EVOLUTIONARY CONSTRUCTOR DESIGN FOR THE SORTING NETWORKS

Michal BIDLO, Master Degree Programme (5th year)
Dept. of Computer Systems, FIT, BUT
E-mail: xbidlo01@stud.fit.vutbr.cz

Supervised by: Dr. Lukáš Sekanina

ABSTRACT

In this paper we present an approach for constructing sorting networks of the arbitrary number of inputs. Genetic algorithm is used to find a constructor that would be able to build an arbitrarily large sorting network. The crucial part of the process is the so-called *development* inspired by biological ontogeny which means that the larger sorting network is created on the basis of its precursor according to the given rules. It is shown that these rules can be designed using genetic algorithm.

1 INTRODUCTION

Many researchers have dealt with building of the sorting networks to optimize their properties. Most of the methods have used evolutionary algorithms to construct a sorting network with a given number of inputs whose implementation cost should be minimized. However, there is a scalability problem in the evolutionary algorithms domain that makes the direct construction of large sorting networks impossible. Hence we will apply the development into the genetic algorithm to obtain a short sequence of valid sorting networks. It is assumed that the subsequent use of the evolved rules will give another valid sorting network [1]. The next two sections briefly introduce the principles of the simple genetic algorithm and sorting networks. Section 4 contains the proposed method and the results obtained. Section 5 gives the discussion and ways for the future research.

2 GENETIC ALGORITHMS

Genetic algorithm belongs to the special class of bio-inspired techniques. It is a stochastic search process working with a set of individuals (a population) that represent suitably coded solutions to a given problem. These individuals are called chromosomes and form a genotype space. Each individual is mapped onto an appropriate solution in a phenotype space. Here we will use the simple form of the genetic algorithm which

works as follows [2]: The initial population is filled with the random chromosomes. In every iteration each solution is evaluated by means of a fitness function. According to the fitness values, two individuals (parents) are selected from the actual population and then they are modified (with a given probability) using genetic operators. The common genetic operators are called crossover and mutation. The two new offspring are placed into the new population. The selection is performed until the new population is fullfilled. If a termination condition is met, then the algorithm is finished, otherwise the new generation is created. We will use elitism in our implementation to prevent loss of the best solution found yet.

3 SORTING NETWORKS

The concept of sorting networks was introduced in 1954; rigorous theory is summarized in [3]. Let *CMPX* operation denote a function with two arguments (a,b) which returns an ordered pair (b,a) if $a > b$, otherwise it returns (a,b). Sorting network is a sequence of the CMPX operations.The sorting network is valid, if it sorts all the sequences applied at its input into the nondecreasing order. According to the zero–one theorem, any sorting network can be tested using 2^n binary vectors [3].

4 EVOLUTIONARY DESIGN WITH DEVELOPMENT

The genetic algorithm performs a special action during the evaluation of each chromosome—development—in which a simple solution known beforehand (an embryo) *grows* progressively to construct a larger sorting network. The development is controlled by rules that take the comparators from the embryo and put them on the next positions to build a valid sorting network. After each developmental step the number of inputs increases. Every developed sorting network contains all the comparators included in the embryo and its ancestors.

4.1 USING THE INSTRUCTION SET FOR THE DEVELOPMENT

We have used a simple instruction set to control the developmental process. The finite sequence of instructions will be called a constructor. Table 1 shows the utilized instructions. There are three important symbols in the instructions which denote various positions in the structure under development (see Figure 1a). The embryo pointer (*ep*) states the position of a comparator which is just processed. The comparator is copied to the location denoted by *np* (next position) with possible modifications of its inputs (according to a type of the instruction). The constructor pointer (*cp*) denotes the position of actual instruction in the constructor. The step of the development is finished if the end of the constructor is reached or if the last comparator of the embryo is processed. The chromosome is implemented as a constant-length array whose elements represent the instructions. The fitness value of a chromosome is defined as the sum of a number of correctly sorted sequences by the sorting network in each developmental step. The number of developmental steps is specified in the algorithm. Let general constructor mean such the constructor that is able to create the sorting network with theoretically infinite number of inputs.

Name	arg1	arg2	Meaning
ModifyA	a	b	$c1 = (c1+a) \bmod w, c2 = (c2+b) \bmod w, cp = cp+1,$ $np = np+1$; w is the size of the actual sorting network
ModifyB	a	b	$c1 = (c1+a) \bmod w, c2 = (c2+b) \bmod w, cp = cp+1,$ $ep = ep+1, np = np+1$
CopyA	k	$-$	copy $w-k$ comparators, $cp = cp+1, np = np+w-k$
CopyB	k	$-$	copy $w-k$ comparators, $cp = cp+1, ep = ep+w-k,$ $np = np+w-k$

Table 1: Instruction set for the development

4.2 OBTAINED RESULTS

Our goal was to evolve a constructor for building the sorting networks of an arbitrary size. By setting the crossover probability at 75%, the mutation probability at 8% and the maximum size of the sorting network under development at 6, we have been able to find a general constructor for the embryo with three inputs. With this setting the algorithm managed to find at least one general constructor in 10 runs. The minimal length of the constructor has emerged to three instructions. For the best result see Figure 1b.

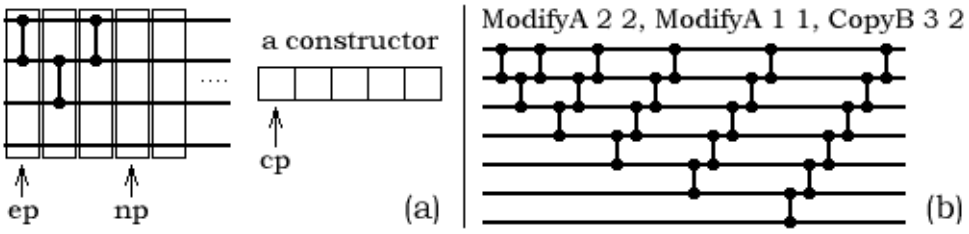

Figure 1: (a) Initial state of the development (embryo), (b) best achieved result

5 CONCLUSIONS

Figure 1b shows that we have found the sorting network corresponding to the straight insertion algorithm. Thus we have rediscovered the principle of this method. We will focus our attention on reducing the number of the comparators in the future research. Another approaches to the development will be studied, namely cellular automata and L-systems.

REFERENCES

[1] Sekanina, L.: Evolving Constructors for Infinitely Growing Sorting Networks and Medians. In: SOFSEM: Theory and Practice of Computer Sc., Springer, 2004, p. 314-323

[2] Bentley, P. (ed.): Evolutionary Design by Computers. Morgan Kaufmann Pub., 1999

[3] Knuth, D.: The Art of Computer Programming (2nd ed.), Addison Wesley, 1998