TRANSNAV
http://www.transnav.eu
the International Journal
on Marine Navigation
and Safety of Sea Transportation
Volume 7
Number 2
June 2013

DOI: 10.12716/1001.07.02.17

Distributed Evolutionary Algorithm for Path Planning in Navigation Situation

R. Śmierzchalski, Ł. Kuczkowski, P. Kolendo & B. Jaworski
Gdansk University of Technology, Gdansk, Poland

ABSTRACT: This article presents the use of a multi-population distributed evolutionary algorithm for path planning in navigation situation. The algorithm used is with partially exchanged population and migration between independently evolving populations. In this paper a comparison between a multi-population and a classic single-population algorithm takes place. The impact on the ultimate solution has been researched. It was shown that using several independent populations leads to an improvement of the ultimate solution compared to a single population approach. The concept was checked against a problem of maritime collision avoidance.

1 INTRODUCTION

One of the ways of solving optimization tasks is using an evolutionary algorithm with many populations. Multi-population distributed evolutionary algorithm (DGA) is a one of such programs, first presented in (Tanese 1989a; Tanese 1989b) as a method of parallel calculations on a simple genetic algorithm (Goldberg 1989). The proposed solution divided global population into several subpopulations. Afterwards a simple generic algorithm was applied to each subpopulation using a single processor, which computed a single evolution cycle. In every generation few individuals were exchanged between subpopulations (in a synchronous or asynchronous way). That research showed the genetic algorithm defined in this way provided better results in terms of computation time and results quality in comparison to a single population algorithm. In (Forrest and Mitchell 1991) Forrest and Mitchell repeated the research on the multi-population algorithm using Tanese's functions. Their work proven that Tenese's results could be connected to the fitness function used in her research and the parallelization of the calculations. In (Belding 1995) Theodore C. Belding continued the work on DGA using Royal Road functions, called R_1, R_2, R_3, R_4 which were presented in detailed in (Mitchell, Forrest and Holland 1992; Forrest and Mitchell 1993; Mitchell and Holland 1993; Mitchell, Holland and Forrest 1994). Author has shown that in the case of complex tasks (R_3, R_4 functions), the multi-population algorithm reached far better results (in comparison to a single population algorithm). In other cases the results were comparable.

One of the examples of an application of a multi-population evolutionary algorithm is a two step variation used for planning networks of parallel connected computers (Cochran, Horng and Fowler 2006). In the first step algorithm optimizes using a weighed fitness function and in the second, based on the results from the previous step, subpopulations are being created which evolve in parallel. The paper compares the achieved results with solution given by other methods. The authors shown the DGA provided better results. In (Martikainen and Ovaska 2006) a hierarchical two population evolutionary algorithm

operating based on a base and elite populations was presented. The elite population consisted of the best individuals and the base one contains individuals of lesser fitness score. Both populations are evolved using different methods i.e. mutation and crossover probability. The paper shows the superiority of a multi-population variation over the single population while using the same number of generations.

Multi-population distributed evolutionary algorithm can work both in parallel and sequential modes. Different variations are used to solve complex optimization tasks. In (Gehring and Bortfeldt 2002) Gehring and Bortfeldt showed multi-population algorithm that calculated the optimal container loading sequence. The DGA modification used by them provided results better than those of a single population algorithm. There are many publication such as (Whitley 1997; Cantu-Paz 1999; Martikainen 2006) that show an overview currently used variations of the multi-population algorithms. Different ways of individual migration and dedicated genetic operators that allow for an exchange of individuals between populations are shown.

One of the tasks that the multi-population algorithm can be used with is remote control of a mobile object (i.e. mobile robot or autonomous overwater ships). It consists in leading an object from a starting position to its destination or the operation (mission) area. In order to do this, an optimal (by the criteria of i.e. length) path has to be plotted. This path has to avoid obstacles: static and dynamic constraints of the environment. The dynamic constraints can present themselves as other moveable objects travelling along their own trajectory with certain speed. The problem can be examined in two modes: off-line and on-line. The off-line mode of path planning is carried out in an environment where the movement parameters of other dynamic objects are known and are constant. The on-line mode takes into account the changes in the environment and the uncertainty of the movement of other objects. As a result of this, a constant control over the environment's changes and the other objects parameters is required. In the event of changes, a modification of previously plotted path takes place.

The problem was reduced to an optimization tasks with static (islands, forbidden areas) and dynamic (other ships, changeable weather conditions) constraints (Śmierzchalski 1997). In order to solve the presented problem an adaptive evolutionary method was used (Goldberg 1989; Michalewicz 1996), which operated based on the Evolutionary Planner/Navigator (νEP/N++) (Xiao and Michalewicz 1999). This algorithm uses the evolutionary algorithm library GALib (Wall 1996). Based on the available library components a multi-population distributed evolutionary algorithm (Tanese 1989a; Tanese 1989b; Belding 1995) was examined and its results were compared with a single population version.

