ON WORST-CASE AND COMPARATIVE ANALYSIS AS DESIGN PRINCIPLES FOR EFFICIENT RECOMBINATION OPERATORS: A GRAPH COLORING CASE STUDY

Pablo E. Coll, Guillermo A. Durán
Departamento de Computación, FCEyN, U.B.A
Ciudad Universitaria, (1428) Cap. Fed., ARGENTINA
email: pecoll{willy}@dc.uba.ar

Pablo Moscato
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Sistemas
Universidade Estadual de Campinas
Caixa Postal 6101 - Campinas, SP - CEP: 13081-970, BRAZIL
email: moscato@densis.fee.unicamp.br
http://www.densis.fee.unicamp.br/~moscato

Introduction, Motivation and Previous Work

In 1995, Costa, Hertz and Dubuis, introduced a new metaheuristic for graph coloring problems [3] and they named it EDM (for Evolutionary Descent Method). Graph coloring is a problem with many applications and it naturally arises in a variety of different areas like scheduling, assignments and timetabling and it is associated to other problems in the NP-Hard class. The graph coloring problem arises in classical areas of Graph Theory like finding the minimum number of colors to color a given map, scheduling of exams in an university (see also Ref. [20]), as well as others coming from the advances of technology like frequency assignment of TV broadcasting stations to channels and to model problems arising in the development of efficient compilers for computer programs.

The metaheuristic can be classified as a *memetic algorithm* since it is based on a population search in which periods of local optimization are interspersed with phases in which new configurations are created from earlier, well-developed configurations or local minima of the previous iterative improvement process. The new population is created using *crossover* or *recombination* operators as in genetic algorithms. In this chapter we will discuss how a methodology inspired in *Competitive Analysis* and the recently introduced *Comparative Analysis* can lead to tight theoretical bounds and also be relevant to the problem of the design of recombination operators with better worst-case performance.

The term *"memetic"* was introduced in 1989 [8] to encompass a class of metaheuristics for combinatorial optimization problems which are based on the use of a population of "agents" engaged in periods of local-search-based optimization interspersed with phases in which new points in configuration space are created using crossover or recombination operators (see [20], [19], [8], [9], [10], [11], and [12]). They are also known as *hybrid genetic algorithms*, although the former denomination is preferred to emphasize the difference with standard genetic algorithms. The latter do not include local search or other forms of *a priori* knowledge of the problem at hand.

Costa and coworkers remarked that their method: *"differs from most of the hybrid algorithms which have been recently developed in the sense that it uses a simple descent method instead of a refined sequential method which accepts non-improvement moves"*. Although [7] is referenced as an example of such a method (those which accept non-

improvement moves at the local optimization phases), in Ref. [7] the *binary perceptron learning* problem is first addressed with a simple descent method (see also [4]). Regardless these minor comments, the techniques have many analogies with new *"hybrid"* genetic algorithms as well as former methods like variants of *Scatter Search* introduced by F. Glover ([6]). New results on memetic algorithms for timetabling problems can be found in Refs. [19], [20].

It is important to mention at this point that a WWW home page for memetic algorithms, with links to papers and different research groups working in related issues, can be found at: **`http://www.densis.fee.unicamp.br/~moscato/memetic_home.html`**. The page contains links to many on-line references and papers cited in this chapter.

The impact that a suitable, custom design of the recombination operator has on the overall performance of memetic and genetic approaches had been previously identified and it was well recognized in Refs. [5], [7], [8], [11], and [12]. However, we face an intrinsic practical difficulty in evaluating the effect that the recombination has by itself and to disassociate its benefits from the other components of a complex metaheuristic. Sometimes the recombination is just a small part of a highly complex algorithmic framework and sometimes, frustrating the hope for a more quantitative treatment of the issue, it seems that some researchers even enjoy having dozens of free parameters they fix on the basis of some preliminary computational experiments. All these facts finally conspire, they foil and disable us from achieving a proper comparison of results. Moreover, they do not allow a clear understanding of the key beneficial mechanisms of each implementation. This said, a good recombination may be part of a complex algorithmic framework and it seems to be very hard to find a reasonable way to evaluate its performance against a different recombination if we can only rely on the results of computer experiments.

The main problem we face is that we are making a comparison which may be biased by many other algorithmic decisions in a memetic framework like the type of local search used, the neighborhood definition for moves among different configurations, the different parameters used, etc. A similar line of argument can be established for implementations of genetic algorithms, since they are not free of other *ad-hoc* parameterized decisions like the rate of mutation, probabilities for selection for recombination, mating strategies, avoidance of inbreeding solutions, etc. Indeed, these problems are part of the still not developed methodological procedures related to the proper performance analysis of metaheuristics techniques. However, we should mention at least one good step forward in this direction regarding memetic algorithms. It was the work on the Euclidean Traveling Salesman Problem (ETSP) by Reimar Hofmann [5] followed by Nick Radcliffe and Patrick Surry [12]. Their work, although it also relied on experimental issues, was based on the use of *Forma Analysis* and *correlation within formae experiments* using solved instances of the ETSP. This technique, which still relies on computational experimentation, attempted to isolate the benefits of recombination *per se*. The practicality of these approaches had been anticipated in [8] and [7].

In this chapter, we present a novel methodology for the analysis and the design strategy of recombination operators. As a case study, which will illustrate the technique, we will discuss the principal recombination of the algorithm for the graph coloring problem introduced in [3]**.** Part of our motivation is based on the good results obtained by Costa and coworkers on large random graphs of the class $G_{n,p}$ (graphs with n vertices and density p) as well as some of the limitations of their method. Our approach to the evaluation of the performance of a recombination is inspired on *Competitive Analysis* and *Comparative Analysis* [21], two techniques developed in connection with *on-line* algorithms. In the next section we will discuss some of the main aspects of the heuristic proposed in [3] and with more detail the

recombination procedure we will analyze. In a later section we will discuss in some detail the reasons which lead us to propose this approach.

The recombination procedure and associated definitions

In order to understand the problem and the recombination procedure we need to introduce some definitions. Our notation convention will follow that of [3] for clarity. Let $G=(V,E)$ be an undirected graph of vertex set V and edge set E. An *independent set* is a set $I \subseteq V$ such that every pair in I is not adjacent in G, i.e. $\{x,y\} \subset I \Rightarrow (x,y) \notin E$. A *partial q-coloring of G* is a partition $(V_1, V_2, \ldots, V_q)$ of a subset $V' \subseteq V$ in q disjoint independent sets. If we assign a color $c(x)=i$ to each vertex $x \in V_i$ then a *partial coloring with q colors* can be interpreted as the problem of coloring with q colors a set $V' \subseteq V$ such that no two adjacent vertices have the same color. The *dimension of a partial coloring with q colors* is the cardinality of the set $V'=V_1 \cup V_2 \ldots \cup V_q$. A *partial coloring with q colors of dimension* $n = |V|$ is known as *a q- coloring of G*. The associated optimization problem is the task of finding a coloring with the minimum q. The *chromatic number* of a graph is the minimum q for which a q-coloring exists.

