Time Requirements of Optimization of a Genetic Algorithm for Road Traffic Network Division Using a Distributed Genetic Algorithm

T. Potuzak

Abstract This paper describes the optimization of a dividing genetic algorithm (DGA). It is used for division of road traffic networks into sub-networks of a distributed road traffic simulation. The optimization is performed by finding optimal settings of the DGA parameters using a distributed optimizing genetic algorithm (distributed OGA). Since the distributed OGA is expected to be extremely time-consuming, the paper is focused on a determination of the total time necessary for the OGA computation. It is determined, performing tests, that the OGA can be completed in range of days at least for lower numbers of OGA generations on a distributed computer consisting of nearly 100 processor cores.

1 Introduction

The computer road traffic simulation is an important tool for analysis, control, and design of road traffic networks. Such simulation can be very time consuming, especially for large road traffic networks (e.g. large cities or entire states). Therefore, many road traffic simulators have been adapted or designed for distributed computing environment where combined computing power of multiple interconnected computers (nodes) is used for faster execution of the simulation. In this case, the simulated road traffic network must be divided into sub-networks, whose simulations are then performed on particular nodes of the distributed computer as processes. The quality of the division influences the resulting performance of the entire distributed simulation. Hence, two important issues should be considered during the division—the sub-networks load-balancing and the inter-process communication minimization (Potuzak 2011).

T. Potuzak (✉)
Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Plzen, Czech Republic
e-mail: tpotuzak@kiv.zcu.cz

Z. S. Hippe et al. (eds.), *Issues and Challenges in Artificial Intelligence*, Studies in Computational Intelligence 559, DOI: 10.1007/978-3-319-06883-1_13,

During our previous research, we have developed a method for road traffic network division, which considers both mentioned issues (Potuzak 2011). This method utilizes a genetic algorithm for multi-objective optimization of the resulting road traffic network division. Although the results of the proposed method are reasonable (Potuzak 2012a) they are not optimal, especially for large road traffic networks (Potuzak 2013a). The probable cause is the suboptimal settings of the parameters of the genetic algorithm that are set manually based on preliminary testing result. These settings can be tuned in order to achieve better performance of the division method.

In this paper, we discuss the time requirements of a distributed genetic algorithm for optimization of the parameters settings of the genetic algorithm for road traffic network division. To avoid confusion in the following text the genetic algorithm of optimization is referred as optimizing genetic algorithm (OGA). The genetic algorithm for road traffic network division (which is optimized using OGA) is referred as dividing genetic algorithm (DGA).

2 Basic Notions

In order to make further reading more clear the genetic algorithms and the road traffic network division are described first.

2.1 Genetic Algorithms Description

A genetic algorithm is an evolutionary algorithm (Poli et al. 2008), which mimics natural genetic evolution and selection in nature in order to solve a specific problem. Since their development in 1975, the genetic algorithms have been widely used for solving of optimization (including multi-objective optimization) and searching problems in many domains (Farshbaf and Feizi-Darakhshi 2009). The basic notions of a typical genetic algorithm are described in following paragraphs.

Using the genetic algorithm for solving of a problem the representation of the solution of the problem must be determined first. Each solution is considered an *individual* and it is usually represented by a vector of binary or integer values. The meaning of the particular values depends on the solved problem. A predetermined number of individuals are then (most often) randomly generated in order to form the *initial population* (Menouar 2010).

For each individual of the initial population, the so-called *fitness value* is calculated using a *fitness function*. The fitness value is an objective assessment of the quality of each individual from the point of view of the solved problem. The better the solution is, the higher its fitness value should be. So, the fitness function is the only part of the genetic algorithm requiring the knowledge of the solved problem (Menouar 2010). Based on the fitness value, a set of individuals are selected to be "parents" of a new generation.

The new generation is created using the selected parents and the *crossover* and *mutation* operators. The crossover uses (most often) two parents to produce (most often) two descendants using mutual exchange of the values of the parents' vectors. The mutation performs random changes of the values in the descendants' vectors (Poli et al. 2008). Once the new generation is created, the fitness values are calculated for all individuals and the process repeats until a stop condition is fulfilled or a preset number of generations is created (Potuzak 2011).

