Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium, Volume 22, No. 1, ISSN 1726-9679
ISBN 978-3-901509-83-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2011
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2011

COMPARISON OF ALGORITHMS FOR EVOLUTIONARY- PROGRAMMING THE TURING MACHINE IMPLEMENTED IN WOLFRAM MATHEMATICA AND MICROSOFT .NET FRAMEWORK

KOURIL, L[ukas] & JASEK, R[oman]

Abstract: *The effectiveness of algorithm execution is an important issue which is necessary to consider during making a descision about implementation of algorithms and selecting runtime environment. The execution time of algorithms doesn't influence the satisfiability of results but the non-effective implementation significantly increases time consumption within execution of algorithm and utilization of the computational appliances is not optimal as well. The effectiveness is fundamental especially for long-term complex computations. This paper provides a comparison of effectivity in the view of execution time of algorithms which ensure evolutionary-estimated programming the Turing machine implemented in Wolfram Mathematica and F# programming language using Microsoft .NET Framework.*

Key words: *wolfram mathematica, F# language, microsoft .NET framework, evolutionary-estimated programming, turing machine*

1. INTRODUCTION

The problem of implementation of algorithms and selection of proper runtime environment is not insignificant. When the suitable form of both of above mentioned is selected, the time consumption of the execution time can be notably decreased thus the effectiveness of algorithm execution and utilization of computational appliances raise.

In this paper, there were considered two environments which ensure the runtime capabilities. These are Wolfram Mathematica – widely-used software tool for functional and symbolic computation which contains own "programming" language and syntax – and Microsoft .NET Framwork 4.0 – an environment for software development. In the case of .NET Framework and the comparison, there was used F# language which is a new functional programming language. Wolfram Mathematica and .NET Framework with F# language are environments which provide suitable resources for algorithms implementation and scientific computations.

The comparison of above-metioned environments is based on research focused on evolutionary-estimated programming the Turing machine. The problems described in (Kouril & Zelinka, 2010) can be considered as sufficiently complex for providing as samples for analysing execution time of related algorithms.

2. METHODS

As metioned above, the analysis is focused on execution time of evolutionary-estimated programming the Turing machine for solving sample problems implemented in Wolfram Mathematica and .NET Framework using F# programming language. Except that, there are also considered core algorithms which provide evolutionary-estimated programming and the Turing machine implementation. It is regarded as necessary to briefly introduce these algorithms, methods and principles of analysis.

2.1 Used algorithms

The evolutionary-estimated programming the Turing machine is explained in (Kouril & Zelinka, 2010) in detail. Nevertheless, the algorithms which the evolutionary-estimated programming utilizes are important for this analysis it is necessary to mention them here. These algorithms are Differential Evolution and the Turing machines.

The Differential Evolution (DE) (Lampinen & Zelinka, 1999; Zelinka et al., 2008) is an algorithm inspired by nature evolution and is considered as a form of artificial intelligence belonging to the evolutionary algorithms. The principle of DE is an evaluation of individuals which represent partial solutions of the problem. The evaluation expresses the quality of individual – the suitability of partial solution in accordance with its arguments. In the cases of nature evolution that is mainly an ability (e.g. strength) of individual to survive. The processing of individuals by DE proceeds in cycles termed as generations which are composed of populations containing individuals. On the basis of quality (which is termed as cost value), the best individuals are passed to the next generation where the process of evaluation repeats. Detailed description of how the Differential Evolution works can be found e.g. here (Lampinen & Zelinka, 1999; Zelinka et al., 2008).

The DE has several parameters. These are number of population (*NP*), mutation constant (*F*), cross-over value (*CR*) and the number of generations (*G*). The mentioned parameters were set as in (Kouril & Zelinka, 2010). The values are presented in following table:

Parameter	Value
Number of population *NP*	100
Mutation constant *F*	0.9
Cross-over value *CR*	0.2
Number of generations *G*	100

Tab. 1. Parameters of Differential Evolution

Second important algorithm for analysis is Turing machine (TM) (Hopcroft et al., 2000). TMs are theoretical automata which can solve problems by sequential approach. TMs have three main parts – the data tape which serves as input/output medium, the head which performs reading and writing operations and the internal stack where states of TM are stored. In each step TM reads one symbol from the data tape. According to the current inner state, TM passes to the new inner state, writes new symbol to the data tape and moves the head.