The algorithm introduced by Costa, Hertz and Dubuis has two phases that can be included in a memetic framework. The first phase tries to find a partial q'-coloring. To achieve that goal, a recombination procedure is used and combined with local search descent steps. The second phase, which is the essential part of the algorithm, tries to find a feasible q-coloring of the yet uncolored graph. It is this recombination procedure the one we are analyzing in this chapter. Clearly, the algorithm would use $q+q'$ colors to finally produce a coloring in this way. In [3], the authors claim that they need to use the two-phase approach since they believe (based on their own experiments and other work which exists in the literature) that they would not be able to find a feasible q-coloring using only the second phase. At least, they find it really improbable when addressing problems with more than 300 nodes.

The recombination procedure we are going to analyze has as input two parent colorings S_1 and S_2 of a graph $G(V,E)$ which are not necessarily feasible. That is, it is accepted the existence of *conflicting edges,* that is edges which have both extremes colored using the same color. Let $c_i(v)$ be the color assigned to vertex v in the coloring S_i and let $NCE_i(v,d)$ be the *number of conflicting edges* which are at a distance d from v in S_i . To each vertex v a penalty $p_i(v)$ is associated with v which measures how *"close"* vertex v is to conflicting edges in S_i. More explicitly

$$p_i(v) = \sum_{d=0}^{2} \omega_d NCE_i(v,d) \qquad (1)$$

where ω_d are *"weights"* which balance the importance of conflicting edges located at different distances from vertex v. Costa and coworkers have decided that ($\omega_0 > \omega_1 > \omega_2$) and not to take into account the cases where $d > 2$ regarding some preliminary, qualitative "cost/benefit" computational experiments they performed. From two "*parent"* colorings S_1 and S_2, one *"child"* coloring S_3 will be created such that each vertex $v \in S_3$ will be colored either with $c_1(v)$ or $c_2(v)$ and preference is given to the color with the smallest penalty $p_i(v)$ in S_1 or S_2. When $p_1(v) = p_2(v)$, we will consider the vertices of v which have been already colored in the current partial coloring of S_3 choosing the color $c \in \{c_1(v), c_2(v)\}$ which minimizes $nadj(v,c)$, the number of adjacent vertices to v colored c in S_3. If we can not decide between $c_1(v)$ and $c_2(v)$ then we break ties choosing between them at random with the same probability. The following pseudocode from [3] would certainly help to clarify the procedure.

Input: $S_1, S_2 \in X$; **Output:** $S_3 = (V_1, \ldots, V_q) \in X$
$nadj(v,c) = 0$; $\forall v \in V$, $\forall c=1,\ldots, q$;
$V_C := \varnothing$; $\forall c=1,\ldots, q$;
for (each vertex $v \in V$) **do**
  if $p_1(v) < p_2(v)$ **then** $c_3(v)=c_1(v)$
  else if $p_2(v) < p_1(v)$ **then** $c_3(v)=c_2(v)$
    else (* $p_1(v) = p_2(v)$ *and we need to break ties* *)
      if $nadj(v,c_1(v)) < nadj(v,c_2(v))$ **then** $c_3(v)=c_1(v)$
      else if $nadj(v,c_2(v)) < nadj(v,c_1(v))$ **then** $c_3(v)=c_2(v)$
        else (* *then* $nadj(v,c_1(v))=nadj(v,c_2(v))$ *)
          (* *now we break ties at random* *)
          $c_3(v)=random(c_1(v),c_2(v),0.5)$;
  $nadj(v',c_3(v))=nadj(v',c_3(v))+1$ $\forall v'$ adjacent to v;
  $V_{c_3(v)} = V_{c_3(v)} \cup \{v\}$

To be fair, this randomized procedure is actually defining not one but a "family" of recombinations since it has ω_0, ω_1 and ω_2 as free parameters. After a series of preliminary computational experiments with 20 graphs belonging to the class $G_{100,0.5}$, Costa and co-workers have observed that the best results have been obtained using $\omega_0 = 100$, $\omega_1 = 3$ and $\omega_2 = 1$. Undoubtedly, this selection leaves many open questions since it may be the case that this values are *instance-dependent*, that is they may be optimal just for the input distributions of the instances used in their computational study. Finite size effect associated to the use of $|V|=100$ might have also influentiated this selection.

A "competitive-inspired" analysis

We will discuss two cases, where the principal recombination introduced in [3] is studied. The analysis is inspired by the concept of *Competitive Analysis* which was first introduced to analyze the performance of *on-line* algorithms.

A problem is said to be *on-line* if it requires that irrevocable decisions which influence the output must be made before having a complete knowledge of the entire input. Some examples of this class of problems can be found in robot motion planning, maintaining dynamic data structures, video on demand, network routing, etc. These problems are a real challenge to standard worst-case analysis since, after examination of the characteristics of an algorithmic procedure an *"adversary* " can choose an input sequence which can foil the performance of the on-line algorithm.

One way to overcome the problem of dealing with the distribution of inputs and still make a relevant worst-case analysis is to use the method known as *Competitive Analysis*, developed by Sleator and Tarjan [13] although the concept can be found in the earlier literature of bin packing [14], [15]. The key idea is to compare an on-line algorithm with the optimal *off-line* algorithm, i.e. one that can see the entire input in advance thus it will have a complete knowledge of the future events. This comparison is done on an input-by-input basis. If we denote as A an on-line algorithm and ξ an input sequence we denote with $A(\xi)$ the cost of algorithm A on ξ. If $A_{opt}(\xi)$ is the cost of the optimal off-line algorithm on input ξ then we say that A is a *k-competitive algorithm* if for all ξ, $A(\xi) - k\, A_{opt}(\xi)$ remains bounded by a constant. The term k-competitive was coined in [16]. For a randomized algorithm A is replaced by $\langle A(\xi)\rangle_a$, which stands for the expectation value over A's random choices. The *competitiveness of A*, denoted β_A is the infimum of k such that A is k-competitive.

