Position papers of the 2014 Federated Conference on Computer Science and Information Systems pp. 21–28
DOI: 10.15439/2014F391
ACSIS, Vol. 3

Solving Graph Coloring Problem with Parallel Evolutionary Algorithms in a Mesh Model

Zbigniew Kokosiński and Piotr Domagała
Cracow University of Technology
Faculty of Electrical & Computer Engineering
ul. Warszawska 24, 31-155 Kraków, Poland
Email: zk@pk.edu.pl

***Abstract*—In this paper a parallel evolutionary algorithm (PEA) for coloring graph vertices is investigated. In the algorithm we apply a diffusion model of parallelism (DM). Evolutionary computations are performed in a regular mesh with either a constant size global population or a constant subpopulation in a single node. The performance of the PEA-DM is verified by computer experiments on standard DIMACS graph coloring instances. For recombination well known crossover and mutation operators are chosen. Selection mechanisms include standard roulette and tournament. The results obtained by PEA-DM are compared with a classical evolutionary algorithm. It is possible to define dimensions of the rectangular mesh and two types of additional local connections: boundary enclosures (cyclic mesh) and diagonal links. The problem of optimal selection of the mesh configuration as well as global population and subpopulation sizes is adressed.**

I. Introduction

GRAPH k–colorability problem belongs to the class of NP–hard combinatorial problems [14], [20]. This decision problem is defined for an undirected graph $G = (V, E)$ and positive integer $k \leq |V|$: is there an assignment of available k colors to graph vertices, providing that adjacent vertices receive different colors ? With additional assumptions many variants of the coloring problem can be defined such as equitable coloring, sum coloring, contrast coloring, harmonious coloring, circular coloring, consecutive coloring, list coloring etc. [18], [27]. In optimization version of the basic problem called GCP, a conflict–free coloring with minimum number of colors is searched. Intensive research conducted in this area resulted in a large number of exact and approximate algorithms, heuristics and metaheuristics [5], [26], [31], [36], [37]. However, the reported results are often difficult to compare due to specific assumptions, different algorithms and their implementation details, tuning of parameters, computing platforms, test data sets etc. GCP was the subject of Second DIMACS Implementation Challenge held in 1993 [19] and Computational Symposium on Graph Coloring and Generalizations in 2002. A collection of graph coloring instances in DIMACS format and summary of results are available at [38], [39], [40].

Evolutionary algorithm (EA) is a metaheuristic often used for GCP [10], [11], [12], [13], [21], [30], [32], [35]. Parallel versions of evolutionary algorithms (PEAs) were reviewed in [1]. One popular model is master–slave in which master processor assignes portions of computations to slave processors [6]. Another approach is based on co-evolution of a number of populations that exchange the genetic information during the evolutionary process according to a communication pattern [2], [3], [9]. That approach includes migration and diffusion models od PEAs.

PEA were applied to many hard optimization problems (e.g. [1], [28]). Parallel metaheuristics for GCP and related problems were recently published in [23], [24], [25], [31].

The main purpose of the master–slave model is speeding up processing of one global population by parallelization of computations. In the migration model of PEA the model of interactions between co–evolving populations can affect the quality of the solution [23]. If speedup is not essential one can simulate and test migration–based PEA with the help of a sequential program. An alternative parallel model is the diffusion model, where evolution takes place among neighbouring subpopulations. There are many possible neighbourhood patterns. Among them 2–dimentional mesh with subpopulations placed in its nodes is a typical option because it corresponds to a model of computations very popular in parallel procesing, i.e. 2D array of processors.

In this article some results of experiments with PEA in diffusion model for graph coloring problem are described. In the paper two recombination operators for coloring chromosomes are used: CEX (Conflict Elimination Crossover) [23] that reduces the number of color conflicts with the help of selective copy operations and GPX (Greedy Partition Crossover) [13]. They both are problem–specific crossovers, designed particularly for GCP and passed a series of experimental verification in EA environment [15], [22].

In experimental part of the paper, widely accepted DIMACS benchmark graphs were used. The obtained results show adventages and limitations of PEA in the diffusion mesh model for hard optimization problems like GCP.

II. Graph Coloring Problem

Let us define formally the optimization problem GCP.

For given graph $G(V, E)$, where : V — set of graph vertices, $|V| = n$, and E — set of graph edges, $|E| = m$, the optimization problem GCP is formulated as follows: find the minimum positive integer k, $k \leq n$, and a function $c : V \longrightarrow \{1, \ldots, k\}$, such that $c(u) \neq c(v)$ whenever

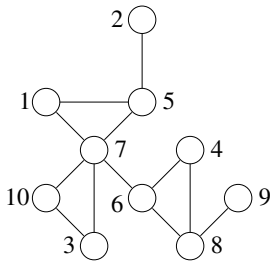

Fig. 1. Exemplary graph G(V,E)

$(u, v) \in E$. The obtained value of k is refered to as graph chromatic number $\chi(G)$.

