Parallel Genetic Algorithm for Graph Coloring Problem

Zbigniew Kokosiński, Marcin Kołodziej, and Krzysztof Kwarciany

Faculty of Electrical & Computer Eng., Cracow University of Technology,
ul. Warszawska 24, 31-155 Kraków, Poland
zk@pk.edu.pl

Abstract. In this paper a new parallel genetic algorithm for coloring graph vertices is presented. In the algorithm we apply a migration model of parallelism and define two new recombination operators: SPPX and CEX. For comparison two recently proposed crossover operators: UISX and GPX are selected. The performance of the algorithm is verified by computer experiments on a set of standard graph coloring instances.

1 Introduction

Graph coloring problem (GCP) belongs to the class of NP–hard combinatorial optimizations problems. GCP is defined for an undirected graph as a problem of assignment of available colors to graph vertices providing that adjacent vertices are assigned different colors and the number of colors is minimal. There are many variants of the problem when some additional assumptions are made [6,10]. Intensive research conducted in this area resulted in a large number of exact and approximate algorithms, heuristics and metaheuristics [9]. GCP was the subject of Second DIMACS Implementation Challenge in 1993 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 [11,12,13].

Genetic algorithms (GA) are metaheuristics often used for GCP [3,4,5,8]. Recently a number of parallel versions of GA were studied. This approach is based on co–evolution of a number of populations that exchange genetic information during the evolution process according to a communication pattern [1,2].

In this paper we present results of our experiments with parallel genetic algorithms (PGA) for graph coloring problem.

In the paper two new recombination operators for coloring chromosomes are proposed: SPPX (Sum–Product Partition Crossover) in which simple set operations and random mechanisms are implemented, and CEX (Conflict Elimination Crossover) that is focused on the offspring quality.

In computer simulations of PGA we used DIMACS benchmarks. The obtained results are very promissing and encourage future research focused on PGA and new genetic operators for graph coloring problems.

2 Migration Model of Parallel Genetic Algorithm

There are many models of parallelism in evolutionary algorithms: master–slave PGA, migration based PGA, diffusion based PGA, PGA with overlaping subpopulations, population learning algorithm, hybrid models etc.

Migration models of PGAs consist of a finite number of subpopulations that evolve in parallel on their ”islands” and only occasionaly exchange the genetic information under the 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 our 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 their 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. The distance between migrations determines how often the migration phase of the algorithm occurs. Two migration schemes are applied: migration of best individuals of the subpopulation or migration of individuals randomly selected.

In our algorithm we applied a specific model of migration in which islands use two copies of genetic information: migrating individuals still remain members of their original subpopulation. In other words they receive new ”citizenship” without losing the former one. Incoming individuals replace the chromosomes of host subpopulation at random. Then, a selection process is performed. The rationale behind such a model is as follows. Even if the best chromosomes of host subpopulation are eliminated they shall survive on other islands where their copies were sent. On the other hand any eliticist scheme or preselection applied to the replacement phase leads to premature elimination of worse individuals and lowers the overall diversity of subpopulation.

3 Genetic Operators for GCP

In graph coloring problem k–colorings of graph vertices are encoded in chromosomes representing set partitions with exactly k blocks. In partition representation each block of partition does correspond to a single color. In assignement representation available colors are assigned to an ordered sequence of graph vertices. In this section we introduce a collection of genetic crossover, mutation and selection operators that are used in our PGA.