Under this perspective, the quality of a specific algorithm is given by the maximum ratio between the cost of an on-line problem and the cost incurred by the optimal algorithm for the off-line problem. Competitive analysis, although it is still a kind of worst-case analysis, implied a conceptual break-through, since by the definition of competitiveness it liberates from any kind of probabilistic assumptions about the input data. It allowed to view an on-line problem as a game between an *on-line player* and an *adversary* who will chose the input of the problem so as to maximize the ratio between the algorithm's cost and that of the optimal off-line algorithm [17].

With this concepts in mind, we will give an example of the kind of analysis we propose for recombination design; we will act as an adversary of the recombination we want to study. We will first consider the problem of coloring $K_{6,6}$, a bipartite complete graph with six vertices in each partition. The vertices are numbered such that $\{v_1,\ldots,v_6\}$ and $\{v_7,\ldots,v_{12}\}$ belong to the first and the second partition, respectively. Note that the recombination of [3] specifies a number of operations which are executed in an order given by the vertex numbering. This said, thinking as a game in which we are the adversaries, we have chosen the input graph and the sequence of events. The input to the recombination procedure will be two parent configurations S_1 and S_2. with no conflicting edges, two feasible colorings. Vertices in each partition have the same colors, and we will also require that $c_1(v_1)=c_2(v_7)=B$ (B for Blue) and $c_1(v_7)=c_2(v_1)=R$ (R for Red). A natural cost measure of the performance of the recombination would be the number of conflicting edges present in the child coloring.

It is obvious to remark that the chromatic number for $K_{6,6}$ is two, so the discussion may look irrelevant since it may seem that it has no sense to cross two optimal solutions. However, the example must be regarded as a subgraph of a larger graph. The decision to work out an example case for only two colors is motivated by the fact that the subgraph can be part of a well-developed configuration since those are the ones engaged in recombination in a memetic algorithm. We will see for this example how the recombination generates a *"child"* S_3 from two *"parent"* colorings S_1 and S_2. For all v in V, $p_1(v) = p_2(v) = 0$, so we need to take into account $nadj(v, c_i(v))$ to break ties. Starting from v_1, clearly $nadj(v_1, c_1(v_1)) = nadj(v_1, c_2(v_1))=0$, thus we need to arbitrarily select the color of v_1 in S_3. The same situation will occur for all vertices in the same partition $v_1,v_2,\ldots,v_6$ if we follow the index order in the **for** loop of the pseudocode which defines the recombination . Due to the randomized procedure, we may end up with a worst-case scenario in which we have $c_3(v_1) = c_3(v_2) = c_3(v_3) = B$ and $c_3(v_4) = c_3(v_5) = c_3(v_6) = R$. When we attempt to color vertices v_7 to v_{12} we note that $nadj(v_j,c_1(v_j))= nadj(v_j,c_2(v_j))=3$ for $7 \leq j \leq 12$, so we must break ties again to color it. Again, it may be possible that $c_3(v_7) = c_3(v_8) = c_3(v_9) = B$ and $c_3(v_{10}) = c_3(v_{11}) = c_3(v_{12}) = R$. In conclusion, starting from two colorings of a graph with 36 edges without conflicts we end up having a new coloring S_3 which has 18 conflicting edges ($|E|/2$ in this case), (see Figure 1).

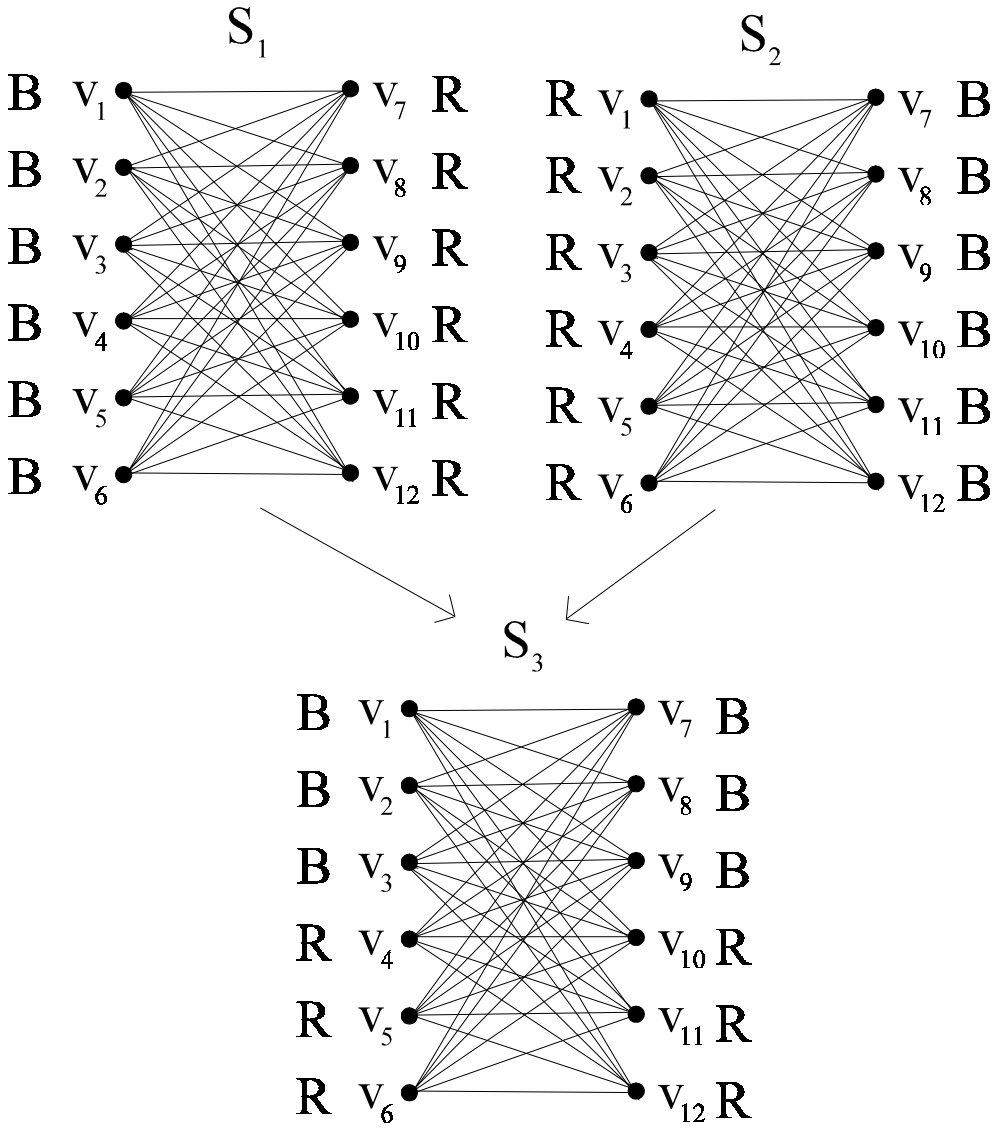


Figure 1: The figure shows two parent colorings S_1 and S_2 and one child coloring S_3 which can be the result by the application of the graph coloring recombination introduced in Ref. [3]. For this particular instance of the problem, two feasible colorings (without conflicting edges) can create an output that has half of the edges in conflict.