Similarly, graph edge coloring problem for given graph $G(V, E)$ can be defined. One can find solution to mimimum edge coloring by solving vertex coloring problem for edge graph $G_e(V_e, E_e)$ associated with the given graph $G(V, E)$ [18], [26]. An exemplary graph $G(V, E)$ with ten vertices is shown in Fig.1.

In graph coloring problem *k*–colorings of graph vertices are encoded in chromosomes representing set partitions with exactly *k* blocks. There are two equivalent notations for vertex colorings that are commonly used in algorithm design.

In *assignment* representation available colors are assigned to an ordered sequence of graph vertices. Thus, the vector c=<c[1],c[2], ...,c[n]>. represents a vertex coloring. For the graph in Fig.1, an optimal 3–coloring is denoted by vector c=<1,2,3,2,3,1,2,3,2,1>.

In *partition* representation a vertex coloring is a unique sequence of partition blocks in Hutchinson representation [16]. Each block of partition p does correspond to a single color. Elements inside each block are ordered in increasing lexicographic order, and all blocks are ordered increasingly according to the value of their first element. For our graph the same optimal 3–coloring is denoted by partition p={1,6,10}{2,4,7,9}{3,5,8}.

III. Models of Parallel Evolutionary Algorithms

There are many models of parallelism in evolutionary algorithms: master–slave PEA, migration based PEA, diffusion based PEA, PEA with overlaping subpopulations, population learning algorithm, hybrid models etc. [4], [6], [7], [8], [17], [29], [33], [34].

The above models are characterized by the following criteria:

- number of populations : one, many;
- population types : disjoint, overlaping;
- population topologies : various graph models;
- interaction model : isolation, migration, diffusion;
- recombination, evaluation of individuals, selection : distributed/local, centralized/global;
- synchronization on iteration level: synchronous/asynchronous algorithm.

The most common models of PEA are:

- master-slave : one global population, global evolutionary operations, fitness functions computed by slave processors);
- massively parallel (cellular): static overlapping subpopulations with a local structure, local evolutionary operations and evaluation;
- migration (with island as a submodel): static disjoint subpopulations/islands, local evolutionary operations and migration;
- diffusion (with mesh as a submodel); static disjoint subpopulations/nodes, local evolutionary operations and migration;
- hybrid : combination of one model on the upper level and other model on the lower level (the speedup achieved in hybrid models is equal to product of level speedups).

A. Migration Model of Parallel Evolutionary Algorithm

Migration models of PEAs consist of a finite number of disjoint subpopulations that evolve in parallel on their "islands" and only occasionally exchange evolutionary informations under control of a migration operator. Co–evolving subpopulations are built of individuals of the same type and are ruled by one adaptation function. The selection process is decentralized.

In the model the migration is performed on a regular basis. During the migration phase every island sends its representatives (emigrants) to all other islands and receives the representatives (immigrants) from all co–evolving subpopulations. This topology of migration reflects so called "pure" island model. The migration process is fully characterized by migration size, distance betweeen populations and migration scheme. Migration size determines the emigrant fraction of each population. This parameter is limited by capacity of islands to accept immigrants. The distance between migrations determines how often the migration phase of the algorithm occurs. Three migration schemes may be applied: no migration, migration of randomly selected individuals and migration of best individuals of the subpopulation.

B. Diffusion Model of Parallel Evolutionary Algorithm

The diffusion model of PEA is a fine-grained EA [6] with its global population distributed on a 2D mesh of size $w \times z$, where subpopulations are placed in the mesh nodes (cells). The crossover operation is local in that sense, that the recombined chromosomes are members of subpopulations in the closest neighbourhood, i.e. have distance 1 in the underlying graph G(V, E). Solutions with outstanding fitness function are able to "diffuse" step by step in the graph across the whole global population. The implementation of PEA-DM in diffusion model can be either synchronous or anynchronous on a parallel machine with shared memory. Parent chromosomes can be selected for recombination at random, but it is recommended to choose at random only the first parental population, all next should be selected on the basis of the fitness function of all chromosomes in the neighbourhood .

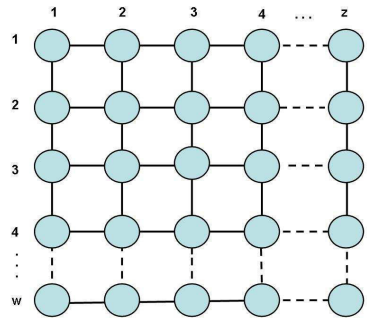

Fig. 2. A simple rectangular mesh of size $w \times z$.

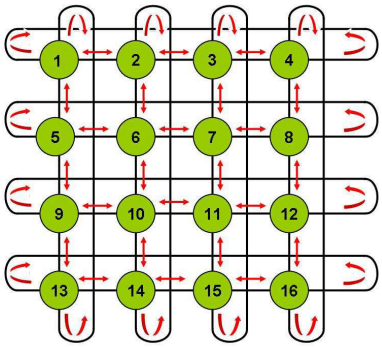

Fig. 3. A cyclic square mesh of size 4×4.