This article presents a multi-population distributed evolutionary algorithm used in the problem of maritime path planning. Its organized so that in the chapter two the maritime path planning problem is presented. The evolutionary method and the multi-population variant is described in chapter three and four. Chapter five defines the simulation environment, while chapter six presents the results. Chapter seven concludes the paper.

2 PATH PLANNING FOR A COLLISION SCENARIO

The problem of collision avoidance consists of plotting a path P, as a part of given route, which the mobile object covers of the initial position (start point) (x_0, y_0) to the actual destination point (x_e, y_e). The path is composed of a linear segment sequence p_i $(i = 1,...,n)$, interconnected by the turning points (x_i, y_i). The start and destination points are chosen by the operator. Taking the above into account, path P is feasible (belongs to the safe paths set): if every segment p_i $(i = 1,\ldots,n)$ remains in the area of the environment and it does not cross with any dynamic or static constrain. The paths that do not meet this requirement are considered unfeasible (Cochran, Horng and Fowler 2003).

According to the collision avoidance rules, when an encountered object is in the area of observation and if the object's course crosses own path in an unsafe distance, we consider the object to be on a potential collision course (target 1, point p_k, Fig 1).

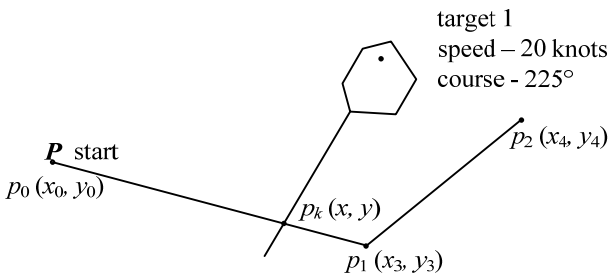


Figure 1. Potential collision scenario.

The safe distance from the own ship depends on the adopted collision danger level (usually a distance of 5-8 nautical miles in front of the bow and 2-4 nautical miles astern, depending on the size, type of vessel and the ratio of own and target ships' speed) (Śmierzchalski 1998). The distance of the closest approach and the time needed to reach this distance has to be also considered. The collision danger condition is reduced to detecting if (Lenart 1986):

$$D_{\min} \le D_{kr} \tag{1}$$

$$T_{D\min} \le T_{kr} \tag{2}$$

where:

D_{min} – closest approach distance,

T_{Dmin} – time to achieve closest approach distance,

D_{kr}, T_{kr} – critical values of D_{min} and T_{Dmin} set by the system's operator.

In case of the collision avoidance task, the objects the present a collision danger are interpreted as mobile objects travelling with certain speed and course.

According to the movement concept, the own ship should cover the given route in a given time. On the other hand it should move safely along the given path in order to avoid objects that present a thread of collision. Path planning for an object in a collision scenario has to be a compromise between a deviation from the given route and the safety of the travel. Thus the problem is defined a multi-criteria optimization task, which includes the safety and the economy of the movement. The total cost of a path's fitness considers the safety cost and the cost of moving along the given path. The safety cost in calculated based on the distance from the constraints and the cost of travel takes into account: total path's length P, the maximum turn angle between the segments p_i and the time needed to cover the route (Śmierzchalski 1998).

3 SINGLE POPULATION ALGORITHM WITH PARTIALLY REPLACED POPULATION.

Evolutionary algorithm processes a set of solutions called the population. The environment on which it operates is defined based on the task pending (fitness function, constrains). Each individual (single population member) represents a different problem's solution. Based on the fitness function each individual is assigned a parameter called the fitness score. The fitness score determines the quality of the solution represented by each member. In the moment of algorithm's initialisation, the initial conditions are set. Each individual is being randomly generated. Afterwards the following steps are executed: reproduction, genetic operations, evaluation and succession (Fig. 2). In reproduction phase a temporary population is created to which random individuals from the base population are being copied. It is possible to introduce more than one copy of any individual. The greater the fitness score, the greater the chance of selecting particular member of the population. In the next step the temporary population is processed by the genetic operations, which modifies individuals. The νEP/N++ program has the following genetic operator build-on: hard mutation, soft mutation, adding a gene, gene removal, gene position swap, smoothing and single point crossover (Śmierzchalski 1997). The set of solutions calculated in this way is called the child population, which is evaluated. The succession phase creates a new base population. In the single population algorithm with partially replaced population the new population gathers individuals from the child population and the old base population. The amount of individuals added to the base population is defined by the user. The individual passes over only if its fitness score is greater than the fitness score of the worst individual of the old base locations. Those algorithm's phases are repeated in a loop until the termination condition is met (The algorithm runs for a specific number of generations or a desired fitness score has been achieved). In the GALib library the algorithm like this is called a Steady-State GA.

