An Integrated Symbolic and Neural Network Architecture for Machine Learning in the Domain of Nuclear Engineering

Ephraim Nissan Hava Siegelmann Alex Galperin

Mathematics
Bar-Ilan University
Ramat-Gan, Israel

Industrial Engineering
Technion
Haifa, Israel

Nuclear Engineering
Ben-Gurion University
Beer-Sheva, Israel

Abstract

On top of FUELCON and NEL, two extant, successful projects in, respectively, expert systems for engineering, and neural networks, we have defined and designed a new phase, meant to greatly increase the significance, for AI, of the combined project with respect to the already recognized merits of the two seed-projects. The NEL symbolic-to-neural conversion schema and language is resorted to in NEURALIZER, a component meant to automatically revise a ruleset, iteration after iteration, within the operation cycle of FUELCON, a generator of families of configurations of fuel assemblies for reloading the core of nuclear reactors.

1 Introduction

NEL is a language and translation schema, for transforming rulesets into neural networks [9]. Its application to the FUELCON project in nuclear engineering expert system. In FUELCON, a ruleset is applied to generate families of good configurations of fuel in nuclear reactor cores. Whereas the practitioner may be satisfied with the output, the domain expert is challenged to optimize not just the configurations, but his or her own heuristics: the results of one given iteration of the expert system are simulated by another component, which prompts the human expert to (manually) improve on the ruleset previously (manually) formulated. The goals of the new phase of the project is to obtain automated ruleset revision, and, thus, full automation of the discovery process.

2 Fuel Management in Nuclear Power Plants

The core of a nuclear reactor can be schematized as a grid; and reasoning upon it can be reduced to just a slice, because of symmetry. Assemblies (i.e. packages resembling parallelepipeds) of nuclear fuel rods are inserted vertically in the square cases of the grid. Once we know there are different kinds of fuel assemblies, e.g. based on the times the given assembly has been already used in the core, during the past one or two operation intervals, how should the available pool of fuel assemblies be allocated into the geometry of the reactor core? Heuristics to do that exist, that are meant to solve the so-called in-core fuel management problem (also called refueling or reloading). Among the rods that make up the asseemblies, liquid coolant circulates: this is required because of security reasons. It is easy to see the kind of constraints heuristics should keep present: on the one hand, security; and on the other hand, the requirement that the core be kept "critical", i.e. with a sustained chain reaction, ensuring long operation cycles. These are periods of about one year of duration. Annualy, plants are shut down, and, because of the loss of power production, the costs of the shut-down periods are staggering. On the other hand, if we manage (as we actually manage) to speed up the refueling design process (and also to enhance the quality of the configurations generated: which FUELCON also, more importantly, does: more importantly, as this is reflected on both efficiency and the duration of the next operation period), then the economical advantage is in the order of millions of dollars for each shut-down period.

The engineer who is responsible of the in-core fuel management, cannot produce beforehand the design for the next fuel reload, and has to wait for the shut-down period to carry out his calculations.

1051-4651/94 $04.00

This is due to the fact that the exact parameters of the shut-down configuration will change the desired solution totally. Indeed, good configurations are known from the research literature of nuclear engineering, but such configurations are based on real-case studies that were carried out on a here and now basis: at a given plant, and, in particular, for a given shut-down situation. You cannot provide an accurate forecast the situation at your own plant at the end of the current cycle. Published configurations are not robust enough to be appropriate for the problem you got once your plant is actually shut down. However, they constitute the archetype for classes of solutions: the solution for the problem you actually got may be found by taking one known solution, and then modifying it – this is called reshuffling – by switching cases in the grid of the reactor core geometry, until you get a solution that is both admissible and good for solving your own problem. Actually, this is the way the practitioner works, and IntelliCorp even developed an expert system, based on the KEE shell, that assists in the reshuffling process.

In the FUELCONE project we allow for a wider serach space, generating hundreds of configurations (a "cloud") belonging to "families," instead of getting to just one solution (that is or is not a local optimum) as usual at the state of the art of nuclear engineering. Then, once you get a family of solutions as generated by a given ruleset, you run all of these solutions through a numeric and visual simulator (ours is called NOXER), and see how good the configurations you got are: this is ascertained according to where the respective dots fall within a window of admissibility, in a plane of cartesian coordinates. All you have to do, is to pick up the configuration (i.e. dot) you find closest (more or less) to the "northeastern" corner of the admissibility window.

The domain expert, however, is likely to be prodded, by the results of the simulation, to try something better: couldn't his or her ruleset be modified (especially by generalizing or particularizing given optional rules), in such a way that during the next iteration of running the configurations generator and then the simulator, the "cloud" of configurations would move closer to the northeastern corner? That is, can the system learn and improve using its own experience? This is what we achieved and describe in this paper. The learning from experience is done by combining neural network methodologies into our ruleset.

We are not going to describe shortly the nuclear engineering task and the symbolic expert system that was developed in the previous phase of the FUELCON project. An extensive description of FUELCON can be found in Galperin, Kimhi and Nissan (1993). As to the task, within nuclear engineering, independently of the tool, a thorough tretment is provided by Cochran and Tsoulfanidis (1990). Parks and Lewins (1992) review computer-assisted approaches to nuclear fuel reload design.