In the simple mesh model presented in Fig. 2 graph edges represent connections between nodes. Boundary nodes have a limited communication ability, because they have smaller vertex degrees then internal nodes of the graph G(V, E). For instance, for the square mesh of size 4×4 the mesh containes 75 % of boundary nodes. Similarly, for the mesh 5×5 there is 64 % of boundary nodes, for the mesh 8×8 there is 43 % of boundary nodes, for the mesh 10×10 there is 36 % of boundary nodes, for the mesh 15×15 there is 25 % of boundary nodes, for the mesh 20×20 there is 19 % of boundary nodes, etc. The lowest degree have the four corner nodes.

It is reasonable to expect that the mesh size and node degree in the graph shall influence the PEA-DM performance. Propagation time of good solutions across the whole network increases with the network size. The simplest way to eliminate node degree irregularities in the ordinary mesh is to add cyclic connections to boundary nodes (boundary enclosures), The resulting cyclic mesh 4×4 is depicted in Fig. 3.

The structure of 2D mesh is very popular since it can be implemented in MIMD computers. The sizes of the mesh can be variable within certain ranges. The number of communication chanels for each node processor is 4. That value is constant and does not depend on the number of processors. Maximum distance between processors in the cyclic mesh is 0,5(w+z).

In order to increase node degree in the cyclic mesh an additional connections are to be added. In Fig. 4 extra diagonal

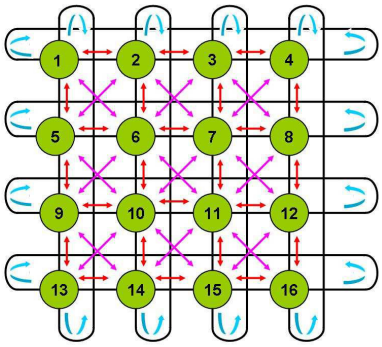

Fig. 4. A cyclic square mesh of size 4×4 with extra diagonal connections.

Algorithm 1 EA for a subpopulation in diffusion model

Require: cell position in the mesh, subpopulation size and chromosome parameters
Ensure: best coloring in P_{t+1}
$t \leftarrow 0$; [reset iteration counter]
initialization of subpopulation P_t;
evaluation of subpopulation P_t;
while not termination condition **do**
parents $T_t \leftarrow$ selection from P_t and its neighbourhood;
offspring population $O_t \leftarrow$ crossover and mutation on T_t;
evaluation of $\{P_t \cup O_t\}$;
$P_{t+1} \leftarrow$ selection from $\{P_t \cup O_t\}$;
$best \leftarrow$ best coloring in P_{t+1};
$t \leftarrow t + 1$;
end while

connections are shown. In this way degrees of internal nodes increase from 4 to 8 and degrees of boundary nodes increase from 4 to 6, except the corner nodes which degrees increase only by one, from 4 to 5. It is possible to complete the network connectivity by adding diagonal cycles for boundary nodes. In such architecture, for the regularity of the extended cyclic mesh, we pay in longer iteration time of PEA-DM.

IV. Recombination Operators

In this section a collection of crossover, mutation and selection operators is introduced that can be used in our PEA-DM. Two recombination operators: CEX, GPX and the mutation operator First Fit were designed especially for GCP. The mutation Transposition is more versatile. Other efficient recombination operators were proposed in [32]. The cost function is adapted for PEA-DM. All examples given in this section refer to the graph instance shown in Fig.1.

A. Conflict Elimination Crossover

In conflict–based crossovers for GCP the assignement representation of colorings is used and the offspring tries to copy conflict–free colors from their parents. The operator CEX (Conflict Elimination Crossover) reveals some similarity to the classical crossover. Each parental chromosome p and r is

partitioned into two blocks. The first block consists of conflict–free nodes while the second block is built of the remaining nodes that break the coloring rules.

The last block in both chromosomes is then replaced by corresponding colors taken from the other parent. This recombination scheme provides inheritance of all good properties of one parent and gives the second parent a chance to reduce the number of existing conflicts. However, if a chomosome represents a feasible coloring the recombination mechanism will not work properly. Therefore, the recombination must be combined with an efficient mutation mechanism. As a result two chromosomes s and t are produced. The operator CEX is almost as simple and easy to implement as the classical crossover (see Algorithm 2).

Behaviour of the CEX crossover is shown in Example 1.

Example 1

Two parents represent different 5–colorings of a graph with 10 vertices i.e. sequences p=<5,2,**3**,**4**,1,**4**,2,5,1,**3**>, and r=<1,4,**5**,2,3,3,**5**,4,2,**5**>. Vertices with color conficts are marked by bold fonts. Thus, the chomosome p has 6 vertices with feasible colors and 4 vertices with color conflicts while the chomosome r has 7 vertices with feasible colors and 3 vertices with color conflicts.