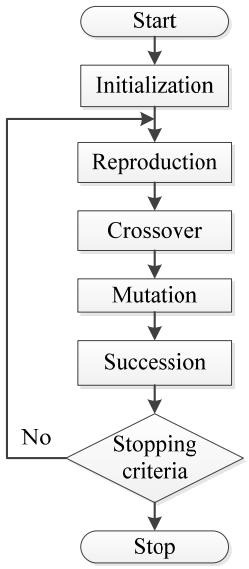


Figure 2. Single population algorithm diagram

4 MULTI-POPULATION ALGORITHM

Multi-population algorithm provides the ability of running simultaneous, independent evolutions of many populations. To achieve this νEPN++ uses the partial exchange evolutionary algorithm. General diagram of the multi-population algorithm is shown on Fig. 3. In the first phase of the algorithm an initialization of user defined number of populations of randomly generated individuals takes place. After that the evolution process is applied to each population separately. This process, similar as in the single population algorithm, consists of the following steps: reproduction, crossover, mutation, individual evaluation and the new base population individual selection. When the evolution cycle of all populations is completed, the migration and succession of the superior population takes place.

The considered algorithm utilizes the stepping-stone migration (Fig. 4a) (Tanese 1989a; Tanese 1989b). From each population, starting with the zero one, a specific number of individuals is being passed on to the neighboring (next in the set) population. This process is repeated until the last population won't export its individuals to the zero population.

In the next step, the succession of the superior population takes place (Fig. 4b). This population stores the best solution and is not undergoing an evolution process. From each inferior population (a population affected by the evolution process) a certain number of best adapted individuals is selected. Afterwards the algorithm verifies if any of the chosen individuals has fitness score better than the worst adapted member of the superior population. If so, the worse individual is replaced by the better one. The ultimate solution is the individual of the superior population with the greatest fitness score.

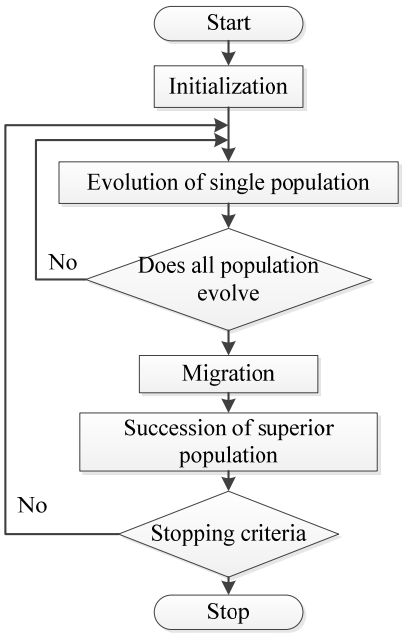


Figure 3. The multi-population evolutionary algorithm diagram.

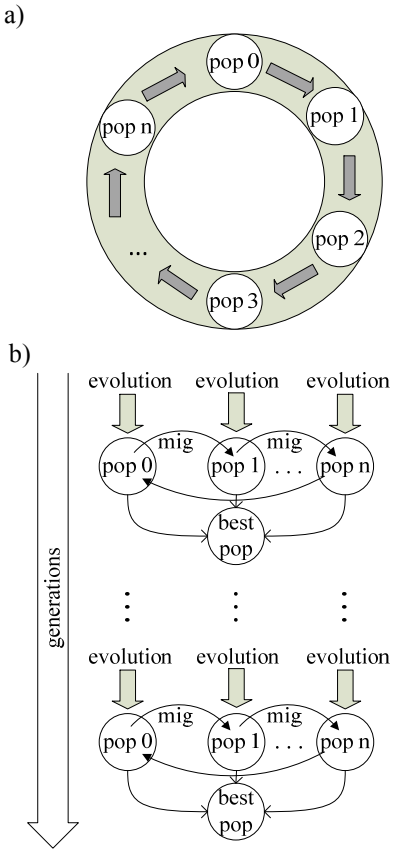


Figure 4 a) Stepping stone migration, b) elite population succession

5 SIMULATION ENVIRONMENT

The multi-population distributed evolutionary algorithm presented has been used to solve the problem of maritime path planning. The algorithm research requires the selection of appropriate test tasks. Three environments representing close to real maritime scenarios were selected. The following parameters were considered: ψ – course, υ – speed.

Environment 1 (Fig. 5a) presents the problem of static forbidden area avoidance. The constraint introduced represents an island. Environment 2 (Fig. 5b) reflects a problem of avoiding a collision with dynamic objects (representing target ships) travelling in opposite directions. One of the objects is travelling with ψ - 180° and υ - 12 knots (target 1) and the other with ψ - 0°, υ - 8 knots (target 2). Environment 3 (Fig. 5c) is a combination of both previous situations, thus it consists of a static obstacle and dynamic ships travelling with ψ - 140° and υ - 8 knots (target 1) and with ψ - 0° and υ - 12 knots (target 2).