The TMs are defined by following parameters which mostly depend on the problem which is processed by the TM.

- Set of inner states Q
- Set of input symbols Σ
- The data tape symbols Γ
- The transition function $\delta: Q\backslash F \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- Initial state $q_0 \in Q$
- Blank symbol $B \in \Gamma$
- Set of final states $F \subseteq Q$

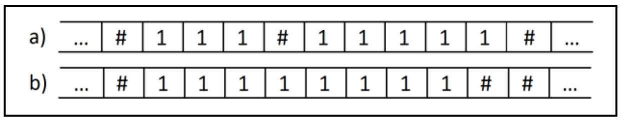
The settings of TM used in relation with the samples and analysis can be seen in 2.2.

The implementations of both algorithms in F# language are parts of F# Artificial Intelligence library, which can be found here (***, 2011a) and downloaded free of charge.

2.2 Sample problems

As mentioned earlier, the samples utilized in the analysis are similar as those introduced in (Kouril & Zelinka, 2010). These are unary addition, problem of divisibility and the problem of primality. Each of them will be briefly explained and settings of TMs will be shown.

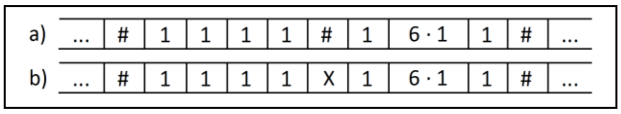
Unary addition can be imagined as addition of numbers 3 and 5 in unary number system. It is necessary to encode the problem to the form of the TM's data tape, which looks like Fig. 1a as well as the requested output of the TM (Fig. 1b).



a)	...	#	1	1	1	#	1	1	1	1	1	#	...
b)	...	#	1	1	1	1	1	1	1	1	#	#	...

Fig. 1. Example of unary addition encoded as the data tapes

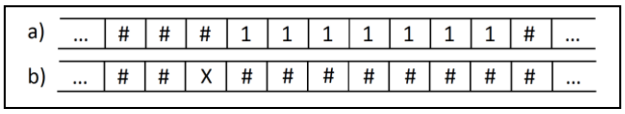
The divisibility problem represents exact divisibility of numbers 8 and 4 realized in unary number system. The input and requested output data tapes which encode this problem are depicted as Fig. 2a and Fig. 2b:



a)	...	#	1	1	1	1	#	1	6 · 1	1	#	...
b)	...	#	1	1	1	1	X	1	6 · 1	1	#	...

Fig. 2. Example of divisibility problem encoded as the data tapes

The primality problem solves the question whether the encoded number is prime number or not. Both of data tapes can be seen in Fig. 3a and Fig. 3b:



a)	...	#	#	#	1	1	1	1	1	1	1	#	...
b)	...	#	#	X	#	#	#	#	#	#	#	#	...

Fig. 3. Example of primality problém encoded as the data tapes

The settings of the TM which were used in relation with above-mentioned problems are:

Problem	Q	Σ	Γ	q_0	B	F
1	$\{q_1 \dots q_{11}\}$	$\{'1'\}$	$\{'\#','1'\}$	q_1	$'\#'$	$\{q_{11}\}$
2	$\{q_1 \dots q_{14}\}$	$\{1','X'\}$	$\{'\#',' 1','X'\}$	q_1	$'\#'$	$\{q_{14}\}$
3	$\{q_1 \dots q_{25}\}$	$\{1','X'\}$	$\{'\#',' 1','X'\}$	q_1	$'\#'$	$\{q_{25}\}$