Replacing the vertices with color conflicts by vertices taken from the other parent we obtain the following two chromosomes: s=<5,2,**5**,2,1,3,2,5,1,**5**> and t=<1,4,**3**,2,3,3,2,4,2,**3**>. (see Fig. 5)

It is observed that obtained chromosomes represent now two different 4–colorings of the given graph (reduction by 1 with respect to initial colorings) and the number of color conflicts is now reduced to 2 in each chromosome.

B. Greedy Partition Crossover

The method called Greedy Partition Crossover (GPX) was designed by Galinier and Hao for recombination of colorings or partial colorings in partition representation [13]. It is assumed that both parents are randomly selected partitions with exactly k blocks that are independent sets. The result is a single offspring (a coloring or partial coloring) that is built successively in a greedy way. In each odd step select the maximum block from the first parent is selected. Then the block is added to the result and all its nodes from the both parents are removed. The procedure is repeated at most k times since in some cases the offspring has less blocks then the parents. This possibility is not considered in the original paper [13]. Finally, unassigned vertices (if they exist) are assigned at random to existing blocks of partition.

Algorithm 2 The crossover operator CEX

Require: V, p, r
Ensure: s, t
$s \leftarrow r$; $t \leftarrow p$;
$b \leftarrow x$;
copy block of conflict-free vertices V_{cf}^{p} from p to s;
copy block of conflict-free vertices V_{cf}^{r} from r to t;

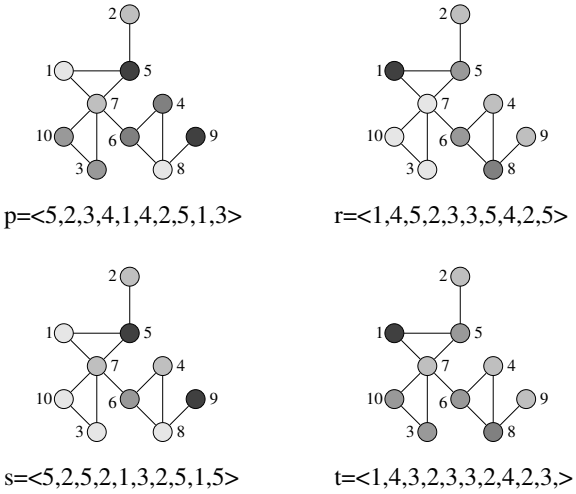

Fig. 5. An illustration of CEX crossover (see Example 1)

The first parent is replaced by the offspring while the second parent is returned to population and can be recombined again in the same generation. GPX crossover is performed with a constant probability.

C. Mutation Operators

Two types of mutation operators described in literature are used. Transposition (T) is a classical type of mutation that exchanges colors of two randomly selected vertices in the assignment representation. The second mutation operation called First Fit (FF) is designed for colorings in partition representation and is well suited for GCP. In First Fit mutation one block of the partition is selected at random and we try to make a conflict–free assignment of its vertices to other blocks using the heuristic First Fit. Vertices with no conflict–free assignment remain in the original block. Thus, as a result of the mutation First Fit the color assignment is partially rearranged and the number of partition blocks is often reduced by one.

D. Selection Operator

Selection process maintains constant size of population selected by means of a fitness function.

In the first phase of EA, when no initial information is available, the quality of a solution is measured by the following cost function:

$$cost(c) = conflicts(c) \cdot colors(c), \tag{1}$$

where:
c – is the current coloring,
$conflicts(c)$ – is the number of conflicts in c,
$colors(c)$ – is the number of colors in the c.

In the second phase of the algorithm, for conflict–free colorings $conflicts(c) = 0$ and $cost(c) = 0$. Therefore, in that case the cost function is computed by the following formula:

$$f(c) = conflicts(c) + colors(c) + p(c), \tag{2}$$

where the penalty function $p(c)$ equals:

$$p(c) = \begin{cases} 2 \cdot (colors(c) - best), & if \quad colors(c) \geq best \\ 0, & if \quad colors(c) < best \end{cases} \tag{3}$$

and:
$best$ – is a number of colors in the best individual so far.

The proportional (roulette) selection can be performed in two phases of the algorithm with the fitness function $1/f(p)$. Alternatively, the tournamet selection can be performed in both phases of the algorithm on randomly selected individuals from subpopulations with the analogous fitness function.

V. Computer experiments

For computer experiments several graph instances were used that are available in the web archives [38], [39]. They are collections of graphs in DIMACS format with known parameters m, n and usually $\chi(G)$.

In our program *PGA-DM for GCP* diffusion models of PEA can be simulated. It is possible to set up most parameters of evolution, monitor evolution process in each node and measure both the number of generations and time of computations. In order to avoid misunderstanding we always report throughout the paper the total execution time of the sequential simulation of the PGA. In the preprocessing phase we converted list of edges representation into adjacency matrix representation. The program generates detailed reports and basic statistics [22]. All computer experiments were performed on a computer with AMD Athlon 2000 processor (1,67 GHz, 512 MB RAM). The performance of the processor was never a critical factor.

The research was focused on the diffusion model of PEA. In the experiments the following aspects of this model were taken into consideration: 1. comparison of PEA-DM versus EA with respect to quality of solution, number of iterations and time of computation. 2. influence of mesh size for constant node population solution; 3. comparison of acyclic versus cyclic square meshes.