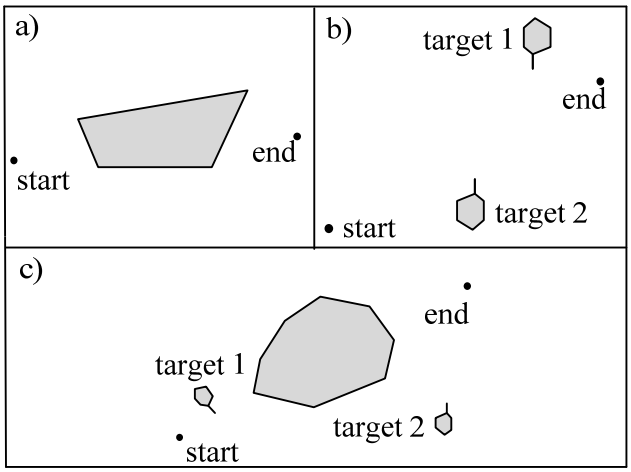


Figure 5. Environment a) 1, b) 2, c) 3

6 SIMULATIONS

The synergy of solutions presented in the tables below was prepared based on the environment type and the initial populations marked as *a* and *b*. Only the parameters analysed were changing. The results below utilize the following abbreviations: SSGA - Steady-State Genetic Algorithm, DGA - Deme Genetic Algorithm, *env* – the type of environment (based on the enumeration from the previous chapter), *init* – the initial population, *pop* – the number of populations, *mig* – the number of migration inviduals, *t* – calculation time, *best* - fitness score of the best individual, *F* – fitness score. Undependably from the simulation, the following algorithm parameters were set:

single population size: 30 individuals, crossover probability: 0.8, mutation probability: 0.15, the number of individuals replaced in the population: 6, selector: proportional roulette, algorithm termination SSGA: 1000 generations, DGA: 200 generations, initial own ship's speed: 20 knots.

Each figure's individual with highest fitness score has his path bolded. The position of the dynamic objects is displayed for the best member of the population. The best calculated solution for each environment is presented in graphical form in Fig. 6

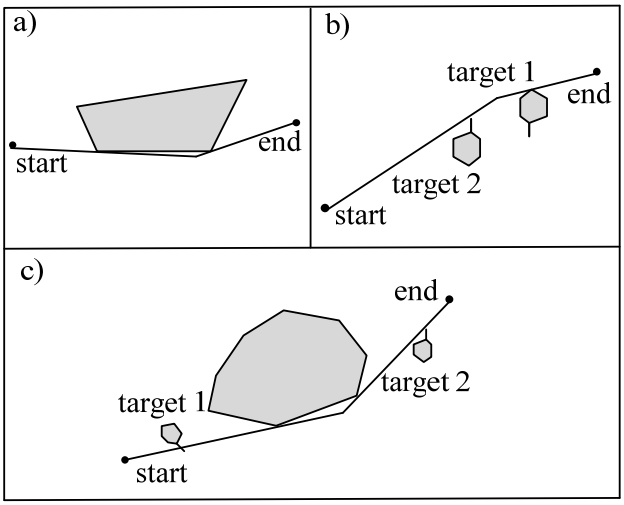


Figure 6. The best calculated solution for each environment

The goal of the research undertaken was to compare the results of SSGA and DGA or also for DGA to establish the impact of the amount of populations and the numbers of migrating individuals on the solution's quality and the computing time. Based on the achieved results a convergence (the ratio between the ultimate fitness score and the number of generation needed to achieve it) analysis was made for SSGA and DGA algorithms.

6.1 *Single and multi-population algorithm comparison*

In the first phase SSGA was compared with the DGA. In the analysed simulations the DGA was operating based on the following parameters: *pop* = 5, *mig* = 3. The use of DGA in comparison to SSGA has extended the calculation time of average 2.5 times. Though the DGA's evolution process was shorter (SSGA – 1000 generation, DGA – 200 generation) it makes higher number of calculations for fitness score for each population. Strong selection pressure used in SSGA makes it so that there is a lot of best individual's copies in the population. Thus it is not necessary to re-calculate its fitness score. Using several independent populations and exchanging individuals between them, DGA introduces higher variety of solutions. In the same time, the probability of creating new individuals during the genetic operations increases. Individuals created this way require evaluation, which extends the calculation process. The time it takes (average 130 seconds) is acceptable for the problem of collision avoidance and meets the near real time requirement (Śmierzchalski and Michalewicz 2000).

After comparing the single population with the multi-population algorithm it has been concluded that regardless of the environment and the initial population, better result were achieved with DGA (Tab. 1). The average score improved by 18%.

Table 1. SSGA and DGA comparison

env	*init*	SSGA		DGA *pop* = 5, *mig* = 3	
		t [s]	*best*	*t* [s]	*best*
1	*a*	18	236.58	55	192.38
	b	16	216.56	44	199.47
2	*a*	42	198.89	129	192.68
	b	32	277.02	130	175.85
3	*a*	54	316.35	217	260.99
	b	59	315.35	218	248.01

6.2 *The impact of the number of populations on the DGA'a performance.*

For the research the *mig* parameter was set to 3. The results are presented in Tab. 2. The algorithm improved or kept the same level of fitness score with the increasing number of populations. The rule doesn't apply to all cases. Simulations with *pop* = 5 provided solutions of the lowest fitness score, while *pop* = 8 provided the best result. Taking the above into account, increasing the population's amount noticeably extends the calculation time, however doesn't guarantee the solution's improvement. Based on the research one can conclude that using *pop* = 3 to 5 allows one to achieve the best results (Tab. 2).