2.2 Road Traffic Network Division Description

As it is mentioned in the Sect. 1 the road traffic network must be divided into required number of sub-networks prior the execution of its simulation in a distributed environment. The resulting sub-networks are then simulated as processes on particular nodes of the distributed computer (a sub-network per node). Since, even with the simulated road traffic network divided the vehicles still need to pass among the sub-networks. Thus, the simulation processes are interconnected by communication links enabling transfer of vehicles between the neighboring sub-networks in the form of messages (Potuzak 2012a). The inter-process communication is also necessary for synchronization of the simulation processes, which ensures that the simulation time of all processes is the same at the same moment (Potuzak 2012a).

There are two issues, which should be considered during the road traffic network division—the sub-networks load-balancing and the inter-process communication minimization. The load-balancing is necessary to achieve similar speeds of the simulation processes and, consequently, to minimize the waiting of the faster processes on the slower processes (Grosu et al. 2008; Potuzak 2012a). If the target distributed computer is a homogenous cluster (i.e. with nodes of the same computing power), the sub-networks should have similar loads (i.e. similar numbers of vehicles moving within them). If the target distributed computer is a heterogeneous cluster (i.e. with nodes of different computing power), the loads of the particular sub-networks should correspond to the computing powers of the nodes, on which the sub-networks will be simulated (Potuzak 2013a).

The communication minimization is necessary, because it is very slow in comparison to the remainder computations of the distributed simulation computations. The communication can be diminished by reducing the number of vehicles transferred among the sub-networks. This, in turn, can be achieved by minimization of the number of traffic lanes interconnecting the sub-networks (Potuzak 2013b).

3 Road Traffic Network Division Using DGA

The method for road traffic network division, which we have developed, considers both mentioned issues (see Sect. 2.2). There are two versions of the method—for homogenous and for heterogeneous clusters. The versions differ only in the

load-balancing, where the road traffic network is divided into sub-networks with the similar loads for the homogeneous clusters and into sub-networks with the loads ratio corresponding to the computing power ratio of the particular nodes for the heterogeneous clusters (Potuzak 2013a).

Regardless the version, the method utilizes a (usually) less-detailed road traffic simulation for assigning of the weights to particular traffic lanes. These weights correspond to the numbers of vehicles moving within the lanes during the simulation run. It is possible to use a macroscopic, a mesoscopic, or a microscopic simulation. However, the macroscopic simulation is usually used, since all the simulations give similar results and the macroscopic simulation is the fastest one (Potuzak 2012a). Once the weights are assigned to the traffic lanes, the road traffic network is considered as a weighted graph. The crossroads are then acting as nodes and the sets of traffic lanes interconnecting neighboring crossroads acting as edges with weights equal to the sum of weights of all traffic lanes of each set (Potuzak 2013b). The road traffic network (i.e. the weighted graph) is then divided using the dividing genetic algorithm (DGA) into required number of load-balanced sub-networks of minimized number of edges (and, consequently, traffic lanes) interconnecting them (i.e. divided traffic edges/lanes) (Potuzak 2012a, 2013a).

3.1 Dividing Genetic Algorithm Description

The DGA is a standard genetic algorithm (see Sect. 2.1). Each individual represents a single assignment of crossroads to particular sub-networks in the form of a vector of integer values. The length of the vector corresponds to the total number of crossroads in the divided road traffic network. Each value then represents the sub-network to which the corresponding crossroad is assigned (Potuzak 2011).

