Optimization of Petri nets structure using genetic programming.

Eva Schwardy

Dept. of Cybernetics and Artificial Intelligence

Faculty of Electrical Engineering and Informatics

University of Technology Košice

Slovakia

Abstract: Petri nets are graphs representing parallel and asynchronous processes. Optimization of a net includes components and their values setting. In large models, it seems to be efficient to get use of computational strength of evolutionary algorithms and to search for the optimal net configuration by means of genetic programming.

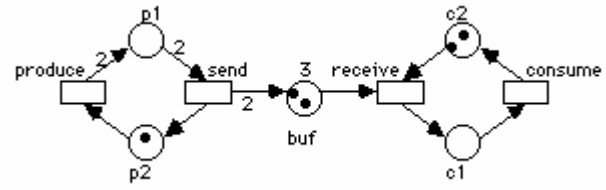
Keywords: Petri nets, evolutionary algorithms, genetic programming.

1 Petri nets

A Petri net is a graphical and mathematical modeling tool. It consists of *places*, *transitions*, and *arcs* that connect them. *Input arcs* connect places with transitions, while *output arcs* start at a transition and end at a place. There are other types of arcs, e.g. *inhibitor arcs*. Places can contain *tokens*; the current state of the modeled system (the *marking*) is given by the number (and type if the tokens are distinguishable) of tokens in each place. Transitions are active components. They model activities that can occur (the transition *fires*), thus changing the state of the system (the marking of the Petri net). Transitions are only allowed to fire if they are *enabled*, which means that all the preconditions for the activity must be fulfilled (there are enough tokens available in the input places). When the transition fires, it removes tokens from its input places and adds some at all of its output places. The number of tokens removed/added depends on the cardinality of each arc. The interactive firing of transitions in subsequent markings is called token game.

Petri nets are a promising tool for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, parallel, non deterministic, and/or stochastic. As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems.

Petri nets are a formal, graphical, executable technique for the specification and analysis of concurrent, discrete-event dynamic systems appropriate for modeling systems with concurrency - graphs representing parallel and asynchronous processes that can demonstrate system contents and actions. A Petri net is an oriented graph defined by a sixtuple of elements [P, T, F, K, W, M_0].



- S = {p1, p2, buf, c1, c2}
- T = {produce, send, receive, consume}
- K(buf) = 3, the capacity of the other places assumed to be infinite
- W(produce, p1) = W(p1, send) = W(send, buf) = 2, other arc weights are assumed to be 1
- M_0 (p2) = 1, M_0 (buf) = 2, M_0 (c2) = 2;

P finite set of places

T finite set of transitions

F forward incidence function

K capacity function which expresses the maximal number of tokens each place may contain

W weight function which expresses how many tokens flow through arcs at each transition involved

M_0 initial marking

Places and transitions are peaks of a graph joined by oriented arcs. To model dynamic characteristics of complex systems as well marking of a graph (placing tokens) is defined. The state of a net is changing by realization (firing) of transitions. Forward accessibility class defines a set of markings that are available from the current marking.

To study performance and dependability issues of systems it is necessary to include a timing concept into the model. There are several possibilities to do this for a Petri net; however, the most common way is to associate a *firing delay* with each transition. This delay specifies the time that the transition has to be *enabled*, before it can actually fire. If the delay is a random distribution function, the resulting net class is called *stochastic Petri net*. Different types of transitions can be distinguished depending on their associated delay, for instance *immediate transitions* (no delay), *exponential transitions* (delay is an exponential distribution), and *deterministic transitions* (delay is fixed).

The concept of Petri nets has its origin in Carl Adam Petri's dissertation *Kommunikation mit Automaten*, submitted in 1962 to the faculty of Mathematics and Physics at the Technische Universität Darmstadt, Germany [7,6].

1 Evolutionary algorithms

2.1 Evolutionary computation

Evolutionary computation encompasses methods of simulating evolution on a computer. The term is relatively new and represents an effort bring together researchers who have been working in closely related fields but following different paradigms. The field is now seen as including research in genetic algorithms, evolutionary strategies, evolutionary programming, artificial life, and so forth.

2.2 Genetic algorithms

Genetic algorithm is a type of evolutionary computation devised by John Holland. A model of machine learning that uses a genetic/evolutionary metaphor. Implementations typically use fixed-length character strings to represent their genetic information, together with a population of individuals/chromosomes which undergo crossover and mutation in order to find interesting regions of the search space.