In all experiments and for all crossover operators we used constant crossover probability = 0.8 and mutation probability = 0.1.

A. Comparison of PEA-DM with EA

In the first experiment PEA-DM was tested against traditional EA. The jean(80, 254, 10) graph was used. Classical EA was obtained as a special case in the program PEA-DM for GCP with parameters: mesh size = 1×1, population size = 320, initial number of colors = 2, CEX crossover, FF mutation and Tournament selection. In diffusion PEA the mesh size = 8×8 and node subpopulation = 5. Computations were repeated 30 times. Termination condition was either optimal coloring for $\chi(G) = 10$ or the number of iterations = 1000. All results of the comparison were collected in Table I.

In all experiments a conflict–free coloring was reached for the given graph. Optimal colorings were more likely to happen with PEA-DM. Average time of computations for obtaining a conflict–free coloring was lower for EA. Execution times of 1000 iterations in PEA-DM and EA were close to each other: the average execution time for EA was 202,7 [s]; for PEA-DM it was 204,3 [s].

B. PEA-DM with a constant population size in a node

In the second experiment PEA-DM was investigated for various sizes of acclic square mesh, constant population size =5 in all nodes and initial number of colors = 2. Two DIMACS graphs were used for this configuration of the program PEA-DM : huck(74,301,11) and queen6.6(36,290,7). Computations were conducted with the following settings: random initial population, CEX crossover, FF mutation and Tournament selection. All experiments were repeated 10 times. Termination condition was either finding an optimal graph coloring ($\chi(G)$ is known for both graphs) or 5000 iterations completed. The computational results are presented in Table II (columns B, W, A contain the best, the worst, and the average results obtained in 10 experiments, respectively).

For huck graph and mesh size 4×4 only in 5 experiments (5/10) a conflict–free coloring was obtained and only 2 optimal colorings. For the mesh size 6×6 all colorings were conflict–free and half of them was optimal. For bigger mesh sizes a higher percentage of optimal coloring was obtained in a shorter time.

For queen6.6 graph conflict–free colorings were possible starting from the mesh size 6×6 (6/10). For bigger mesh sizes more conflict–free colorings were found with less colors used. However, due to graph difficulty the PEA-DM was not able to find any optimal solution with $\chi(G)$=7 colors. The minimal coflict–free coloring was found with the number of colors = 8.

In conclusion, the efficiency of the PEA-DM with constant population size in a node strongly depends on the size of the mesh. For small meshes small global population (rather tens then hundreds of individuals) can not provide sufficient diversity of solutions. The bigger mesh size implies in general more computations (longer processing time), but usually less iterations and sufficient diversity of population to achieve a satisfactory solution. The optimal mesh size for huck graph is 14×14. The other possibility to improve PEA-DM efficiency is to increase the population size in nodes. Thus, PEAs with small meshes and bigger population sizes in nodes become similar to PEAs in migration models, where the optimal number of populations is moderate. No mesh of optimal size was found for queen6.6 graph. For this graph the PEA efficiency can be improved with bigger subpopulations and higher number of iterations.

C. PEA-DM with a constant global population size

Taking into account results of the previous experiments the constant global populations size was set to 700, approximately. The are minor deviations from that size due to variable mesh

TABLE I
COMPARISON OF EA AND PEA-DM

graph $G(V,E)$	algo–rithm	colo–rings		cost / number of iterations / time			
				min	max	avg.	std. dev.
jean |V|=80 |E|=254 $\chi(G)$=10	EA	30/30 conflict–free	colors	10	13	11,23	0,68
			it	19	345	46,56	57,05
			t[s]	3,8	68,7	9,42	11,35
		2/30 optimal (7%)	colors	10	10	10	0
			it	37	39	38	1,41
			t[s]	7,0	7,7	7,35	0,50
	PEA-DM	30/30 conflict–free	colors	10	15	11,17	0,89
			it	26	471	180,1	216,4
			t[s]	7,0	94,7	36,56	0,68
		4/30 optimal (14%)	colors	10	10	10	0
			it	68	153	97,5	39,64
			t[s]	14,5	32,3	20,3	8,1