The initial population of 90 individuals is randomly generated. So, the crossroads are randomly assigned to the particular sub-networks (Potuzak 2012a). The fitness value of each individual is calculated using the fitness function consisting of two parts—the *equability* and the *compactness*. The equability represents the load-balancing of the sub-networks and it is the only part of the DGA, which is different for homogenous and heterogeneous clusters. The compactness represents the minimization of the number of divided edges (and, consequently, traffic lanes) and is the same for both types of clusters. The fitness function can be then expressed as:

$$F_{DGA} = r_E \cdot E + (1 - r_E) \cdot C \tag{1}$$

where a F_{DGA} is the fitness value, E is the equability, C is the compactness, and r_E is the equability ratio determining, which part (the equability E or the compactness C) of the fitness function is more important. Different values of the r_E can be convenient in different situations. Usually, the r_E is set to 0.25.

Once the fitness values are calculated for all individuals, ten individuals are selected to be parents of the new generation using truncation selection (see Fig. 1a), which means than the individuals with the highest values from entire

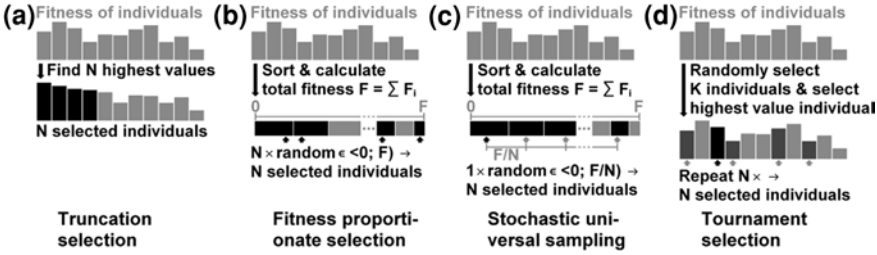

Fig. 1 Comparison of the implemented selection methods

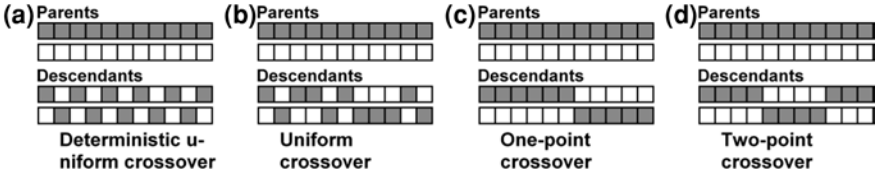

Fig. 2 Comparison of the implemented crossover methods

population are selected (Bäck 1996). Using all possible pairs formed from the selected individuals and the deterministic uniform crossover (see Fig. 2a), the new generation of 90 individuals is created. Using this crossover, every second value is exchanged between the two parents' vectors to form two descendants. Each descendant is then mutated using a preset number of random changes of its vector. This way, a new generation is created. The entire process repeats for the preset number of generations (up to 100,000) (Potuzak 2013b).

3.2 Implemented Improvements of DGA

In order to achieve a higher speed of the DGA, its implementation is recently refined. A new data structure is created to enable faster calculation of the fitness values and the utilization of integer instead of real numbers is employed where possible. This leads to a substantial increase of the DGA speed due to reducing its computation time to roughly 30 % of its original implementation.

Moreover, three other selection methods and three other crossover methods are implemented in order to include them to the OGA optimization of the DGA. A specific selection and crossover methods can be set prior to the DGA execution. The selection methods include truncation selection, fitness proportionate selection (Bäck 1996), stochastic universal sampling (Baker 1987), and tournament selection (Xie and Zhang 2013). The difference between the selection methods is depicted in Fig. 1.

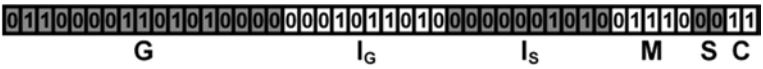

Fig. 3 An example of the OGA individual

The crossover methods include deterministic uniform crossover (see Sect. 3.1), uniform crossover, one-point crossover, and two-point crossover (Ahmed 2010). The difference in the crossover methods is depicted in Fig. 2.

4 DGA Optimization Using OGA

In order to find optimal settings of the parameters of the DGA, the optimizing genetic algorithm (OGA) is used. There are several parameters of the DGA to optimize—the number of generations, the number of individuals in the generation, the number of selected individuals, and the number of mutations per individual. Moreover, the optimal combination of the selection and crossover methods (four and four options, respectively—see Sect. 3.2) should be determined. All the mentioned parameters can significantly influence the quality of road traffic network division and the speed of the division method (Potuzak 2013b).