Table 2. Impact of the number of population on the DGA performance

env	*init*	DGA *pop* = 2, *mig* = 3		DGA *pop* = 5, *mig* = 3		DGA *pop* = 8, *mig* = 3	
		t [s]	*best*	*t* [s]	*best*	*t* [s]	*best*
1	*a*	23	205.16	55	192.38	75	201.49
	b	24	201.03	44	199.47	91	197.07
2	*a*	54	223.14	129	192.68	201	192.86
	b	51	176.44	130	175.85	212	176.5
3	*a*	82	258.82	217	260.99	334	272.67
	b	90	255.43	218	248.01	330	252.63

6.3 *The impact of the number of migrating individuals on the DGA's performance.*

Simulations were run with *mig* = 3, 6 and 15 individuals (Tab. 3). The results showed that the number of migrating individuals doesn't affect the ultimate solution. In case of *init b* both *env* 1 and 3 produced similar fitness score. Using *init a* gives various solutions. It was noticed that increasing the amount of migrating individuals helps to smoothen the algorithm convergence characteristic, which eliminates the step aspect of the changes to the fitness score of the best individual (Fig. 8).

Table 3. The impact of the number of the migrating individuals on the DGA's performance

env	*init*	DGA *pop* = 5, *mig* = 3		DGA *pop* = 5, *mig* = 6		DGA *pop* = 5, *mig* = 15	
		t [s]	*best*	*t* [s]	*best*	*t* [s]	*best*
1	*a*	55	192.38	48	203.02	59	202.45
	b	44	199.47	52	195.07	48	198.96
2	*a*	129	192.68	127	163.25	134	163.31
	b	130	175.85	126	175.92	132	175.96
3	*a*	217	260.99	217	308.07	288	239.96
	b	218	248.01	234	250.57	223	252.37

6.4 *The course of the algorithm's convergence*

Fig. 7 - Fig. 8 show the diagram of the solution's convergence depending on the algorithm used, parameters, initial population and the environment. Intermediate results were recorded every 50 generations for the SSGA (Fig. 7) and every 10 for the DGA (Fig. 8). This way 20 samples were achieved for each algorithm. Based on Fig. 7 it was concluded that using SSGA results in poor solution's diversity through the process of evolution. After the exploration phase (regardless of the environment up until about 200th generation) it stagnates within the local minimum. The best result does not improve. The only exception was the simulation made for *env* 2 using *init a*. A sudden improve of the best solution takes place between 850th and 900th generation due to the mutation operator. In DGA the best fitness score is constantly improving. This is due to the number of population which diversifies the solutions. Based on Fig. 8, one can deduce that DGA in comparison with SSGA requires significantly fewer generations to achieve the ultimate fitness score. For *env* 2 it was about 70 and for *env* 3. about 100 generations. While using DGA the solutions provided are consistent regardless of the initial population.