It is clear that the situation described above has already identified the problem of creating new configurations which have at least $|E|/2$ new conflicting edges even when coming from two well developed solutions. This may constitute a simple instance of a more general case. Let $K_{j1, \ldots, jk}$ be a k-partite complete graph such that $j_1 = j_2 = \ldots = j_k = j$. To simplify the discussion we will suppose that j and k are even. We will start again with a situation in which S_1 and S_2 have no conflicting edges. Let V_i be the i^{th} partition, ($|V_i| = j_i = j$). Let $c_1(v_1) = 1 + \mathbf{int}((l-1)/j)$ and $c_2(v_1) = k+1 - c_1(v_1)$ where $\mathbf{int}(x)$ is the integer part of x. In this case the chromatic number for this graph is k but again we must recall that it can be regarded as a subgraph of a larger graph which is the one we would like to color. Again to generate S_3, for all v in V, $p_1(v) = p_2(v) = 0$, since the two parents have no conflicting edges. Starting from v_1, clearly $nadj(v_1, c_1(v_1)) = nadj(v_1, c_2(v_1)) = 0$ with $c_1(v_1)=1$ and $c_2(v_1)=k$, thus we need to arbitrarily select among them. The same situation will occur for all vertices $v_2,\ldots,v_j \in V_1$ if we

follow the index order. Due to randomization, we may end up with a worst-case scenario in which we have $c_3(v_1) = c_3(v_2) = \ldots = c_2(v_{(j/2)}) = c_1(v_1) = 1$ and $c_2(v_{(j/2)+1}) = \ldots = c_3(v_j) = c_2(v_{(j/2)+1}) = k$. Again, due to the randomization procedure we may end up with a situation in which the first half of the vertices of partition $|V_i|$ are colored with color i in S_3 and the second half is colored with color $k-i+1$ (note that for v_l with $l>kj/2$, the argument still holds but with $nadj(v_1, c_i(v_1)) = j/2$). In conclusion, starting with two colorings of $K_{j,\ldots,j}$, a k-partite complete graph with $|V|=kj$ vertices and $|E|=|V| \times(|V|-j)/2$ edges which have no conflicting edges, we end up having a new coloring S_3 which has $|E|/(2(k-1))$ conflicting edges. This number is $O(|E|)$ for "small" k and $O(|V|)$ for larger values of k. For $k=2$ we recover the result we presented in the first example.

Upper bound

In this section we will prove that the first example of the preceding section is a worst-case example for the principal recombination of [3].

Lemma:

Given a graph $G(V,E)$ and two feasible q-colorings of G, S_1 and S_2, let S_3 be the q-coloring of G (not necessarily feasible) created by the recombination of [3]. For a node $v \in V$ we denote as $NCE_3(v)$ the number of conflicting edges created in S_3 at the moment of coloring node v with either $c_1(v)$ or $c_2(v)$, the color assigned to vertex v in the colorings S_1 and S_2, respectively. We will denote as $|AC(v)|$ the number of nodes adjacent to node v in G which have already been chosen to coloring in S_3 when we are coloring node v. Then

$$NCE_3(v) \leq |AC(v)|/2$$

Proof:

Let $c_1(v)$, $c_2(v)$ the colors assigned to node v in S_1 and S_2 respectively. Let's suppose without loosing generality, that node v will be colored as in parent coloring S_1, that is $c_3(v) = c_1(v) = R$. We face two possibilities here. Let's first consider what happens when $c_2(v) = c_1(v) = R$. In this case, we are sure that no other vertex adjacent to vertex v has been colored with color R since by hypothesis we have assumed that both S_1 and S_2 are feasible colorings, then they have no conflicting edges. Then $NCE_3(v) = 0$, and, since $|AC(v)| \geq 0$ then $NCE_3(v) \leq |AC(v)|/2$ which proves the result in this case. Now let's consider what happens when $c_2(v) \neq c_1(v)$. We will suppose that $NCE_3(v) \geq \mathbf{int}(|AC(v)|/2)+1$, where $\mathbf{int}(x)$ represents the integer part of the argument x, and we will prove the sought result via *reductio-ad-absurdum*. First note that $p_1(v) = p_2(v)=0$, since S_1 and S_2 are two *feasible* colorings of G. Then the coloring of v in S_3 was decided as a consequence of two possibilities according the recombination of [3]:

1. $nadj(v,c_1(v)) < nadj(v,c_2(v))$, or
2. $nadj(v,c_1(v)) = nadj(v,c_2(v))$,

and the selection of $c_3(v)$ was made at random with probability 0.5.

Since we have supposed that $NCE_3(v) \geq \mathbf{int}(|AC(v)|/2)+1$ then $nadj(v,c1(v)) > \Sigma_X nadj(v,X)$ where X runs through all the colors. Since $\Sigma_X nadj(v,X) \geq nadj(v,c_2(v))$, we get $nadj(v,c_1(v)) \geq nadj(v,c_2(v))$ which contradicts the two cases above. This finally proves the lemma.

Now we are in condition of proving the following worst-case upper bound.

Theorem:

Given a graph $G(V,E)$ and two parent colorings as defined in the lemma above, the number of conflicting edges (NCE_3) in the final coloring S_3 generated by the recombination of Ref. [3], is bounded by $|E|/2$.

Proof: It immediately follows from the Lemma.

Since $NCE_3 = \Sigma_{i=1}^{|V|} NCE_3(v_i) \leq \Sigma_{i=1}^{|V|} |AC(v_i)|/2 = |E|/2$. ∈

A weaker adversary

In our previous analysis we have used randomization for the benefits of our thesis. We have shown that we can get an extremely bad result due to the application of the standard procedure given in Ref. [3]. An open question that motivates this section is: How relevant where these random choices ? How well would the recombination procedure do if it would not face such an extremely ``unfair'' sequence of events ?