3.1 Sum–Product Partition Crossover

The first recombination operator called Sum–Product Partition Crossover (SPPX) employs for offspring generation simple set sum and set product operations on block of partitions and a random mechanism of operand selection

```
procedure: SPPX (p1,p2,r1,r2,s1,s2,t1,t2,Prob(PRODUCT),Prob(SUM))
begin
  s1=t1=s2=t2=$\emptyset$;
  generate random numbers rand1, rand2 : 0 $\leq$rand1,rand2$\leq$1;
  if rand1 $\leq$ Prob(PRODUCT) then PRODUCT(p1,r1,s1,t1);
  if rand2 $\leq$ Prob(SUM) then SUM(p2,r2,s2,t2);
end SPPX;
PRODUCT(p,r,s,t)
begin
  select random h (1$\leq$h$\leq$k) and j (1$\leq$j$\leq$l);
  $V_1^s = V_1^t = (V_h^p \cap V_j^r)$;
  for i = 1 to k do
    if i $\neq$ h do if ($V_i^p \setminus V_1^s$) nonempty then
      add next block $V_i^p \setminus V_1^s$ to s;
  for i = 1 to l do
    if i $\neq$ j do if ($V_i^r \setminus V_1^t$) nonempty then
      add next block $V_i^r \setminus V_1^t$ to t;
end PRODUCT;
SUM (p,r,s,t)
begin
  select random h (1$\leq$h$\leq$k) and j (1$\leq$j$\leq$l);
  $V_1^s = V_1^t = (V_h^p \cup V_j^r)$;
  for i = 1 to k do
    if i $\neq$ h do if ($V_i^p \setminus V_j^r$) nonempty then
      add next block $V_i^p \setminus V_j^r$ to s;
  for i = 1 to l do
    if i $\neq$ j do if ($V_i^r \setminus V_h^p$) nonempty then
      add next block $V_i^r \setminus V_h^p$ to t;
end SUM;
```

Fig. 1. The recombination operator SPPX.

from randomly determined 4 parental chromosomes. SPPX is composed of two procedures PRODUCT and SUM which are applied to the pair of chromosomes p=$\{V_1^p, \ldots, V_k^p\}$, r=$\{V_1^r, \ldots, V_l^r\}$ and produce a pair of chromosomes s=$\{V_1^s, \ldots, V_m^s\}$ and t=$\{V_1^t, \ldots, V_n^t\}$ with probabilities of elementary operations satisfying: $0 \leq$ Prob(PRODUCT) $<$ Prob(SUM) ≤ 1. A pseudocode of the procedure SPPX is presented in Fig.1.

Example1

Four parents represent different 3–colorings of a graph with 10 vertices: p1={ABC,DEFG,HIJ}, r1={CDEG,AFI,BHJ}, p2={CDG,BEHJ,AFI} and r2={ACGH,BDFI,EJ}. Let us assume Prob(PRODUCT)=0.5, Prob(SUM)=0.7. Let rand1=0.4 and PRODUCT is computed with h=3, j=2. Thus, V_3^p={HIJ} and V_2^r={AFI}. We obtain $V_1^s = V_1^t =$ {I} and s1={I,ABC,DEFG,HJ} and t1={I,CDEG,AF,BHJ}. Let rand2=0.3 and SUM is computed with h=2, j=1. Thus, V_2^p={BEHJ} and V_1^r={ACGH}. We obtain $V_1^s = V_1^t =$ {ABCEGHJ} and then s2={ABCEGHJ,D,FI} and t2={ABCEGHJ,DFI}. As a result of the crossover we obtain four children: s1, t1, s2 and t2 representing 2,3 and 4–colorings of the given graph.

Let us notice that operation PRODUCT may increase the initial number of colors while the operation SUM may reduce this number. The probability of PRODUCT should be lower then the probability of SUM. The recombination operator SPPX which can be used as a versatile operator in evolutionary algorithms for many other partition problems.

3.2 Conflict Elimination Crossover

In conflict–based crossovers for GCP an assignement representation of colorings is used and the offspring try to copy conflict–free colors from their parents. The next recombination operator called Conflict Elimination Crossover (CEX) reveals some similarity to the classical crossover. Each parental chromosome 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 is not working. Therefore, the recombination must be combined with an efficient mutation mechanism. The operator CEX is almost as simple and easy to implement as the classical crossover (see Fig.2).

```
procedure: CEX (p,r,s,t)
begin
  s = r;
  t = p;
  copy conflict-free vertices $V_{cf}^{p}$ from p to s;
  copy conflict-free vertices $V_{cf}^{r}$ from r to t;
end
```

Fig. 2. The recombination operator CEX.

Example2

Two parents represent different 5–colorings of a graph with 10 vertices i.e. sequences: p=<5,2,**3**,**1**,1,**4**,3,5,1,**2**> and r=<1,4,**5**,2,3,3,**2**,4,2,**1**>. Vertices with conflict colors are marked by bold fonts.