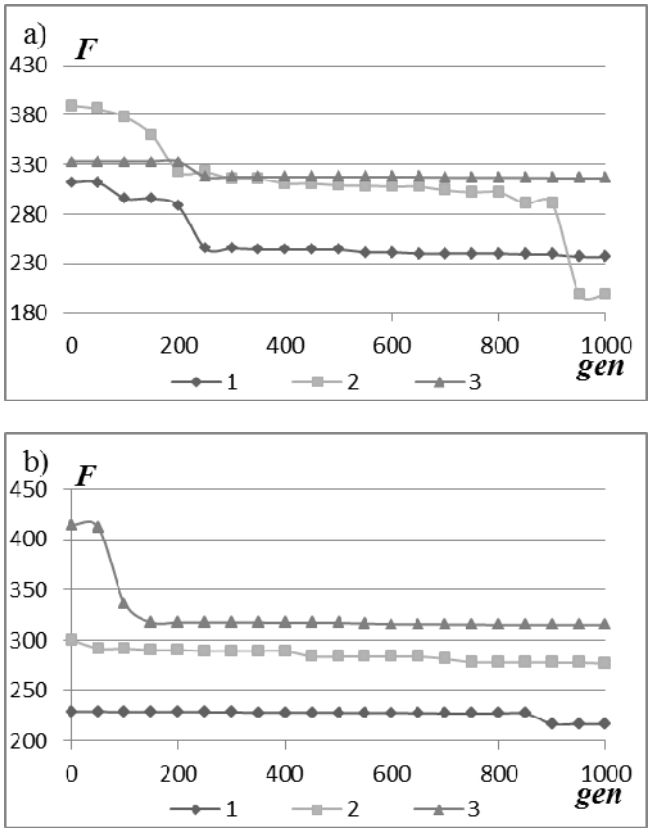
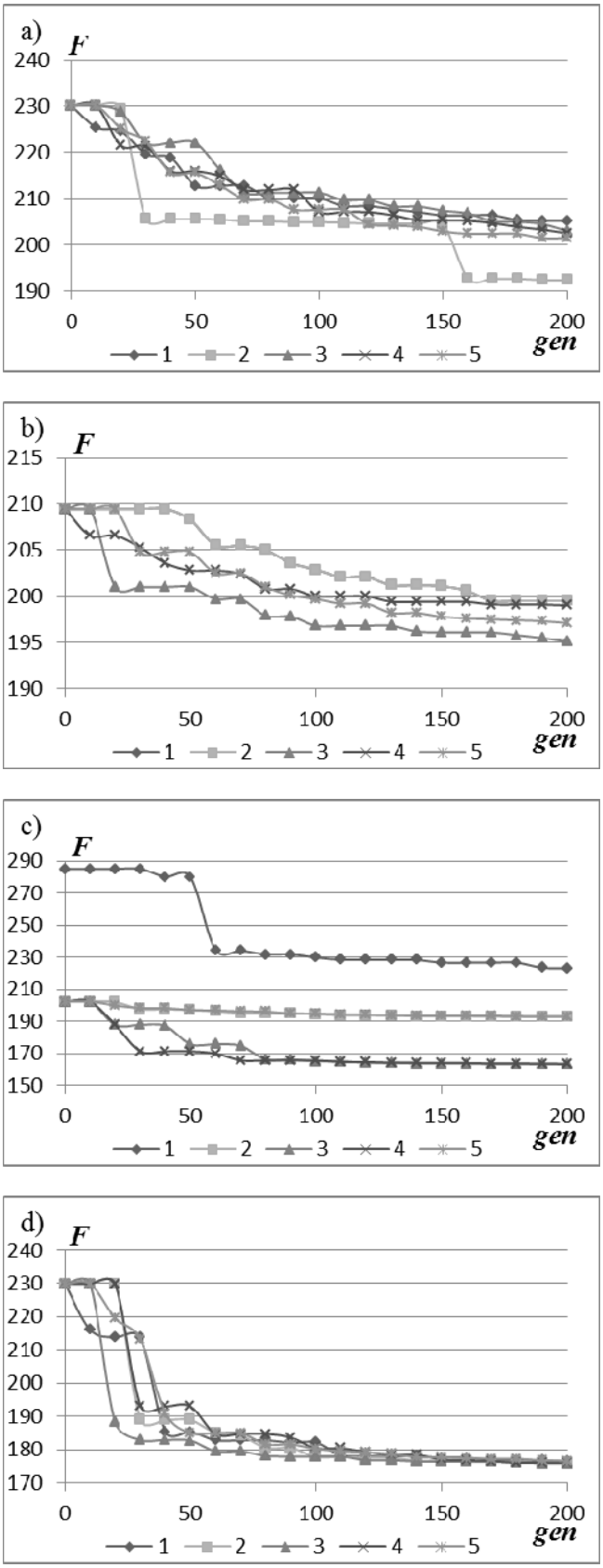


Figure 7. Convergence of the SSGA a) *init a*, b) *init b* (trend: **1** – *env* 1, **2** – *env* 2, **3** – *env* 3)



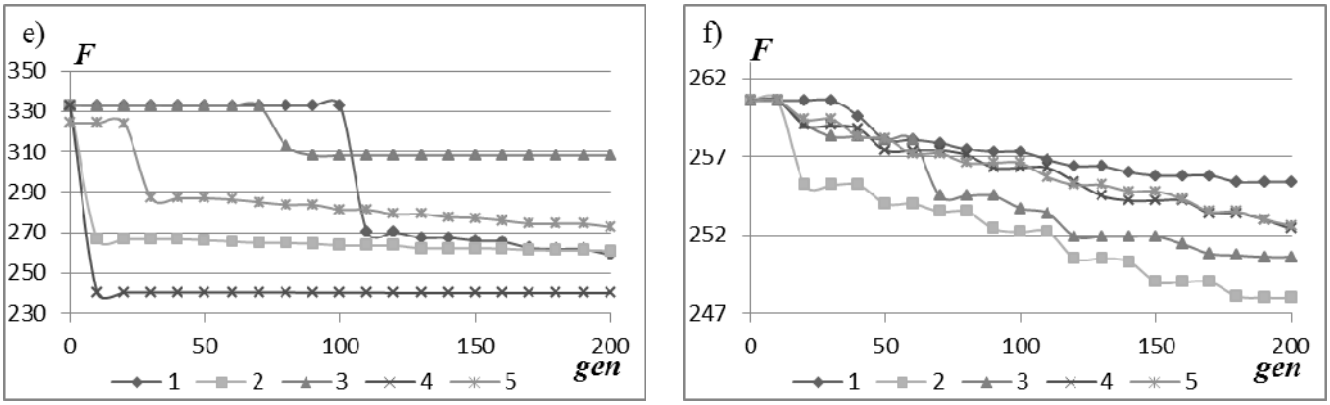


Figure 8. Convergence of the DGA a) *env* 1, *init a*, b) *env* 1, *init b*, c) *env* 2, *init a*, d) *env* 2, *init b*, e) *env* 3, *init a*, f) *env* 3, *init b* (trend: **1** – *pop* = 2, *mig* = 3, **2** – *pop* = 5, *mig* = 3, **3** – *pop* = 5, *mig* = 6, **4** – *pop* = 5, *mig* = 15, **5** – *pop* = 8, *mig* = 3)