3 The Ruleset

Let us discuss the expert's rules of thumb of which FUELCON rulesets are made up. A rule either forbids or suggests positioning of fuel of some given kind in certain regions of the core. For each configuration generated, the generator starts with an empty core schema (actually, just a slice, as in a one-eighth symmetry), and gradually fills it, with one fuel-assembly at a time. The rules concern certain regions of the core, and given types of fuel-assemblies.

For example, one rule is *Don't load a fresh assembly in such a position that is adjacent to another position where there is another assembly of the same kind, except when one of those two positions is in a corner position*

This is a *mandatory rule.* In FUELCON, it is the same as an *elimination rule:* as expressed in English, such rules start by *Don't.* The ruleset of FUELCON consists of two parts. The first subset of rules, in FUELCON, is dictated by reactor physics considerations. Such rules have to be enacted under all circumstances, and are executed before the rules of the second subset. The latter consists of *preference rules,* that are meant to yield effectiveness. The subset of preference rules prescribes a course of action; a sample such a rule is: *If it is a twice-burned assembly that is currently being considered, then choose for it — from amongst those positions that were not forbidden by the elimination rules — that position whose distance from the center of the core is minimal.*

4 Rule Translation: An Example

NEL, developed by Siegelmann (1993) is a high-level language along with a translation schema from symbolic code to neural representation. NEL is syntactically rich, and allows to express the constructs

and structure data of conventional or symbolic, sequential and parallel programming.

Let us see how to construct a network out of the rules of FUELCON. Consider the prohibition rule from previous section. The rule can be written as a NEL function that receives as input the record A and a position s, and decides whether the position contradicts the rule considered. In the following function, we write the reserved words of NEL in boldface and the predicates in italics. Lines are numbered successively.

1. **Function** rule-2 (A, s): Boolean;
2. **var** p: Integer, flag: Boolean;
3. **Begin**
4. p=0 ;
5. flag = Good-position ;
6. **If**
7. $((A.\text{burnt} = \text{fresh}) \bigwedge (\neg\ \textit{corner(s)})$
8. **then**
9. **Repeat**
10. $p = p + 1$;
11. **If**
12. $(\textit{neighbor}(s,p) \bigwedge \neg \textit{corner}(p) \bigwedge \textit{kind}\ (A) = \textit{kind}(\textit{assembly}\ (p)))$
13. **then**
14. flag = Bad;
15. **Until**
16. (flag = Bad-position) $\bigvee$ $(p = 20)$;
17. rule-2 = flag
18. **End**;

This rule can be translated into either a simple feed-forward network that tests the 20 positions simultaneously or into a recurrent network that tests them serially. This tradeoff of hardware and time will be decided upon the exact application.

In the recurrent network implementation: there is a neuron for each variable, temporary variable, as well as for the distributed representation of the program counter. The function includes the variables p and flag, as well as the rule-2 itself. In addition, each expression implies an expression variable (and possibly some temporary variables as well). The program counters are pc_1 to pc_{18}. Sample substitutions are: $p = 0 \cdot pc_4 + (p+1) \cdot pc_{10} + p(1\text{-}pc_4\ \text{-}pc_{10})$; rule-2=flag$\cdot\ pc_{16}$; $pc_8 = \sigma(pc_7 + v_7\ \text{-}1)$

The following remark is called for, concerning the appropriateness of the neural-net reinforcement learning adopted in the FUELCON / NEL project. The rulesets that the human experts developed for FUELCON, are not large. Of about a dozen rules, one half or more are mandatory rules, that should not modified. Just half a dozen of less rules are those concerned by the optimization effort. This makes the situation very satisfactory, for learning. Indeed, had we large rulesets to optimize, than learning could be expected to be very slow.

References

[1] Cochran, R.G. and N. Tsoulfanidis. *The Nuclear Reactor Cycle: Analysis and Management.* American Nuclear Society (La Grange Park, IL, 1990).

[2] Galperin, A., Kimhi, Y. and E. Nissan. FUELCON: An Expert System for Assisting the Practice and Research of In-Core Fuel Management and Optimal Design in Nuclear Engineering. Computers and Artificial Intelligence 12, 4 (1993): pp. 369-415.

[3] A. Ginsberg, 1988, Theory Revision via Prior Operationalization, Proceedings of the Sixth National Conference on Artificial Intelligence, pp: 590.

[4] J. Hertz and A. Krogh and R.G. Palmer, 1991, Introduction to the Theory of Neural Computation, Addison-Wesley Publishing Company, Inc., Redwood City, CA

[5] R. Maclin and J.W. Shavlik, 1993, "Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding" Machine Learning 11, pp. 195–215.

[6] D. Oursten and R.J. Mooney, 1990, "Changing Rules: A Comprehensive Approach to Theory Refinement", Proceedings of the Eighth National Conference on Artificial Intelligence, p. 815 ff.

[7] G. T. Parks and J. D. Lewins, "In-core fuel management and optimization: the state of the art," *Nuclear Europe Worldscan* **12** (3/4), p. 41 (1992).

[8] M.J. Pazzani, 1989, "Detecting and Correcting Errors of Omission after Explanation-Based Learning", Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, p. 713 ff.

[9] Siegelmann, H.T., 1993. Foundations of Recurrent Neural Networks. Ph.D. Dissertation, Rutgers University.

[10] Towell, G. and J. Shavlik, 1993, "Extracting Refined Rules from Knowledge-Based Neural Networks," Machine Learning 13(1), pp. 71-101.