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

Zbigniew Kokosiński and Piotr Domagała
Cracow University of Technology
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, harmonic 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 assigns 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 of 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 neighboring subpopulations. There are many possible neighborhood 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 processing, 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 advantages 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 \( c : V \rightarrow \{1, \ldots, k\} \), 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 referred 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 minimum edge coloring by solving vertex coloring problem for edge graph $G_e(V_e, E_e)$ associated with the given graph $G(V, E)$. 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 $c=<c[1], c[2], \ldots, c[n]>$. 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 overlapping 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, overlapping;
- 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 information 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 between 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 recombinated 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 asynchronous 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.
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 than internal nodes of the graph \(G(V, E)\). For instance, for the square mesh of size \(4 \times 4\) the mesh contains 75% of boundary nodes. Similarly, for the mesh \(5 \times 5\) there is 64% of boundary nodes, for the mesh \(8 \times 8\) there is 43% of boundary nodes, for the mesh \(10 \times 10\) there is 36% of boundary nodes, for the mesh \(15 \times 15\) there is 25% of boundary nodes, for the mesh \(20 \times 20\) 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 \(4 \times 4\) 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 channels 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 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 assignment 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 chromosome 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).

Algorithm 2 The crossover operator CEX

Require: \( V, p, r \)
Ensure: \( s, t \)

1. \( s \leftarrow r; \ t \leftarrow p; \)
2. \( b \leftarrow x; \)
3. copy block of conflict-free vertices \( V^b_c \) from \( p \) to \( s \);
4. copy block of conflict-free vertices \( V^b_t \) from \( r \) to \( t \);

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 conflicts are marked by bold fonts. Thus, the chromosome \( p \) has 6 vertices with feasible colors and 4 vertices with color conflicts while the chromosome \( r \) 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. In each even step the maximum block is selected from the second parent. 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.

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:

\[
\text{cost}(c) = \text{conflicts}(c) \cdot \text{colors}(c),
\]

where:
- \( c \) – is the current coloring,
- \( \text{conflicts}(c) \) – is the number of conflicts in \( c \),
- \( \text{colors}(c) \) – is the number of colors in the \( c \).
In the second phase of the algorithm, for conflict–free colorings \(\text{conflicts}(c) = 0\) and \(\text{cost}(c) = 0\). Therefore, in that case the cost function is computed by the following formula:

\[
f(c) = \text{conflicts}(c) + \text{colors}(c) + p(c),
\]

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

\[
p(c) = \begin{cases} 
2 \cdot (\text{colors}(c) - \text{best}), & \text{if } \text{colors}(c) \geq \text{best} \\
0, & \text{if } \text{colors}(c) < \text{best}
\end{cases}
\]

\[
(3)
\]

and:

\(\text{best}\) – 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 tournament 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 \times 1, population size = 320, initial number of colors = 2, CEX crossover, FF mutation and Tournament selection. In diffusion PEA the mesh size = 8 \times 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 acyclic 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 \(4 \times 4\) only in 5 experiments (5/10) a conflict–free coloring was obtained and only 2 optimal colorings. For the mesh size \(6 \times 6\) 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 \(6 \times 6\) (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 conflict–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 \(14 \times 14\). 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
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, even with 1500 iterations.

For david graph the conflict–free colorings were found, even with 1500 iterations.

For david graph the conflict–free colorings were found, even with 1500 iterations.

For david graph the conflict–free colorings were found, even with 1500 iterations.
and the worst were received for $12 \times 12$ 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 $10 \times 10$ 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 $10 \times 10$ and $12 \times 12$ 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, $2 \times 2$ 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.

**TABLE III**

PEA-DM with approx. constant global population size = 700

<table>
<thead>
<tr>
<th>Graph</th>
<th>population size</th>
<th>acyclic mesh</th>
<th>cyclic mesh</th>
<th>acyclic mesh</th>
<th>cyclic mesh</th>
<th>acyclic mesh</th>
<th>cyclic mesh</th>
<th>acyclic mesh</th>
<th>cyclic mesh</th>
<th>acyclic mesh</th>
<th>cyclic mesh</th>
<th>acyclic mesh</th>
<th>cyclic mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>W</td>
<td>A</td>
<td>B</td>
<td>W</td>
<td>A</td>
<td>B</td>
<td>W</td>
<td>A</td>
<td>B</td>
<td>W</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>482</td>
<td>221.5</td>
<td>46</td>
<td>297</td>
<td>149</td>
<td>68</td>
<td>397</td>
<td>211.1</td>
<td>6</td>
<td>236.6</td>
<td>117.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>10</td>
<td>7=000</td>
<td>12</td>
<td>12</td>
<td>5=720</td>
<td>12</td>
<td>12</td>
<td>5=720</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44.7</td>
<td>314.6</td>
<td>152.2</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>12.9</td>
<td>12</td>
<td>119.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
<td>405</td>
<td>185</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>112</td>
<td>113</td>
<td>288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44.7</td>
<td>314.6</td>
<td>152.2</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>12.9</td>
<td>12</td>
<td>119.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
<td>405</td>
<td>185</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>112</td>
<td>113</td>
<td>288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44.7</td>
<td>314.6</td>
<td>152.2</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>12.9</td>
<td>12</td>
<td>119.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
<td>405</td>
<td>185</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>112</td>
<td>113</td>
<td>288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44.7</td>
<td>314.6</td>
<td>152.2</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>12.9</td>
<td>12</td>
<td>119.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
<td>405</td>
<td>185</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>68</td>
<td>380</td>
<td>217</td>
<td>112</td>
<td>113</td>
<td>288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44.7</td>
<td>314.6</td>
<td>152.2</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>63</td>
<td>352.5</td>
<td>201.5</td>
<td>12.9</td>
<td>12</td>
<td>119.4</td>
</tr>
</tbody>
</table>

References


Kubale, M. (Ed.): Graph colorings, American Mathematical Society, 2004. DOI: 10.1090/conm/552


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