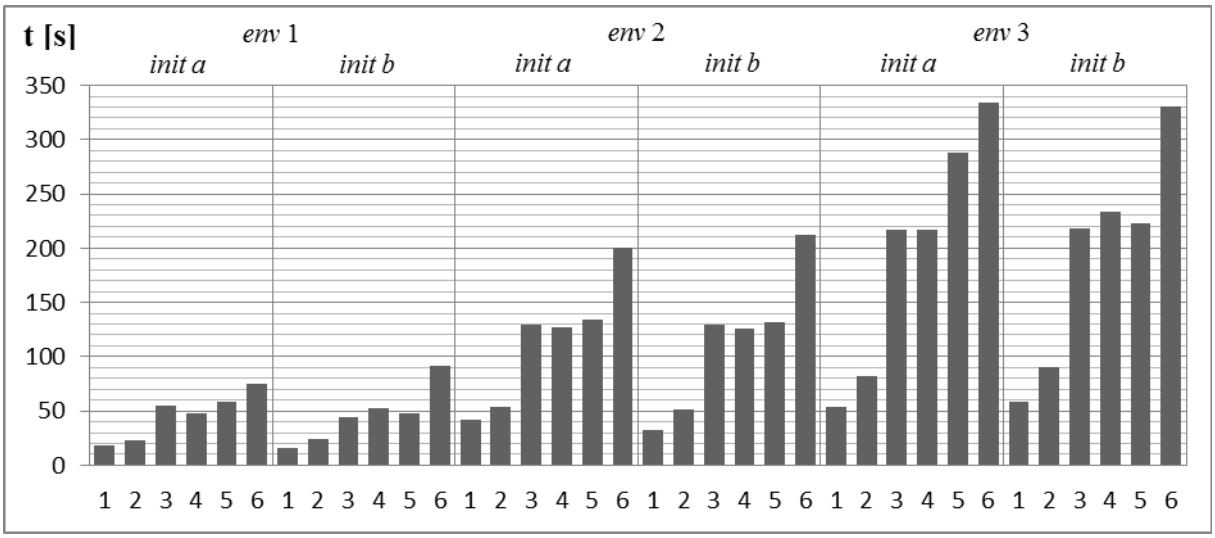


Figure 9. Juxtaposition of the simulation times for the simulations (horizontal axis: **1** - SSGA, **2** - DGA pop=2, mig=3, **3** - DGA pop=5, mig=3, **4** - DGA pop=5, mig=6, **5** - DGA pop=5, mig=15, **6** - DGA, pop=8, mig=3)

6.5 *Calculation time analysis*

The next research step was to analyse the program's performance depending on the environment, initial population, number of populations and the amount of migrating individuals. Fig. 9 presents a juxtaposition of calculation times for selected simulations. It was concluded that calculation times are dependent from the environment's complexity and at a small degree from the initial population. Calculations for *env* 1which consists of a single static obstacle took the shortest amount of time. The calculation time extension was observed for *env* 2 and 3 which is connected with triangulating the position of the dynamic obstacles and the points of potential collision.

In Tab. 4 percentage comparison between calculation time and fitness score of the best individual was presented. All the data are shown for the single population algorithm. In example, using DGA with *pop* = 2 and *init a* (first row in the table) for *env* 1 the calculation time has extended by 27% and the solution improved by 13% in comparison to the result achieved by SSGA.

In terms of computing time, the best solutions were provided by the SSGA. Using DGA with *pop* = 2 results in average solution improvement by 14% and increase of the calculation time by 45%. Similar for *pop* = 5 we get 18% solution improvement and 260% calculation time increase and for *pop* = 8 16% and 450%.

From the data above one can conclude that it's best to use 2 or 3 populations. Increasing the number of generations increases the computation time disproportionate to the quality of the solution.

Table 4. The quality of the solution depending on the calculation time

init	pop	env 1		env 2		env 3	
-	-	t [%]	best [%]	t [%]	best [%]	t [%]	best [%]
a	2	27.7	13.2	28.5	-12.1	51.8	18.1
b	2	50	7.1	59.3	36.3	52.5	19.0
a	5	200	15.7	209.5	12.9	345.6	14.7
b	5	200	8.6	304.1	36.5	281.3	20.6
a	8	316.6	14.8	378.5	3	518.5	13.8
b	8	468.7	9	562.5	36.2	459.3	19.8