In certain way, we are making an analysis with a weaker adversary, one that can not control at will those random choices. However, the order in which the vertex are going to be colored is at the control of the adversary and it will be the same we have used in the previous worst-case analysis. In a certain way, we see some analogies with the relevance that knowing the future has for certain on-line problems [18].

In our previous analysis, the selection of colors for vertices in S_3 was random, but we were always choosing among the two possibilities the one which would be more "unfair" for the recombination of [3]. We will now analyze what happens when the selection of which color to use (B or R) is equiprobable (that is, it has probability of selection 0.5), whenever ties must be broken. It is a way of relaxing the worst case scenario by reducing the "strength" of the adversary (using jargon from Competitive Analysis).

We will maintain the same numbering order for the vertices. We want to calculate the expected value of conflicting edges in S_3 for a $K_{2t,\,2t}$ colored with two colors, where each node (whenever ties must be broken) can take color c_1 with probability 0.5. Colors start to be assigned to vertices starting with v_1 and following the same indexing order we used before. This leads to the fact that the $V_1 = \{ v_1 , v_2 ,..., v_{2t} \}$ partition can get 2^{2t} different configurations of two colors.

Let's denote with NCE_3 the number of conflicting edges in S_3.. When the first partition is finally colored, we have to consider two different cases:

a) The number of nodes with color c_1 is different to the number of nodes with color c_2. Without losing generality, we can suppose that there are more nodes colored with c_1 than with c_2, the other case is analogous. When coloring the other partition, and according the algorithm proposed in [3], all other nodes will be assigned color c_2 so the number of conflicting edges would be $2t - |c_2|_{(1)}$, where $|c_2|_{(1)}$ is the number of nodes with color c_2 in V_1.. The distribution of the number of nodes colored with color c_2 in V_1 is a binomial distribution with parameters $2t$ and $|c_2|_{(1)}$ ($B(2t, |c_2|_{(1)})$). Since the total number of different ways in which we can color V_1 with 2 colors is equal to 2^{2t}, then the probability of having $2kt$ conflicting edges is given by:

$$p(NCE_3 = 2kt) = 2\binom{2t}{k}2^{-2t} \qquad \text{for} \quad k = 0, 1,..., t\text{-}1;$$

where the factor of 2 is due to the fact that k can be either $|c_1|$ or $|c_2|$ regarding which one is the minimum of both.

b) The number of nodes colored with c_1 and c_2 is the same. Then the nodes of V_2 will have

colors which will be randomly chosen, but any of the 2^{2t} final configurations of V_2 will have the same number of conflicting edges, exactly $2t^2$. Then the probability of having exactly $2t^2$ conflicting edges is given by

$$p(NCE_3 = 2t^2) = \binom{2t}{t} 2^{-2t}.$$

For $k > t$ this probability is zero. Then the expected value of conflicting edges is given by

$$E(NCE_3) = \sum_{i=1}^{2t^2} iP(|CE| = i),$$

then

$$E(NCE_3) = \frac{2t}{2^{2t}} \left(2\sum_{j=1}^{t-1} j\binom{2t}{j} + t\binom{2t}{t} \right).$$

Using the equality

$$\binom{j}{1}\binom{2t}{j} = \binom{2t}{1}\binom{2t-1}{j-1},$$

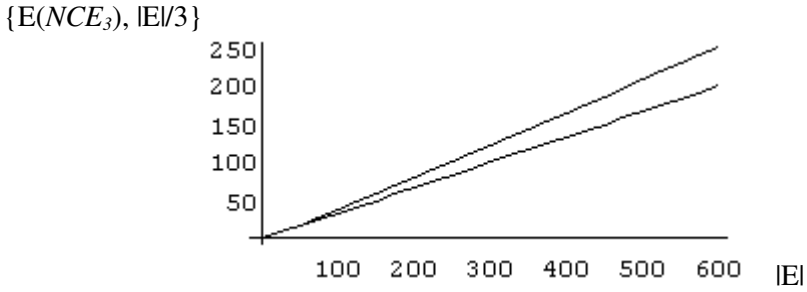
we have

$$\sum_{j=1}^{t-1} j\binom{2t}{j} = 2t\sum_{j=1}^{t-1}\binom{2t-1}{j-1} = 2t\,\frac{2^{2t-1} - 2\binom{2t-1}{t-1}}{2},$$

and finally we reach the sought result

$$E(NCE_3) = 2t^2\left(1 - \frac{1}{2^{2t}}\binom{2t}{t}\right).$$

The relation between |E| and t is given by $|E| = 4t^2$ which allows us to say that $E(NCE_3) \geq |E|/3$ for $t \geq 3$, which can be easily proved by induction on t. Figure 2 shows the difference between both functions up to |E| = 600.



For the specific example given above (a $K_{6,6}$ graph colored with two colors) the expected value of conflicting edges in S_3 is 99/8.

This result let us claim that if the same indexing order is used, then the expected value of conflicting edges is at least of the same order than the number of edges. We have proved that in this average case scenario, breaking ties uniformly at random does not help significantly over the worst-case bound.

Optimal marriages

The previous analysis has lead us to consider the introduction of another procedure before recombining two parent colorings S_1 and S_2 to create a new child coloring S_3. Since the color names are arbitrarily chosen, any given coloring with q colors is basically equivalent to $q!$ different assignments (coming from the $q!$ possible permutations of the colors' indexes). It is quite evident from the two examples introduced above that an optimal "off-line" algorithm would recognize that, in both cases, the two colorings are actually the same and would act accordingly. We remark here that the previous quotation marks have been used since the problem is definitely not an on-line problem. The characteristics of the problem faced by Costa and coworkers let us free to design any type of recombination and for that purpose we can benefit from the fact that the complete colorings S_1 and S_2 are completely known in advance. It is the "myopia" introduced by the sequential operations in the recombination we are currently studying which is the core source of the inefficiency.

We will see in the rest of this section how the previous analysis suggests the use of another procedure. It must be a fast heuristic to address the problem of reindexing the colors in one of the parents, say S_2 , in order to maximize the number of vertices which have the same color in both parents before using them to create a new coloring S_3.