A parameter, which will not be optimized using the OGA is the equability ratio (r_E), which determines whether the load-balancing of the sub-networks or the minimization of the number of divided traffic lanes should be more important during the division (see Sect. 3.1). Since the requirements can vary in different situations, it would be difficult to find a single optimal value (Potuzak 2013b).

4.1 Optimizing Genetic Algorithm Description

The utilization of the OGA for optimization of the DGA seems to be a viable approach, since the genetic algorithms are generally convenient for optimization problems (Farshbaf and Feizi-Darakhshi 2009). The OGA individual representation must incorporate all the parameters, which shall be optimized (see Sect. 4). These parameters can be expressed as nonnegative integer numbers of different sizes. So, the OGA individual is represented by a binary vector with several multi-bit parts. Each part represents a single DGA parameter to optimize encoded using the unsigned magnitude representation. More specifically, each individual consists of 46 bits—17 bits for the number of generations (G), 10 bits for the number of individuals in a generation (I_G), 10 bits for the number of selected individuals (I_S), 5 bits for the number of mutations per individual (M), 2 bits for the type of selection (S), and 2 bits for the type of crossover (C) (Potuzak 2013b). The example is depicted in Fig. 3.

The initial population of the OGA consists of 90 randomly generated individuals (similarly to DGA). For all individuals, the fitness value is calculated. For an OGA individual, it is necessary to perform the complete DGA on a road traffic network

with parameters set according to the OGA individual. Since it is necessary for the OGA optimization to be universal and not solely for one road traffic network, three road traffic networks of different sizes divided into three different numbers of sub-networks will be used. These networks are regular square grids of 64, 256, and 1,024 crossroads (86, 326, and 1,267 km of total traffic lanes length, respectively). They are divided into 2, 4, and 8 sub-networks. Originally, the division into 16 sub-networks was considered as well (Potuzak 2013b). However, for the road traffic networks of the given sizes, the division into 16 sub-networks is quite improbable to be convenient, because the resulting sub-networks will be too small.

The OGA fitness value is then calculated using the maximal achieved DGA fitness values of all combinations of the divided road traffic networks and the number of sub-networks. So, the fitness function can be expresses as:

$$F_{OGA} = \frac{\sum_{i=1}^{3} \sum_{j=1}^{3} F_{ij}^{\max}}{9}, \tag{2}$$

where F_{OGA} is the OGA fitness function and $F_{ij}^{\max}$is the maximal achieved DGA fitness value for *i*th road traffic network and *j*th number of sub-networks.

Once the fitness values are calculated for all OGA individuals, ten individuals are selected to be parents of the new generation. The truncation selection is used (see Fig. 1a), which means that the individuals with the highest fitness values are selected. For the creation of the descendants, the deterministic uniform crossover (see Fig. 2a) of all possible pairs of two parents is used. Each descendant can be mutated using random but limited number of random changes of its bits. Once the new generation is created, the fitness values are calculated for all OGA individuals and the process repeats until the preset number of generations is created. So, the OGA is very similar to the original DGA.

4.2 Distributed OGA Description

Because the OGA is expected to be extremely computation intensive (Potuzak 2012b, 2013b), only its distributed implementation seems to be feasible. Still a distributed computer with large number of nodes will be required to keep the OGA computation time in acceptable limits. For the OGA, we can utilize two classrooms, which are at our disposal at Department of Computer Sciences and Engineering of University of West Bohemia (DSCE UWB). Each classroom contains twelve desktop PCs. Each PC incorporates Intel i5-2400S Quad-Core processor at 3.1 GHz, 8 GB RAM, and 250 GB HDD. The PCs are used for the education purposes during the working days, but are usually idle or deactivated during the late afternoons, nights, and weekends. As these times, they can be readily used for the OGA computations (Potuzak 2013b).

