Evolving Petri Nets with a Genetic Algorithm

Holger Mauch

University of Hawaii at Manoa, Dept. of Information and Computer Science,
1680 East-West Road, Honolulu, HI 96822
hmauch@hawaii.edu

Abstract. In evolutionary computation many different representations ("genomes") have been suggested as the underlying data structures, upon which the genetic operators act. Among the most prominent examples are the evolution of binary strings, real-valued vectors, permutations, finite automata, and parse trees. In this paper the use of place-transition nets, a low-level Petri net (PN) class [1,2], as the structures that undergo evolution is examined. We call this approach "Petri Net Evolution" (PNE). Structurally, Petri nets can be considered as specialized bipartite graphs. In their extended version (adding inhibitor arcs) PNs are as powerful as Turing machines. PNE is therefore a form of Genetic Programming (GP). Preliminary results obtained by evolving variable-size place-transition nets show the success of this approach when applied to the problem areas of boolean function learning and classification.[1]

1 Overview

PNs are a nice formalisms that allow the complex modeling of numerous real world problems. Their strength is the "built in" concurrency mechanism, which allows a formal description of systems containing parallel processes. For this study small sample problems have been used to test the feasibility of PNE. The PNs evolved belong to a class of low level PNs called *place-transition nets*.

This work has been inspired by the generalization of the recombination operator from trees to bipartite graphs. Since recombination seems to be an important genetic operator in PNE, this approach follows the GA stream within EC.

The common feature of traditional GP and PNE is the variable length genotype. This is in contrast to the standard GA which employs fixed-length strings. In contrast to traditional tree-based GP, there is no need to specify a function set explicitly for PNE. Functions are implicitly built into the semantics of the chosen PN model. Because the place-transition PN model used is integer-based our focus is on discrete problems rather than continuous ones.

PNE evolves a population of PNs in an evolution-directed search of the space of possible PNs for ones, which, when executed, will exhibit high fitness. To the best of our knowledge the idea of evolving a population of individuals which are variable-sized PNs does not appear in previous literature.

[1] This research was supported in part by DARPA grant NBCH1020004.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1810–1811, 2003.

2 Elements of Petri Net Evolution

In experiments PNs have been evolved for boolean function learning and classification. We partition the set of places $P = P_{in} \cup P_{center} \cup P_{out}$. For a classification problem such as *PlayTennis* [3, p.59] the input places P_{in} contain the values of the attributes of a fitness case in its initial marking. After firing a suitable number of transitions the PN is evaluated on its output places P_{out} according to the induced submarking M_{out}. Every PN individual is executed on several fitness cases to determine its fitness.

One problem with the evaluation of a PN is its potentially nondeterministic behavior due to conflicts (i.e. the situation when two transitions are enabled, but firing one of the two disables the other one.) To solve this difficulty the fitness measure works with the mean value of a sample of simulation runs. This introduces a random component (the firing order) into the fitness evaluation process.

The PNE implementation uses the incidence matrix representation as described in [2] to represent the directed, weighted, bipartite graph data structure of a place-transition net. The incidence matrix and the initial marking vector M_0 are implemented by a low-level data structure that can grow dynamically.

After random generation of generation 0 PNs evolutionary computation enters the cycle of evaluation, selection and recombination/mutation. Several mutation operators to manipulate the various components (i.e. arcs, places, transitions, initial marking) of a PN have been designed. Mutation also allows a PN to grow and shrink in size. At the heart of PNE is the specifically designed recombination operator which creates 2 offspring PNs from 2 mating parent PNs. The parents do not need to match in size, so every individual in the population can mate with any other individual. A conventional tournament selection with tournament size 2 is used.

In addition to the usual parameters (population size, maximum number of generations, parameters for recombination, mutation and selection) PNE uses a parameter for the maximum duration of a single PN execution, since termination is not guaranteed when a PN is executed.

First results in the area of classification and boolean function learning show that numerous runs produced highly fit PN individuals. However, the examples examined are very simple in nature. Future work will involve larger sized problem instances. That will allow the evaluation of the robustness of PNE. What has been shown is the feasibility of successfully evolving PNs when suitable procedures for the evaluation, mutation and recombination are employed.

References

1. Reisig, W.: Petri Nets: an Introduction. Springer-Verlag, Berlin, Germany (1985)
2. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77** (1989) 541–580
3. Mitchell, T.M.: Machine Learning. WCB/McGraw-Hill, Boston, MA (1997)