Again, we will have as input two colorings (not necessarily feasible) $S_1=(V_{1,(1)},\ldots, V_{q,(1)})$ and $S_2=(V_{1,(2)},\ldots, V_{q,(2)})$ where $V_{j,(i)}$ denotes the set of vertices colored with color j in parent S_i. We will create an auxiliary graph $K_{q,q}$, a bipartite complete graph with q vertices in each partition and with weighted edges. The first partition is the set $\{v_{1,(1)},\ldots, v_{q,(1)}\}$ and the second one is $\{v_{1,(2)},\ldots, v_{q,(2)}\}$. The weight $(w_{ij})$of the edge identified by $(v_{i,(1)},v_{j,(2)})$ stands for the number of vertices in common between $V_{i,(1)}$ and $V_{j,(2)}$ (Figure 3). This said, if we find a *maximum weighted perfect matching* (in this case, it is clear that the maximum weighted matching will be perfect because the bipartite complete graph has non negative weights in the edges) of $K_{q,q}$, as defined above, and then we relabel the set of indexes of $\{V_{j,(2)}\}$ according to the best matching found, we have a way to overcome some of the problems discussed in the previous section.

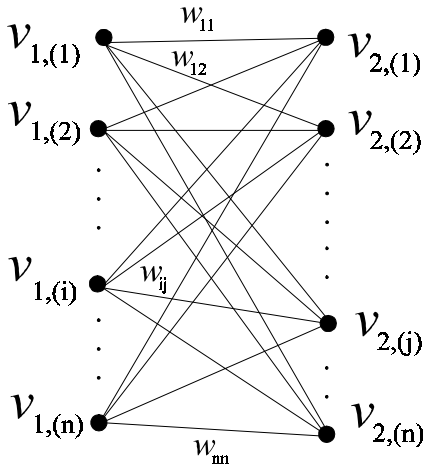


Figure 3: An auxiliary graph used for reindexing the colors before crossing two parent colorings. It is a $K_{q,q}$, a bipartite complete graph. Each vertex of a given partition represents a given color in one of the parents. The weight (w_{ij})of an edge which connects vertices representing colors i and j in partitions 1 and 2 stands for the number of vertices colored with color i in parent 1 which are colored with color j in parent 2.

For example, we will discuss again the first example which involved $K_{6,6}$. Both parents have two colors so we will need to construct an auxiliary graph $K_{2,2}$. The first partition of $K_{2,2}$ $\{v_{1,(1)}, v_{2,(1)}\}$ and the second one is $\{v_{1,(2)}, v_{2,(2)}\}$. The weights of the edges are $(v_{i,(1)},v_{i,(2)}) = 0$ where $i = \{1, 2\}$ and the others weights have a value of 6. Clearly, there are only two possible perfect matchings. Obviously, the one that has the maximum weight is the one composed by the two edges with weight equal to 6 (Figure 4). This gives a proper relabeling of the colors in S_2. We left as an exercise for the reader to work out what is the situation for the second example which involves $K_{j,\ldots,j}$.

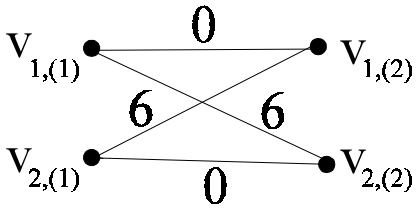


Figure 4: The auxiliary graph $K_{2,2}$ which can be created for the case posed in Figure 1.

In spite of the fact that the maximum weighted perfect matching problem has polynomial-time complexity, we think that we should constrain ourselves to lower order greedy heuristics for it. The reason is that in a memetic algorithm we expect to have a low order complexity procedure for the recombination s, having as a natural upper bound the time-complexity of the local search steps used. In addition, we are not seeking for the ultimate best algorithm to solve the maximum weighted perfect matching since this is just one possible way to deal with the problem of indexing the colors in one of the parents as we discussed in the previous sections. In a certain way, its formulation is just a model, another heuristic to solve a problem. On the other hand, there is a clear advantage associated to the use of a greedy randomized heuristic for this problem since it would deliver different solutions from the same input parents S_1, S_2. This, in turn, guarantees that we will efficiently explore neighboring

configurations between local minima when intensification steps are needed in the memetic paradigm [7] [11].

Comparative analysis

The competitive analysis of on-line algorithms has been criticized as being too crude and unrealistic. In 1994, Koutsoupias and Papadimitriou proposed two refinements of competitive analysis in two directions: The first restricts the power of the adversary by allowing only certain input distributions. The second one allows for comparisons between different information regimes for on-line decision-making. This latter refinement was used to explore the power of lookahead in server and task systems [21].

To understand their proposal, which they named *comparative analysis*, it will be good to follow their discussion, suppose that $\mathtt{A}$ and $\mathtt{B}$ are classes of algorithms where typically but not necessarily $\mathtt{A} \subseteq \mathtt{B}$, that is $\mathtt{B}$ is usually a broader class of algorithms, a more powerful information regime, then the *comparative ratio* $R(\mathtt{A},\mathtt{B})$ is defined as

$$R(\mathtt{A},\mathtt{B}) = \max_{b \in \mathtt{B}} \min_{a \in \mathtt{A}} \max_{\xi} \frac{a(\xi)}{b(\xi)},$$

where $a(\xi)$ and $b(\xi)$ are the costs of algorithms a and b on input ξ. Koutsoupias and Papadimitriou have proposed a game-theoretic interpretation of this formula: $\mathtt{B}$ wants to demonstrate to $\mathtt{A}$ that it is a more powerful class of algorithms. With this purpose, $\mathtt{B}$ choses an algorithm $b \in \mathtt{B}$ among its own. Then, responding to this choice, $\mathtt{A}$ cames up with one algorithm $a \in \mathtt{A}$. Then $\mathtt{B}$ chooses an input problem ξ. After this sequence, $\mathtt{A}$ must pay $\mathtt{B}$ the ratio $a(\xi)/b(\xi)$. It is assumed that the larger this ratio, the more powerful $\mathtt{B}$ is in comparison to $\mathtt{A}$. It should be remarked that, if we let $\mathtt{A}$ be the class of on-line algorithms and $\mathtt{B}$ be the class of all algorithms - on-line or off-line - then the equation for the comparative ratio reduces to the standard definition of competitive ratio of a problem.

We will now discuss our results on the graph coloring recombination problem we have studied having in mind the definition of competitive ratio. We have analyzed the problem of creating a new (child) coloring for a graph *G(V,E),* using two (parent) colorings S_1 and S_2. Since the recombination procedure to be analyzed was the one introduced by Costa and coworkers in Ref. [3], in fact we have studied a class of algorithms $\mathtt{A}$ which is composed of only one algorithm, named a which is this same recombination . Acting as an adversary of the algorithm a (or class $\mathtt{A}$) we have selected a class $\mathtt{B}$ composed to all possible reindexings of colors in one of the parent colorings. Note that in this case $\mathtt{A} \subseteq \mathtt{B}$ since when no reindexing is done, we apply the same basic recombination procedure *a,* then $a \in \mathtt{B}$. Using the graph-theoretic interpretation, we have selected an algorithm $b \in \mathtt{B}$ which is composed on a reindexing (based on a best-matching procedure for reindexing the colors in one of the parent colorings) followed by the application of the standard recombination procedure (algorithm a). The input problem chosen has as input to feasible colorings of a bipartite complete graph. We have used as a measure of performance of the recombination procedures (i.e. cost) the number of conflicting edges, which is a natural choice in the memetic framework. Our worst-case theorem, and the fact that the class $\mathtt{A}$ has only one element then indicates that the comparative

ratio $R(A, B) = |E|/2$ (though a slightly different definition of cost should be used to avoid the denominator to be zero).