TABLE II
PEA-DM WITH CONSTANT POPULATION SIZE IN A NODE = 5

Graph	mesh size	4×4			6×6			8×8		
huck |V|=74 |E|=301 $\chi(G)$=11		B	W	A	B	W	A	B	W	A
	colors	11	13	12	11	13	11,6	11	12	11,4
	it	26	1440	336,8	5	4594	1022,3	3	1748	262,3
	t[s]	1,2	67,5	17,4	0,5	515,6	115,1	2,7	317,3	54
	mesh size	10×10			12×12			14×14		
		B	W	A	B	W	A	B	W	A
	colors	11	11	11	11	11	11	11	11	11
	it	6	74	38	4	75	32,7	8	29	16,6
	t[s]	1,6	20,9	11	1,4	32	14	5,2	16,6	9,6
	mesh size	16×16			18×18			20×20		
		B	W	A	B	W	A	B	W	A
	colors	11	11	11	11	11	11	11	11	11
	it	15	48	23,7	5	37	20,4	8	25	16,4
	t[s]	10,8	34,8	17,5	4,9	34,3	18,8	8,2	28,4	17,3
queen6.6 |V|=36 |E|=290 $\chi(G)$=7	mesh size	4×4			6×6			8×8		
		B	W	A	B	W	A	B	W	A
	colors	-	-	-	8	11	9,8	8	11	9,2
	it	-	-	-	49	4596	1990	10	4507	1529
	t[s]	-	-	-	2,2	185,5	89,5	0,6	319,5	108
	mesh size	10×10			12×12			14×14		
		B	W	A	B	W	A	B	W	A
	colors	8	10	8.9	8	10	8,8	8	9	8,5
	it	60	2933	638	7	3580	694,2	31	1725	762,4
	t[s]	6,2	319	69,2	1	552,7	197,5	6,3	360,2	150,3
	mesh size	16×16			18×18			20×20		
		B	W	A	B	W	A	B	W	A
	colors	8	9	8,3	8	9	8,1	8	8	8
	it	49	936	428,3	59	886	469,6	130	917	627,9
	t[s]	13	250,8	115,6	19,9	299,3	158,5	54,2	380,3	260,9

size. Two DIMACS graphs were used for this configuration of the program PEA-DM : games120(120, 638, 9) and david(87, 406, 11). Computations for games120 graph were conducted with the following settings: random initial population, CEX crossover, FF mutation and Tournament selection. All experiments were repeated 10 times. Termination condition was either finding an optimal graph coloring or 500 iterations completed. Both acyclic and cyclic meshes were considered. The computational results are presented in Table III (columns B, W, A contain the best, the worst, and the average results obtained in 10 experiments, respectively).

For games120 graph the optimal coloring were found for all acyclic mesh sizes. Only for the mesh 4×4 a single solution was not optimal. Changing the mesh size does not influence the computational time of PEA-DM. In average 500 iterations lasted about 450 [s]. Also the time of finding solutions was not influenced.

A variant of PEA-DM with 12×12 mesh and CEX crossover replaced by GPX was also tested for games120 graph. Unfortunately no conflict–free coloring was found, even with 1500 iterations.

For david graph the conflict–free colorings were found for all acyclic mesh sizes, but no optimal colorings were met. The best average results were obtained for 4×4 mesh (141,4 [s])

TABLE III
PEA-DM WITH APPROX. CONSTANT GLOBAL POPULATION SIZE = 700

Graph	population size	4 × 4 × 43=688			6 × 6 × 19=684			8 × 8 × 11= 704		
	acyclic mesh	B	W	A	B	W	A	B	W	A
	colors	9	10	9,1	9	9	9	9	9	9
	it	25	482	221,5	46	297	149	68	397	211,1
	t[s]	22	433	232,1	35,5	236,6	117,9	61,7	359,7	190,5
	population size	10 × 10 × 7=700			12 × 12 × 5=720					
	acyclic mesh	B	W	A	B	W	A			
	colors	9	10	9,1	9	9	9			
games120	it	58	405	185	68	380	217			
|V|=120	t[s]	44,7	314,6	152,2	63	352,5	201,5			
|E|=638	population size	4 × 4 × 43=688			6 × 6 × 19=684			8 × 8 × 11= 704		
$\chi(G)$=9	cyclic mesh	B	W	A	B	W	A	B	W	A
	colors	9	10	9,2	9	9	9	9	9	9
	it	12	329	119,4	19	156	131,7	32	234	120,6
	t[s]	12,9	362,5	120	16,2	131,7	81,3	28,7	232,6	110
	population size	10 × 10 × 7=700			12 × 12 × 5=720					
	cyclic mesh	B	W	A	B	W	A			
	colors	9	9	9	9	9	9			
	it	87	307	167,2	44	306	178,6			
	t[s]	76,8	273	155	40	280	163			
	population size	4 × 4 × 43=688			6 × 6 × 19=684			8 × 8 × 11= 704		
	acyclic mesh	B	W	A	B	W	A	B	W	A
	colors	13	15	13,7	13	14	13,9	14	15	14.1
	it	39	493	301,9	123	449	264	151	348	233
	t[s]	27	355	216,6	88,5	327	190,1	103	271	168,7
	population size	10 × 10 × 7=700			12 × 12 × 5=720					
	acyclic mesh	B	W	A	B	W	A			
	colors	14	14	14	14	15	14,4			
david	it	121	476	197,7	142	364	225			
|V|=87	t[s]	85,9	337	141,4	122	288,4	196,5			
|E|=406	population size	4 × 4 × 43=688			6 × 6 × 19=684			8 × 8 × 11= 704		
$\chi(G)$=11	cyclic mesh	B	W	A	B	W	A	B	W	A
	colors	11	16	13,8	13	15	14,2	13	14	13,9
	it	56	400	193,4	94	496	223,3	82	426	205
	t[s]	41	397	160,5	63	317	148,8	64	336	161
	population size	10 × 10 × 7=700			12 × 12 × 5=720					
	cyclic mesh	B	W	A	B	W	A			
	colors	13	15	13,9	13	15	13,8			
	it	90	483	220,4	119	456	263,4			
	t[s]	73,4	392	179,3	105	396	229			

and the worst were received for 12×12 mesh (229 [s]).