7 CONCLUSIONS

The undertaken research has shown that using a distributed genetic algorithm improves the ultimate fitness score in comparison with an algorithm working on a single population. This is achieved regardless of the initial population and the environment's type. It was also shown, that DGA requires far fewer generation to reach a solution comparable to or better than the SSGA. The convergence analysis of both types of the algorithms shows that the distributed approach has a positive impact on maintaining the population's diversity. At the same time the deviations in the solution search for optimum characteristic have decreased. The increase of the number of evolved populations impacts the improvement of the ultimate fitness score. A critical number of populations which prevented further results' improvement was shown. Changing the amount of the migrating individuals had no effect on the final solution, however it did shape the algorithm's convergence characteristic. Using the 20-40% of the population size migration prevented sudden changes of the best fitness score value. After comparing the results of all simulations it was concluded that the greatest impact on the calculation time comes from the environment's complexity and the amount of populations. It was also recognized that using more than 2-3 population brings an unsatisfactory improvement of the ultimate solution taking into account the extended calculation time.

REFERENCES

Belding T. C. (1995). "The Distributed Genetic Algorithms Revised." Proc. of 6th Int. Conf. Genetic Algorithms: 114-121.

Cantu-Paz E. (1999). "Topologies, Migration Rates, and Multi-Population Parallel Genetic Algorithms." Proceedings of GECCO.

Cochran J.K., Horng S. and Fowler J.W. (2003). "A Multi-Population Genetic Algorithm to Solve Multi-Objective Scheduling Problems for Parallel Machines." Computers and Operations Research, Vol 30: 1087-1102, Oxford, UK.

Forrest S. and Mitchell M. (1991). "The performance of genetic algorithms on Walsh polynomials: Some anomalous results and their explanation." Proceedings of the Fourth International Conference on Genetic Algorithms, In Belew, R. & L. Booker (Eds.): 182-189, San Mateo.

Forrest S. and Mitchell M. (1993). "Relative building-block fitness and the Building Block hypothesis." In Whitley L.D. (Ed.), Foundations of Genetic Algorithms 2: 109-126, San Mateo, CA: Morgan Kauf-mann.

Gehring H. and Bortfeldt A. (2002). "A parallel genetic algorithm for solving the container loading problem." International Transactions in Operational Research, Vol. 9, No. 4: 497–511.

Goldberg D.E. (1989). "Genetic Algorithms in Search, Optimization, and Machine Learning." Boston: Addison-Wesley Longman Publishing Co., Inc.

Lenart A.S. (1986). "Wybrane problemy analizy i syntezy okrętowych systemów antykolizyjnych." Budownictwo okrętowe nr XLIV, Zeszyty naukowe Politechniki Gdańskiej nr 405, Gdańsk 1986.

Martikainen J. (2006). "Methods for Improving Reliability of Evolutionary Computation Algorithms and Accelerating Problem Solving." Ph.D. Thesis, Helsinki University of Technology, Dep. of Electrical and Communications Engineering, Espoo.

Martikainen J. and Ovaska S.J. (2006). "Hierarchical two-population genetic algorithm." International Journal of Computational Intelligence Research vol. 2, No. 4.

Michalewicz Z. (1996). "Genetic Algorithms + Data Structures = Evolution Programs." Spriger - Verlang.

Mitchell M., Forrest S. and Holland J. H. (1992). "The royal road for genetic algorithms: Fitness landscapes and GA performance." In Proc. of the First European Conference on Artificial Life: 245-254, Cambridge, MIT Press.

Mitchell M., Holland J.H. (1993). "When will agenetic algorithm outperform hill climbing?" Santa Fe Institute working paper 93-06-037, Santa Fe, NM: Santa Fe Institute.

Mitchell M., Holland J.H. and Forrest S. (1994). "When will a genetic algorithm outperform hill climbing?" Advances in Neural Information Processing Systems 6 San Mateo, CA: Morgan Kaufmann.

Śmierzchalski R. (1997). "Trajectory planning for ship in collision situations at sea by evolutionary computation." In Proc. of the IFAC MCMC'97, Brijuni, Croatia.

Śmierzchalski R. (1998). "Synteza metod i algorytmów wspomagania decyzji nawigatora w sytuacji kolizyjnej na morzu." DSc. dissertation, Gdynia.

Śmierzchalski R. and Michalewicz Z. (2000). "Modeling of a Ship Trajectory in Collision Situations at Sea by Evolutionary Algorithm." IEEE Transaction on Evolutionary Computation, Vol.4, No.3.

Tanese R. (1989a). "Distributed Genetic Algorithms." Proc. of 3rd Int. Conf. Genetic Algorithms: 432-439.

Tanese R. (1989b). "Distributed Genetic Algorithms." Ph.D. Thesis, University of Michigan, Ann Arbor.

Wall M. (1996). "GAlib: A C++ Library of Genetic Algorithm Components." MIT.

Whitley D. (1997). "Island Model Genetic Algorithms and Linearly Separable Problems." Proc. of AISB Workshop on Evolutionary Computation.

Xiao J. and Michalewicz Z. (1999). "An Evolutionary Computation Approach to Planning and Navigation." Chapter in Soft-Computing and Mechatronics, Physica-Verlag.