Conclusions

In the last years, we have witnessed a growing interest on a more scientific approach to the usual methodology behind computational experimentation. J.N Hooker has clearly depicted the situation in two of his most recent papers (see [22] and [23]). C.C. McGeoch (see [24] and [25]) has also discussed issues regarding analysis of algorithms in the context of computer experimentation. Barr and coworkers have also discussed some issues for the proper design and analysis of computational experiments to analyze heuristics [26].

The aim of establishing a proper methodology for the analysis of heuristics is far from being a new interest, see for example Refs. [27] and [28]. In comparison, the theoretical analysis of metaheuristics for combinatorial optimization problems is still in its infancy. One of the intrinsic difficulties to be faced in order to develop a more scientific approach resides on the fact that the more widespread methodology, being currently used, only relies on the results of computer experiments. It is evident that this methodology has many flaws. Perhaps the most recurring one takes place when two metaheuristics, say *M1* and *M2*, are compared based on the results of computer experiments on some set of instances of a problem *P*. Not only one of the metaheuristics may be more suitable for *P* than the other, the main concern is that *M1* and *M2* are *instantiated*, i.e. we are actually comparing them after some parameterized decisions which govern the metaheuristic behavior are chosen, we can denote that we are comparing $M1(p_1, \ldots, p_n)$ against $M2(q_1, \ldots, q_m)$. For instance, if *M1* represents standard genetic algorithms (those that do not use local search, but are not restricted to binary representations), then, in our proposed notation, the parameters $\{p_1, \ldots, p_n\}$ can be numbers (population size, mutation rate, etc.) but can also denote other algorithmic procedures like the recombination operators for example. If *M2* represents a memetic algorithm, some of the parameters can be the same (population size, for instance), but others may not belong to *M1* (clearly, the type of neighborhood used for the local search technique is one outstanding example).

This said, when the two metaheuristics are instantiated, we are comparing two heuristic algorithms $M1(p_1, \ldots, p_n)$ and $M2(q_1, \ldots, q_m)$ on a particular problem *P*, which takes us back to the issues of comparison of heuristics. Unfortunately this whole picture is generally not perceived and many researchers; if $M1(p_1, \ldots, p_n)$ performs better than $M2(q_1, \ldots, q_m)$ in their particular setting they just report *M1* as a successful metaheuristic and *M2* as a failure, which is clearly a mistake due to the unproper, or at least unjustified, generalization. On the other hand, the comparison between the performance of heuristics $M1(p_1, \ldots, p_n)$ and $M2(q_1, \ldots, q_m)$ on problem *P* do not escape to the usual problems we face when comparing heuristics.

In this chapter we have shown, using a particular example of a graph coloring recombination, that a novel design methodology and theoretical analysis inspired in some concepts of *Competitive Analysis* and *Comparative Analysis* may be very useful to provide some worst-case bounds to understand the performance of at least one characteristic of metaheuristics, the recombination or recombination operators. We should also remark that it might be possible to extend this type of analysis to other type of recombinations between feasible solutions, most notably the *Path Relinking* and *Structured Weighted Combinations* procedures in Tabu Search [29]. The approach described in this chapter would be very useful when the input distribution of instances is not known or can vary across a wide range. It will also help in the cases where for the specific problem under study there are not many studied instances available in the literature. In addition, tight upper and lower bounds on the "competitiveness" of the recombination s (when properly defined) would lead to avoid cumbersome computational experiments and doubtful comparisons. However, we must warn

the reader that we are still far from developing a complete formal theory since such an analysis seems, at this moment, only linked with the "myopia" generated by the introduction of some sequential processes in the design of the recombination . But we must remark again that to act as an *"adversary"* , as it is generally the case in competitive and comparative analysis, would be beneficial to design more robust recombination operators by analyzing their comparative ratios.

We should also mention that we envision some other alternatives to formulate what constitutes a "best" matching to induce a reindexing before rec ombination. In this chapter we have selected a criterion based on finding the minimum weight perfect matching. However we would like to remark that it might be possible to use the concept of a *stable marriage* and use the *Proposal Algorithm* due to Gale and Shapley [30] for reindexing (for a comprehensive treatment of this problem and its applications see [31] and Knuth's monograph [32], see also the analysis in [33]). Instead of working with the actual value of the weights of the auxiliary graph $K_{q,q}$ we would be working with a set of preference lists indicating a partial order, a ranking among preferences. An important result to be mentioned is that it can be shown that for every choice of preference lists there exist at least one stable marriage and that the Proposal Algorithm always terminates with one of them. However, it is evident that the results presented in this chapter do not change if a reindexing based on the stable marriage formulation is used.

Regarding the usefulness of our result for the graph coloring problem itself, in [3] we can read: *"... it is difficult, if not impossible, to find a q -coloring of a large graph G (more than 300 nodes) with q close to the chromatic number of G by applying directly a given algorithm A on the graph G."* In connection with this statement, other algorithms are referenced that consists in consecutively constructing color (stable) sets of *G "which are as large as possible until we are left with at most a certain number* n_{left} *of vertices. The algorithm A is then invoked to color the remaining vertices. In this paper* ([3]) *we generalize this approach by removing consecutively partial q'-colorings (*$1 \leq q' \leq q$*) instead of independent sets (*$q' = 1$*) of G."* After their satisfactory results and the theoretical analysis presented in this chapter, we think that the challenge of finding such a *"direct"* algorithm is still wide open for future research.

Addendum – After this chapter was finally completed we received a letter from Prof. Jin-Kao Hao calling our attention to two papers he had finished on graph coloring using a memetic algorithm approach [34][35]. Most remarkable was the fact that Hao and co-workers have identified the same limitations of the *assignment approach* for recombination procedures on which our worst-case and comparative analysis is based. Computational experiments using DIMACS challenge benchmark instances showed the benefits of *reindexing* the colours before recombination, as we proposed in our theoretical analysis. These *partition-based* recombination procedures are reported to deliver highly effective memetic algorithms outperforming previous results with several metaheuristics.