The above experiments were repeated for cyclic meshes. It results, that in general a smaller number of iterations is sufficient for the same result. Only for david graph and 10×10 mesh the number of iterations increases but the smaller number of colors is received. In other cases the number of colors is similar for acyclic and cyclic meshes. It seems that using PEA-DM with cyclic meshes is advantageous, since the number of iterations decreases and the time of computations decreases too. One exception is david graph and cyclic 10×10 and 12×12 meshes when longer computations lead to colorings with lower number of colors. Computational time for performing the same number of iterations is longer for cyclic meshes.

One can expect that with extra diagonal connections the processing time for one iteration in cyclic meshes will increase, but on the other hand it is quite possible that the computed solutions will be closer to optimal in terms of the required number of colors.

VI. CONCLUSIONS

From the above experiments results that efficient computations with diffusion-based PEA in mesh model are obtained in configurations with relatively small cyclic meshes with sufficiently large global population what is very similar result as that obtained in migration-based PEA in island model. For instance, 2×2 mesh with cross connections is equivalent to the migration model with four islands, if subpopulations in mesh nodes and on islands are of equal size.

REFERENCES

[1] Alba, E.: Parallel metaheuristic - a new class of algorithms, John Wiley & Sons, 2005. DOI: 10.1002/0471739383

[2] Alba, E.—Tomasini, M.: Parallelism and evolutionary algorithms, IEEE Trans. Evol. Comput. Vol. 6, No. 5, 2002, pp. 443–462. DOI: 10.1109/TEVC.2002.800880

[3] Bäck, T.: Evolutionary algorithms in theory and practice, Oxford University Press, 1996.

[4] Barbucha, D.—Jędrzejowicz, P.—Ratajczak, E.—Forkiewicz, M.: Population learning algorithm versus evolutionary computation, 15th International Parallel and Distributed Processing Symposium, IPDPS'2001, IEEE Comput. Society, 2001, pp. 1315–1322 (CD-ROM edition) DOI: 10.1109/IPDPS.2001.925108

[5] Bouziri, H.—Jouini, M.: A tabu search approach for the sum coloring problem, Electronic Notes in Discrete Mathematics, Vol. 36, 2012, pp. 915–922. DOI: 10.1016/j.endm.2010.05.116

[6] Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis Vol. 10, No. 2, Paris, Hermes, 1998, pp. 141–171.

[7] Cantú-Paz, E.: Topologies, migration rates, and multi–population parallel genetic algorithms, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'99, Morgan Kaufmann, 1999, pp. 91–98.

[8] Cantú-Paz, E.: Efficient and accurate parallel genetic algorithms, Kluwer, 2000.

[9] Cantú-Paz, E.—Goldberg, D. E.: Parallel genetic algorithms: theory and practice. Computer Methods in Applied Mechanics and Engineering, Elsevier, 2000. DOI: DOI: 10.1016/S0045-7825(99)00385-0

[10] Croitoriu, C.—Luchian, H.—Gheorghies, O.—Apetrei A.: A new genetic graph coloring heuristic, Computational Symposium on Graph Coloring and Generalizations COLOR'02. In: Constraint Programming, Proceedings of the International Conference, CP'02, 2002.

[11] Filho, G. R.,—Lorena, L. A. N.: Constructive genetic algorithm and column generation: an application to graph coloring, Proc. Asia Pacific Operarions Research Symposium, APORS'2000, 2000.

[12] Fleurent, C.—Ferland, J. A.: Genetic and hybrid algorithms for graph coloring, Annals of Operations Research Vol. 63, 1996, pp. 437–461. DOI: 10.1007/BF02125407

[13] Galinier, P.—Hao, J-K.: Hybrid evolutionary algorithms for graph coloring, J. Combinatorial Optimization, 1999, pp. 374–397. DOI: 0.1023/A:1009823419804

[14] Garey, R.—Johnson, D. S.: Computers and intractability. A guide to the theory of NP-completeness, Freeman, 1979.

[15] Glass, C. A.—Prügel-Bennett, A.: Genetic algorithm for graph coloring: Exploration of Galinier and Hao's algorithm, J. Combinatorial Optimization, 2003, pp. 229–236. DOI: 10.1023/A:1027312403532

[16] Hutchinson, G.: Partitioning algorithms for finite sets, Comm. ACM No. 6, 1963, pp. 613–614.