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

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

3.3 Union Independent Set Crossover

The greedy operator proposed by Dorne and Hao [4] and called Union Independent Sets (UISX) works on pairs of independent sets taken from two parent colorings. In any feasible graph coloring all graph vertices are partitioned into blocks that are disjoint independent sets (IS). A coloring is not feasible if it contains at least one block which is a non–independent set. Each block of a partition is assigned one color. "If we try to maximize the size of each IS by a combination mechanism, we will reduce the sizes of non–independent sets, which in turn helps to push these sets into independent sets" [4].

In the initial step disjoint ISs in both parents are determined. Then we compute coloring for the first child. At first, we select the maximum IS from the first parent and compute set intersections with ISs from the second parent. The union of a pair of ISs with maximum intersection is colored in the offspring with the IS color from the first parent. In the case of a tie a random IS is always chosen. Then the colored vertices are removed from the both parents and the coloring procedure is repeated as long as posible. The vertices without any color are assigned the original color from the first parent. The coloring for the second child is computed with reversed roles of both parents.

3.4 Greedy Partition Crossover

The method called Greedy Partition Crossover (GPX) was designed by Galinier and Hao [5] for recombination of colorings or partial colorings in partition representation. 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 we select the maximum block from the first parent. Then, we add the block to the result and remove all its nodes from the both parents. In each even step we select the maximum block from the second parent. Then, we add the block to the result and remove all its nodes from the both parents. The procedure is repeated at most k times since, in some cases, the offspring has less blocks then the parents (see Example 3). This possibility is not considered in the original paper [5]. Finally, unassigned vertices (if they exist) are assigned at random to existing blocks of partition. A corrected version of GPX is shown in Fig.3.

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

Example 3

Two parents represent different 3–colorings of a graph with 10 vertices, i.e. partitions: p0={ABFGI,CDE,HJ}, p1={ABF,CDEGHJ,I}.

```
procedure: GPX (p0,p1,s)
begin
  s = $\emptyset$;
  i = 1;
  repeat
    select block $V$ with maximum cardinality from the partition p(i mod2);
    s = s$\cup V$ -- add the block $V$ to partition s;
    remove all vertices of $V$ from p0 and p1;
    i = i + 1;
  until (i $>$ k) or (all blocks of p1 and p2 empty);
  assign randomly all unassigned vertices to blocks of s;
end
```

Fig. 3. The modified recombination operator GPX.

For i=1 the maximum block {CDEGHJ} is selected from p1 and is added to s. After removing the block vertices from the parents we obtain p0={ABFI}, p1={ABF,I}. For i=2 the maximum block {ABFI} is selected from p0 and is added to s. Termination condition is satisfied and we obtain result partition s={ABFI,CDEGHJ} that is a valid 2–coloring.

3.5 Mutation Operators

Transposition is a classical type of mutation that exchange colors of two randomly selected vertices in the assignment representation. The second mutation operation called First Fit is designed for colorings in partition representationand and is well suited for GCP. In the mutation First Fit 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.

3.6 Selection Operator

The quality of a solution is measured by the following cost function:

$$f(p) = \sum_{(u,v)\in E} q(u,v) + d + C \text{ , where:}$$

p – is a graph coloring,
q – is a penalty function for pairs of vertices connected by an edge (u,v) $\in$ E:
  $q(u,v) = 2$ when $c(u) = c(v)$, and $q(u,v) = 0$, otherwise,
d – is a general penalty function applied to graph colorings:
  $d = 1$, when $\sum_{(u,v)\in E} q(u,v) > 0$, and $d = 0$ when $\sum_{(u,v)\in E} q(u,v) = 0$,
C – is the number of colors used.

The proportional selection is performed in our PGA with the fitness function $1/f(p)$.

Table 1. Performance of the migration–based PGA with various crossover operators

no	graph	vertices	edges	colors	Crossover operator							
					UISX		GPX		SPPX		CEX	
					n	t[s]	n	t[s]	n	t[s]	n	t[s]
1	anna	138	493	11	19	2.0	21	6.0	61	3,2	15	1.3
2	david	87	406	11	22	2.0	24	3,9	74	3,6	20	1.0
3	huck	74	301	11	8	0,8	7	1.0	29	0,5	12	0.7
4	miles500	128	1170	20	95	9,5	59	40	152	38	100	3.0
5	myciel7	191	2360	8	18	2.0	21	7.9	76	7.6	20	1.0
6	mulsol$_1$	197	3925	49	90	31	60	35	180	34	58	2.0

4 Experimental Verification

For computer experiments we used graph coloring instances available in the web archive [11]. This is a collection of graphs in DIMACS format with known parameters, including graph chromatic numbers.