For the distributed OGA, the well-known farmer–worker paradigm is used. So, there is one control process (farmer) and a number of working processes

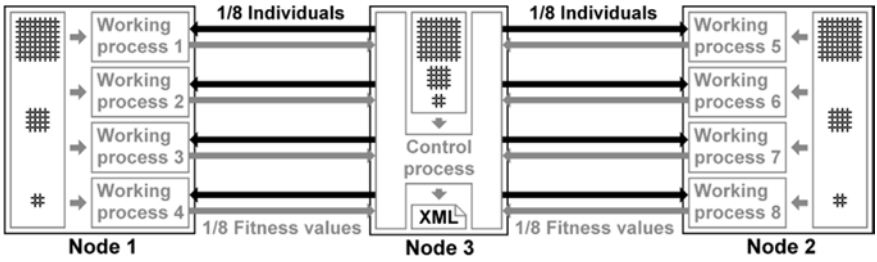

Fig. 4 The OGA communication scheme for one control and eight working processes

(workers). The OGA is implemented in the DUTS Editor system (developed at DSCE UWB). This system with all three road traffic networks (see Sect. 4.1) must be installed on each computer, which will participate on the OGA computation. It is possible to run either the control process or the working process. It is convenient, on a multi-core node (computer), to run number of working processes corresponding to the number of processor cores. Each working process is connected by a bidirectional communication link to the control process for the message passing.

Assume now that the distributed OGA is performed on three quad-core nodes, one of them hosting the control process while two other hosting four working processes each (see Fig. 4). The OGA computation is then performed as follows. Each working process loads all three road traffic networks. The control process loads them as well and performs the macroscopic simulation on each of them in order to obtain weights of the traffic lanes (i.e. input for the DGA—see Sect. 3). These weights are sent to all working processes, which assign them to the corresponding traffic lanes of the loaded road traffic networks. The control process then creates the initial OGA population and distributes the individuals uniformly among the working process. Each working process then calculates the OGA fitness values of the received individuals by performing nine DGA runs per OGA individual (i.e. using three road traffic networks divided into three different numbers of sub-networks—see Sect. 4.1). The OGA fitness values are then sent to the control process. Once the control process receives the fitness values of all OGA individuals, it performs the selection, crossover, and mutation and creates a new OGA generation.

Its OGA individuals are again distributed to the working processes and the entire process repeats for the preset number of generations. The inter-process communication scheme of the OGA is depicted in Fig. 4.

Because the fitness values calculation is by far the most computation-intensive part of the OGA and only relatively short vectors of integer or real values are transferred two times per generation between the control and each working process, the time necessary for inter-process communication is negligible, which is a very convenient feature for a distributed computation. Nevertheless, the OGA computation time is in range of days and weeks (see Sect. 5) and the classrooms intended for the OGA are regularly used for education purposes. Hence, a single continuous computation of the OGA is out of question. Instead, it is possible for

the control process to store the weights of the traffic lanes and the current generation in a XML file during the computation. So, the computation can be ended whenever necessary and then resumed using the data in the XML file.

5 Tests and Results

Currently, an implementation of the distributed OGA is nearly finished. In order to determine the time necessary for its execution a set of tests have been performed using one processor core of one node (computer) of the distributed computer, on which the distributed OGA is performed (see Sect. 4.2).

The set of tests has been focused on the time necessary for the homogenous and heterogeneous DGA runs in dependence on the settings of its three optimized parameters. The settings of these parameters—the number of DGA generations, the DGA selection type, and the DGA crossover type (see Sect. 4) can significantly influence the time necessary for the DGA run, and consequently, the distributed OGA execution, which utilize nine DGA runs for the computation of the fitness value of a single OGA individual (see Sect. 4.1). The tests have been performed for all three road traffic networks (see Sect. 4.1), 1,000, 10,000, and 100,000 DGA generations, and all combinations of the DGA selection and DGA crossover types. All road traffic networks have been divided only into four sub-networks, since the number of sub-networks has negligible effect on the DGA computation time (Potuzak 2013b). The results for the DGA for homogeneous cluster are summarized in Table 1. Each value is averaged from ten attempts. The results for the DGA for heterogeneous clusters are very similar and differ in range of several percents. Therefore, from now on, we will not distinguish between these two versions of the DGA.