[17] Iorio, A.—Li, X.: Parameter control with a co-operative, co-evolutionary genetic algorithm, Parallel Problem Solving from Nature, Proceedings of the International Conference, PPSN'2002, LNCS Vol. 2439, 2002, pp. 247–256. DOI: 10.1007/3-540-45712-7_24

[18] Jensen, T. R.—Toft, B.: Graph coloring problems, Wiley Interscience, 1995.

[19] Johnson, D. S.,—Trick, M. A.: Cliques, coloring and satisfiability: Second DIMACS Implementation Challenge, DIMACS Series in Discr. Math. and Theor. Comp. Sc. Vol. 26, 1996.

[20] Karp, R. M.: Reducibility among combinatorial problems, In: Miller R. E. and Thatcher J. W. (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.

[21] Khuri, S.—Walters, T.—Sugono, Y.: Grouping genetic algorithm for coloring edges of graph, Proc. 2000 ACM Symposium on Applied Computing, 2000, pp. 422–427. DOI: 10.1145/335603.335880

[22] Kokosiński, Z.: Effects of versatile crossover and mutation operators on evolutionary search in partitions and permutation problems, Intelligent Information Systems: New Trends in Intelligent Information Processing and Web Mining 2005, Proceedings of the International Conference, IIS:IIPWM'2005, Advances in Soft Computing, Springer, 2005, pp. 299–308. DOI: 10.1007/3-540-32392-9_31

[23] Kokosiński, Z.—Kolodziej, M.—Kwarciany, K.: Parallel genetic algorithm for graph coloring problem, Computational Science, Proceedings of the International Conference, ICCS'2004, LNCS Vol. 3036, 2004, pp. 215–222. DOI: 10.1007/978-3-540-24685-5_27

[24] Kokosiński, Z.—Kwarciany, K.: On sum coloring of graphs with parallel genetic algorithms, Adaptive and Natural Computing Algorithms, Proceedings of the International Conference, ICANNGA'2007, LNCS Vol. 4431, 2007, pp. 211-219. DOI: 10.1007/978-3-540-71618-1_24

[25] Kokosiński, Z.: Parallel metaheuristics in graph coloring, Bulletin of the National University "Lviv Politechnic", No. 744 , 2012, pp. 209–214.

[26] Kubale, M.: Introduction to computational complexity and algorithmic graph coloring, GTN, Gdańsk, 1998.

[27] Kubale, M. (Ed.): Graph colorings, American Mathematical Society, 2004. DOI: 10.1090/conm/352

[28] Levine, D.: A parallel genetic algorithm for the set partitioning problem, Argonne Nat. Lab., Argonne, IL, 1996.

[29] Lis, J.: Parallel genetic algorithm with the dynamic control parameter, Evolutionary Computation, ICEC'1996, Proceedings of the International Conference, IEEE Computer Society, 1996, pp.324–329. DOI: 10.1109/ICEC.1996.542383

[30] Lorena, L. A. N.—Filho, G. R.: Constructive genetic algorithm for graph coloring, Proc. Asia Pacific Operarions Research Symposium APORS'97, 1997.

[31] Łukasik, S. Kokosiński Z. Świętoń G.: Parallel simulated annealing algorithm for graph coloring problem, Parallel Processing and Applied Mathematics, Proceedings of the International Conference, PPAM'2007, LNCS Vol. 4967, 2008, pp. 229-238. DOI: 10.1007/978-3-540-68111-3_25

[32] Myszkowski, P.B.: Solving scheduling problems by evolutionary algorithms for graph coloring problem, [in :] Xhafa F., Abraham A.: Metaheuristics for scheduling in industrial and manufacturing applications, Studies in Computational Intelligence, Vol. 128, 2008, pp. 145–167. DOI: 10.1007/978-3-540-78985-7_7

[33] Nowostawski, M.—Poli, R.: Parallel genetic algorithm taxonomy, Knowledge–based Intelligent Information Engineering Systems, KES'99, Proceedings of International Conference, IEEE, 1999, pp. 88–92. DOI: 10.1109/KES.1999.820127

[34] Nowostawski, M.—Poli, R.: Dynamic demes parallel genetic algorithm, Knowledge–based Intelligent Information Engineering Systems, KES'99, Proceedings of International Conference, IEEE, 1999, pp. 93–98. DOI: 10.1109/KES.1999.820128

[35] Szyfelbein, D.: Genetic algorithms for graph coloring. Neural Networks and Their Applications, Proceedings of the Conference, Polish Neural Network Society, 1999, pp. 605–610.

[36] Szyfelbein, D.: Metaheuristics in graph coloring. In: Kubale M. (Ed.): Discrete optimization. Models and methods for graph coloring, WNT, Warszawa, 2002, pp. 26–52 (in Polish).

[37] de Werra, D.: Heuristics for graph coloring, In: Tinhofer, G. et all. (Eds.) Computational graph theory, Springer–Verlag, 1990, pp. 191–208.

[38] COLOR web site.
Available at: http://mat.gsia.cmu.edu/COLOR/instances.html.

[39] DIMACS ftp site.
Available at: ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/.

[40] COLORING3 web site.
Available at: http://mat.gsia.cmu.edu/COLORING03/.