and the outputs could be the recommended actions (whether or not to replenish and by how much). We must then train the ANN so that its recommendations correspond to a good plan. No training data is available for such a problem so the usual ANN backpropagation training algorithm cannot be applied. Instead we may use an evolutionary algorithm to train the network to minimise costs. This *neuroevolutionary* approach has been applied to control problems [8,9,21] and to playing strategies for games such as Backgammon [16] and Go [14], but it has not been extensively applied to inventory control. In this paper we apply neuroevolution to stochastic inventory control in multi-echelon systems. Section 2 presents our method, Section 3 evaluates the method experimentally, Section 4 surveys related work, and Section 5 concludes the paper.

2 A Neuroevolutionary Approach

To approximate the scenario tree, we construct a function whose input is a vector containing the time period and current inventory levels, and whose output is a vector of order quantities (which might be zero). We design the function automatically by simulation optimisation.

2.1 Scenario Tree Compression by Neural Network

An obvious choice for this function is an artificial neural network (ANN), which can approximate any function with arbitrary accuracy given a sufficient number of units. ANNs also come with a ready-made algorithm for optimisation: the well-known backpropagation algorithm. However, there is a problem with this approach: we do not have training data available (this also precludes the use of Support Vector Machines). To obtain training data we would have to solve a set of instances, and there is no known method for solving the harder instances to optimality. Instead we must use an ANN to solve a problem in *reinforcement learning*, in which we must choose its weights in order