In our computer program *PGA for GCP* two basic models of PGA: migration and master–slave can be simulated. It is possible to set up most parameters of evolution, monitor evolution process and measure both the number of generations and time of computations. In the preprocessing phase we converted list of edges representation into adjacency matrix representation. The program generates detailed reports and basic statistics.

We tested the influence of migration scheme on the PGA efficiency measured by the number of generations n needed to obtain an optimal coloring when the chromatic number is known. Computations were performed on 5 graphs with the following parameters: *population size* = 60 , *number of islands* = 5, *migration rate* = 5, *crossover* = SPPX, *mutation* = First Fit, and *mutation probability* = 0.1 . All experiments were repeated 30 times. For all graphs migration of best individuals always gives the best results. Migration of random individuals is almost useless except huge graphs like mulsol.i.4 where random migration is also efficient.

The experiments confirmed that the mutation First Fit is superior to the Transposition mutation for graph coloring problems. It works particularly well with CEX crossover.

In the main experiment the efficiency of all 4 crossover operators was tested in the migration model. Computations were performed on 6 graphs with the parameters: *population size* = 60 , *number of islands* = 3, *migration rate* = 5, *migration size* = 5, *mutation* = First Fit, and *mutation probability* = 0.1 . All experiments were repeated 30 times. The results are presented in Table 1.

We can observe that the proposed crossover operators are efficient in terms of computation time. SPPX requires more generations then GPX in order to find an optimal coloring but it is simpler and therefore a bit faster then GPX. In some cases the operator UISX requires less generations then GPX but it always produces an optimal solution faster then two previous operators. The most efficient operator in the experiment is CEX which dominates all others operators under the both criteria (except the smallest graph instance).

All computer experiments were performed on a computer with AMD Athlon 1700+ processor (1472 MHz) and 256 MB RAM.

5 Conclusions

In the paper we proved by computer simulation that parallel genetic algorithms can be efficiently used for the class of graph coloring problems. In island model of PGA the searched space is significantly enlarged and the migration between co–evolving subpopulations improves the overall convergence of the algorithm.

PGA is particularly efficient for large scale problems like GCP. The results presented in this paper encourage further research in this area. The authors intend to continue their work. One obvious direction is to extend the experiments on other DIMACS benchmarks including a class of random graphs. It is also worth to consider some variants of SPPX operator that will make it more problem–oriented. The search for new efficient genetic operators for GCP still remains an open question.

References

1. Alba E., Tomasini M.: Parallelism and evolutionary algorithms, IEEE Trans. Evol. Comput. Vol.6 (2002) No.5, 443–462
2. Bäck T.: Evolutionary algorithms in theory and practice, Oxford U. Press (1996)
3. Croitoriu C., Luchian H., Gheorghies O., Apetrei A.: A new genetic graph coloring heuristic, Computational Symposium on Graph Coloring and Generalizations COLOR'02, [in:] Proc. Int. Conf. Constraint Programming CP'02 (2002)
4. Dorne R., Hao J-K.: A new genetic local search for graph coloring, Parallel Problem Solving from Nature 1998, LNCS 1498 (1998) 745–754
5. Galinier P., Hao J-K.: Hybrid evolutionary algorithms for graph coloring, J. Combinatorial Optimization (1999) 374–397
6. Jensen T.R., Toft B.: Graph coloring problems, Wiley Interscience (1995)
7. Johnson D.S., Trick M.A.: Cliques, coloring and satisfiability: 2nd DIMACS Impl. Challenge, DIMACS Series in Discr. Math. and Theor. Comp. Sc. Vol.26 (1996)
8. Khuri S., Walters T. Sugono Y.: Grouping genetic algorithm for coloring edges of graph, Proc. 2000 ACM Symposium on Applied Computing (2000) 422–427
9. Kubale M.: Introduction to computational complexity and algorithmic graph coloring, GTN, Gdańsk (1998)
10. Kubale M. (ed): Discrete optimization. Models and methods for graph coloring, WNT, Warszawa (2002) (in Polish)
11. http://mat.gsia.cmu.edu/COLOR/instances.html
12. ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/
13. http://mat.gsia.cmu.edu/COLORING03/