Tab. 2. Settings of the TM used for processing the problems

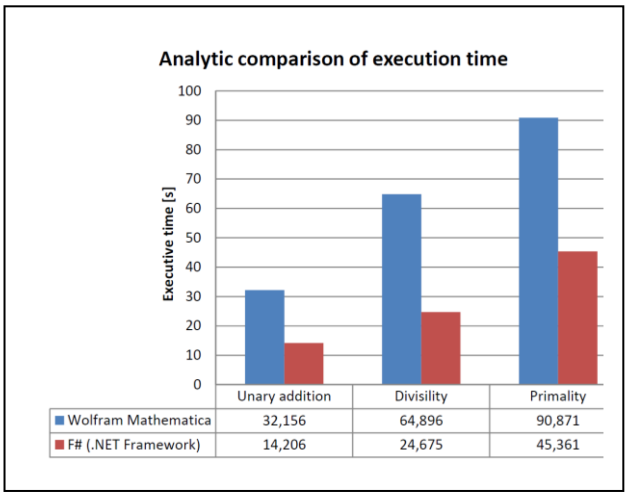
In the case of unary addition (problem 1), the initial position of the TM's head was at position before first input symbol. The TM's head position when processing problem 2 (divisibility) was at blank symbol between both numbers. The third problem (primality) used the head position of the TM at the blank symbol before the first input symbol.

2.3 Backgrounds of analysis

The analysis was focused on execution time thus the rate of successfulness of estimation of the programming the Turing machine is not important (it depends on settings of DE especially). Each of test in the form of evolutionary-programming the Turing machine in order to processing above-mentioned problems was severaly repeated. The execution time which is presented was counted as average of measured values of each run.

3. RESULTS

As can be seen in Fig. 4, when F# language and .NET Framework are used as implementation environment, the execution time is significantly lower. In the case of algorithms implemented in Wolfram Mathematica, the slowdowns are about 55.8 % (unary addition), 61.9 % (the divisibility problem) and 50 % (the primality problem).



	Unary addition	Divisility	Primality
🟦 Wolfram Mathematica	32,156	64,896	90,871
🟥 F# (.NET Framework)	14,206	24,675	45,361

Fig. 4. Comparsion of execution time

4. CONCLUSION

The analysis described in this paper proved that it is higly important to consider suitable environment for implementation of algorithms. If execution time of algorithms which were used in (Kouril & Zelinka, 2010) and implemented in Wolfram Mathematica is compared to algorithms implemented in F# language and .NET Framework 4.0, the latter is significantly faster. On the basis of the analysis, Microsoft .NET Framework will be considered as default implementation environment for evolutionary-programming the Turing machine.

5. ACKNOWLEDGEMENTS

This paper is a part of research No. IGA/42/FAI/10/D supported by the Internal Grant Agency of Tomas Bata University and by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

6. REFERENCES

Hopcroft, J.E. et al. (2000). *Introduction to Automata Theory, Languages and Computation*, 2nd s.l., Pearson Education. ISBN 0-201-44124-1

Kouril, L. & Zelinka, I. (2010). Evolutionary-Estimated Programming the Turing Machine by Differential Evolution, *Proceedings of 16th International Conference on Soft Computing MENDEL 2010*, Brno, Czech Republic, ISBN 978-80-214-4120-0, pp. 41-48

Lampinen, J. & Zelinka, I. (1999). Mechanical Engineering Design Optimization by Differential Evolution, In: *New Ideas of Optimization*, 1st London, McGraw-Hill, ISBN 007-709506-5

Zelinka, I. et al. (2008). *Evolucni vypocetni techniky – principy a aplikace*, BEN – technicka literatura, ISBN 80-7300-218-3, Praha

*** (2011a) http://fsai.codeplex.com – F# Artificial Intelligence Library – An Inplementation of Selected AI Methods, *Accessed on: 2011-09-25*

*** (2011b) http://www.fsharp.net – Microsoft F# Developer Center, *Accessed on: 2011-09-20*