The results depicted in Table 1 can be used for the determination of the mean computation time of the fitness value of a single OGA individual using one processor core depending on the number of DGA generations. Following the calculation of the OGA fitness function (see Eq. 2) and using the last row of Table 1, this mean time can be calculated as:

$$\overline{T_G^{OGA}} = 3 \cdot \left(\overline{T_{64,G}^{DGA}} + \overline{T_{256,G}^{DGA}} + \overline{T_{1,024,G}^{DGA}} \right), \tag{3}$$

where $\overline{T_G^{OGA}}$ is the mean computation time necessary for the calculation of the OGA individual for G DGA generations, and $\overline{T_{X,G}^{DGA}}$ are the mean times necessary for the DGA for G generations and X crossroads of the divided road traffic network (i.e. values in last row of Table 1). It should be noted that the distributed OGA is intended to utilize all four processor cores of each node of the distributed computer. It has been determined in (Potuzak 2012b) that it is possible for the price of 7 % slowdown in comparison to the utilization of only one core. The mean computation time of the fitness value of a single OGA individual for different numbers of DGA generations is summarized in Table 2.

Table 1 Time necessary for the DGA for homogeneous clusters

Crossroads		64			256			1,024		
Generations		10^3	10^4	10^5	10^3	10^4	10^5	10^3	10^4	10^5
S	C	DGA computation time (s)								
TR	DU	0.2	2.2	22.3	0.7	6.2	58.4	2.8	24.2	211.8
	U	0.3	2.8	28.8	0.9	8.5	81.4	3.7	33.4	325.4
	1P	0.2	2.2	22.4	0.7	6.1	58.3	2.7	23.7	207.2
	2P	0.2	2.2	22.7	0.7	6.2	59.6	2.8	24.1	215.4
FP	DU	0.2	2.6	26.9	0.8	7.6	75.8	2.8	27.8	764.0
	U	0.3	3.2	33.3	1.0	10.1	100.9	3.8	37.8	394.3
	1P	0.2	2.6	26.4	0.7	7.5	74.5	2.8	27.2	271.0
	2P	0.3	2.6	26.8	0.8	7.5	75.3	2.8	27.6	273.9
SUS	DU	0.2	2.6	26.4	0.8	7.5	75.1	2.8	27.7	275.2
	U	0.3	3.2	32.8	1.0	10	100.3	3.8	37.8	393.7
	1P	0.2	2.5	26.1	0.7	7.4	73.9	2.8	27.2	270.2
	2P	0.2	2.5	26.3	0.7	7.5	74.7	2.8	27.4	273.1
TO	DU	0.2	2.2	22.6	0.7	6.2	58.9	2.8	24.4	213.6
	U	0.3	2.8	28.8	0.9	8.6	82.2	3.7	34.0	326.2
	1P	0.2	2.1	22.3	0.7	6.0	57.8	2.8	23.8	209.1
	2P	0.2	2.2	22.6	0.7	6.1	58.5	2.8	23.9	211.8
Average		0.2	2.5	26.1	0.8	7.4	72.8	3.0	28.2	302.2

Selection (S) types: *TR* truncation selection, *FP* fitness proportionate selection, *SUS* stochastic universal sampling, *TO* tournament selection. Crossover types: *DU* deterministic uniform crossover, *U* uniform crossover, *1P* one-point crossover, *2P* two-point crossover

Table 2 The mean computation time of the fitness value of a single DGA individual

	Without 7 % slowdown	With 7 % slowdown
DGA generations count	Computation time (s)	
10^3	12.0	12.8
10^4	114.3	122.3
10^5	1,203.3	1,287.5
Average	443.2	474.2

The mean computation time of the fitness value of a single OGA individual for mean number of generations increased by 7 % (see most right cell in last row of the Table 2) can be designated the *OGA individual computation time*. Comparing the new results depicted in Table 2 with the results described in Potuzak (2013b), the new OGA individual computation time is only roughly 17 % of the OGA individual computation time described in Potuzak (2013b). The reasons of this speedup are the optimization of the DGA (see Sect. 3.2) and the change of the calculation of the OGA fitness function (the division into 16 sub-networks is no longer considered—see Sect. 4.1).