2.3 Genetic programming

Genetic programming is a variant of EA - genetic algorithm applied to programs. Genetic programming is more expressive than fixed-length character string GAs, though GAs are likely to be more efficient for some classes of problems. GP works with genomes of variable length and is used to evolve symbolic information. The evolution of computer code, symbolic regression, and automatic electrical circuit design are main application domains. A common form of representation is the tree-like arrangement of genes as a program [1,2].

3 Why to use GP for Petri nets design optimization?

Genetic algorithms (GA) were used for instance to solve an accessibility problem [5] and time scheduling [3] in Petri nets already. However, they have not been used to optimize the net design although advantages of evolutionary algorithms computational strength in electrical circuit design have been proved and especially in large scales nets are hopeful. In a kind of a sense, the problem of a graph structure optimization turns to a Traveling Salesman one that was successfully approached by the method.

A method of solving the Traveling Salesman Problem (TSP) using Genetic Algorithms has been developed. In the TSP, the goal is to find the shortest distance between N different cities. The path that the salesman takes is called a tour. Testing every possibility for an N city tour would be N! math additions. A 30 city tour would be $2.65 * 10^{32}$ additions. Assuming 1 billion additions per second, this would take over 8,000,000,000,000,000 years. Adding one more city would cause the number of additions to increase by a factor of 31. Obviously, this is an impossible solution.

A genetic algorithm can be used to find a solution is much less time. Although it probably will not find the best solution, it can find a near perfect solution in less than a minute. There are two basic steps to solving the travelling salesman problem using a GA (and so would be in a searching for optimal Petri net). First, create a group of many random tours in what is called a *population*. These tours are stored as a sequence of numbers. Second, pick 2 of the better (shorter) tours *parents* in the population and combine them, using *crossover*, to create 2 new solutions *children* in the hope that they create an even better solution. Crossover is performed by picking a random point in the parent's sequences and switching every number in the sequence after that point.

The idea of Genetic Algorithms is to simulate the way nature uses evolution. The GA uses *Survival of the Fittest* with the different solutions in the population. The good solutions reproduce to form new and hopefully better solutions in the population, while the bad solutions are removed.

Eventually, the GA will make every solution look identical. This is not ideal. There are two ways around this. The first is to use a very large initial population so that it takes the GA longer to make all of the solutions the same. The second method is *mutation*. Mutation is when the GA randomly changes one of the solutions. Sometimes a mutation can lead to a better solution that a crossover would not have found.

The difficulty in the TSP using a GA is encoding the solutions [4]. Similar problem seems to occur in Petri nets. Terminating function should include a forward accessibility condition and a natural representation is advisable.

Conclusions

While not at simple Petri nets, at large scale ones genetic programming seems to provide an efficient tool for their design optimization that has been proved at electrical circuit design and Traveling Salesman Problem already. Problems of representation and evaluation function have to be solved and considerably large and complex problems that resisted to be approached by Petri nets method by time due to their size and complexity can be modeled. Though it was not worked out yet the results are hopeful.

References

[1] Bayer, H.-G., Brucherseifer, E., Jakob, W., Pohlheim, H., Sendhoff, B., Thanh, B. T.: Glossary (Evolutionary Algorithms - Terms and Definitions) http://ls11-www.cs.uni-dortmund.de/people/beyer/EA-glossary/def-engl-html.html

[2] Heitkötter, J., Beasley, D.: Hitch Hiker's Guide to Evolutionary Computation Issue 8.1 29 March 2000 http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/Q99.htm

[3] Kang, S.J.: Jang, S.H.: Hwang, H.S.: Woo, K.B.: Colored timed Petri nets modeling and job scheduling using GA of semiconductor manufacturing. In: *IEICE Trans. on Information and Systems, Vol. E82-D, No. 11*, pages 1483-1485. 1999. http://www.informatik.uni-hamburg.de/TGI/pnbib/k/kang_s_j1.html

[4] LaLena, M.: Travelling Salesman Problem Using Genetic Algorithms http://www.lalena.com/ai/tsp/

[5] Takahashi, K., Yamamura, M., Kobayashi, S.: A GA approach to solving reachability problems for Petri nets. In: *IEICE Trans. on Fundamentals in Electronics, Communications and Computer Science, Vol. E79-A, No. 11*, pages 1774-1780. 1996. http://www.informatik.uni-hamburg.de/TGI/pnbib/t/takahashi_k2.html

[6] Trompedeller, M.: A Classification of Petri Nets http://www.daimi.au.dk/PetriNets/classification/

[7] Zimmermann, A.: Petri Nets http://pdv.cs.tu-berlin.de/~azi/petri.html