Using the OGA individual computation time and knowing there are 90 individuals per OGA generation, it is possible to determine the total computation time of the OGA. Considering that the distributed computer for the OGA would consist of

Table 3 Total computation time of the OGA depending on the number of OGA generations

OGA generations	Computation time (h)	Computation time (days)
10^3	123.496	5.146
10^4	1,234.958	51.457
10^5	12,349.583	514.566

24 nodes (i.e. computers from both considered classrooms—see Sect. 4.2), there are $24 \times 4 = 96$ processors' cores available for the OGA working processes. The OGA control process can reside in another computer at our disposal. The OGA computation time in dependence on the number of OGA generations is summarized in the Table 3.

As can be seen in the Table 3, the OGA computation time ranges from 5 to 515 days, depending on the number of OGA generations. So, at least the lower numbers of OGA generations seem to be feasible. The time necessary for the macroscopic simulation used for the obtaining of the weights of traffic lanes is not included in the OGA computation time. The reason is that the macroscopic simulation is performed only once and is very fast (several seconds for the largest road traffic network). Similarly, the time necessary for inter-process communication is not included, because the communication is performed rarely during the OGA computations (two messages per OGA generation per working process).

6 Conclusions

In this paper, we described the optimization of the genetic algorithm for road traffic network division (DGA) using a distributed optimizing genetic algorithm (distributed OGA). We focus on the determination of the total time necessary for the distributed OGA computation. Depending on the number of OGA generations, this computation time ranges from 5 to 515 days on 24 quad-core computers. So, the OGA seems to be feasible at least for lower numbers of OGA generations.

Currently, the OGA implementation is nearly complete. The debugging and preliminary testing, which are crucial for such a long computation, are now ongoing. The further step in our research is to perform the OGA in order to find optimal settings of the DGA parameters.

References

Ahmed ZH (2010) Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. Int J Biom Bioinform 3(6):96–105

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, New York

Baker JE (1987) Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms and their application, pp 14–21

Farshbaf M, Feizi-Darakhshi M (2009) Multi-objective optimization of graph partitioning using genetic algorithms. In: 2009 third international conference on advanced engineering computing and applications in sciences, Sliema, pp 1–6

Grosu D, Chronopoulos AT, Leung MY (2008) Cooperative load balancing in distributed systems. Concurr Comput Pract Exp 20(16):1953–1976

Menouar B (2010) Genetic algorithm encoding representations for graph partitioning problems. In: 2010 international conference on machine and web intelligence, Algiers, pp 288–291

Poli R, Langdon WB, McPhee NF (2008) A field guide to genetic programming. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk (with contributions by Koza JR)

Potuzak T (2011) Suitability of a genetic algorithm for road traffic network division. In: KDIR 2011—proceedings of the international conference on knowledge discovery and information retrieval, Paris, pp 448–451

Potuzak T (2012a) Methods for division of road traffic networks focused on load-balancing. Adv Comput 2(4):42–53

Potuzak T (2012b) Issues of optimization of a genetic algorithm for traffic network division using a genetic algorithm. In: Proceedings of the international conference on knowledge discovery and information retrieval, Barcelona, pp 340–343

Potuzak T (2013a) Methods for division of road traffic network for distributed simulation performed on heterogeneous clusters. Comput Sci Inf Syst 10(1):321–348

Potuzak T (2013b) Feasibility study of optimization of a genetic algorithm for traffic network division for distributed road traffic simulation. In: Proceedings of the 6th international conference on human system interaction, pp 372–379

Xie H, Zhang M (2013) Parent selection pressure auto-tuning for tournament selection in genetic programming. IEEE Trans Evol Comput 17(1):1–18