It has been very exiting to find out that the theoretical worst-case analysis and the experimental results are not contradictory but also coherent. Indeed, it seems that theory can help at its best, being a prediction tool to avoid unnecessary research efforts. It is evident that there is hope for a better, more rational theory for the design of recombination operators for evolutionary computation. We consider it a *novel* approach, being explored here for the first time. But how *new* can a *good* idea be ? We have recently found the following excerpt from R. Karp´s contribution in a panel discussion which appeared in the same book that contains his seminal work [36] on NP-Completeness (see page 176):

> *'The so -called adversary approach in which we think of an algorithm as a dialogue between somebody who is executing the algorithm step by step by asking questions like: 'Is this key bigger than that key or not ?' and an adversary who tries to throw him off (a kind*

of a game theoretic approach to this worst case analysis of algorithms) is very important to keep in mind and very fruitful."

We can do little more than humbly subscribe his point of view.

Acknowledgment – We would like to thank D. Costa, J.K. Hao, R. Dorne, P. Galinier, and E. Koutsoupias for sharing with us their preprints and results before publication. P.M. wants to thank CICPBA for supporting this work at its initial stages. He wants to acknowledge current support by FAPESP, Brazil. This work is also supported by UBA, Argentina.

References

[1] P. Coll, G. Durán and P. Moscato, A discussion on some design principles for efficient recombination operators for graph coloring problems, "*Anais do XXVII Simpósio Brasileiro de Pesquisa Operacional"*, Vitória-Brazil, 1995.

[2] D. Costa, On the use of some known Methods for T-colorings of Graphs, *Annals of Operations Research* **41** (1993), 343-358.

[3] D. Costa, A. Hertz and O. Dubuis, Embedding of a sequential Procedure within an evolutionary Algorithm for coloring Problems in Graphs, *Journal of Heuristics*, **1**, 1 (1995), 105-128.

[4] L. Eshelman., The CHC Adaptive Search Algorithm: How to have safe search when engaging in nontraditional genetic recombination, in *Foundations of Genetic Algorithms* (G.J.E. Rawlins ed.). Morgan Kaufmann Publishers. San Mateo, CA, USA, (1991), 265-283.

[5] R. Hofmann, *Examinations of the Algebra of Genetic Algorithms*, Diplomarbeit, Technische Universität München (Institut für Informatik), 1993.

[6] F. Glover, *Genetic Algorithms and Scatter Search: unsuspected potentials*, Technical Report, University of Colorado, School of Business, Boulder, 1993.

[7] P. Moscato, An Introduction to Population Approaches for Optimization and Hierarchical Objective Functions: The role of Tabu Search, *Annals of Operations Research*, **41**, (1993), 85-121.

[8] P. Moscato, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*, Report 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, USA, 1989.

[9] P. Moscato and F. Tinetti, *Blending Heuristics with a Population-Based Approach: A Memetic Algorithm for the Traveling Salesman Problem*. Report 92-12, Universidad Nacional de La Plata, C.C. 75, 1900 La Plata, Argentina, unpublished manuscript, 1992.

[10] P. Moscato and M.G. Norman, A "Memetic" Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems, in *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, Amsterdam, IOS Press, **1** (1992), 177-186.

[11] N. Radcliffe and P.D. Surry, Formal Memetic Algorithms. *AISB94*, 1994.

[12] N. Radcliffe, Fitness Variance of Formae and Performance Prediction. Report TR-94-17, The University of Edinburgh, 1994. submitted to *Foundations of Genetic Algorithms.*

[13] D.D. Sleator and R.E. Tarjan, Amortized efficiency of list update and paging rules. *Communications of the ACM*, **28** (1985), 202-208.

[14] D.S. Johnson, Fast algorithms for bin packing, *Journal of Computer and System Sciences*, **8** (1974), 272-314.

[15] A. Yao, New algorithms for bin packing, *Journal of the ACM*, **27**, (1980) 207-227.

[16] A. Karlin, M. Manasse, L. Rudolph and D. Sleator, Competitive snoopy caching, *Algorithmica*, **3** (1988), pp. 79-119.

[17] M.S. Manasse, L.A. McGeoch and D.D. Sleator, Competitive algorithms for on-line problems, in *Proc. of 20th ACM Symposium on Theory of Computing*, (1988), pp. 322-333.

[18] R.M. Karp, *On-line algorithms versus off-line algorithms: how much is it worth to know the future?* , Technical Report TR-92-044, ICSI, Berkeley, 1992.

[19] E.K. Burke, J.P. Newall and R.F. Weare, A Memetic Algorithm for Univerisity Exam Timetabling, in *The Practice and Theory of Automated Timetabling*, (E.K. Burke and P. Ross eds.). Springer-Verlag, Lecture Notes in Computer Science. **1153**.

[20] B. Paechter, A. Cumming, M.G. Norman and H. Luchian, Extensions to a Memetic Timetabling System, in *The Practice and Theory of Automated Timetabling*, (E.K. Burke and P. Ross eds). Springer-Verlag, Lecture Notes in Computer Science. **1153**, 251-265.

[21] E. Koutsoupias and C.H. Papadimitriou, Beyond competitive analysis, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 394-400, Santa Fe, New Mexico, 20-22, 1994.

[22] J.N. Hooker, Needed: An empirical science of algorithms, *Operations Research*, **42** (1994), pp. 201-212.

[23] J.N Hooker, Testing heuristics: we have it all wrong, *Journal of Heuristics*, **1** (1996), 33-42.

[24] C.C. McGeogh, Towards an experimental method for algorithm simulation, *INFORMS Journal on Computing*, **8** (1996), pp. 1-15.

[25] C.C. McGeoch, Challenges in algorithm simulation, *INFORMS Journal on Computing*, **8** (1996), pp. 27-28.

[26] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, and W. Stewart, Designing and reporting on computational experiments with heuristic methods, *Journal of Heuristics*, **1** (1), (1995), pp. 9-32.

[27] D.S. Johnson and C.H. Papadimitriou, Performance guarantees for heuristics, in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* (E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys), John Wiley & Sons Ltd., Chichester, 1985, pp. 145-180.

[28] B.L. Golden and W.R. Stewart, Empirical analysis of heuristics, in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* (E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys), John Wiley & Sons Ltd., Chichester, 1985, pp. 207-249.

[29] F. Glover, Búsqueda Tabú, in "*Optimización Heurística y Redes Neuronales",* (Belarmino Adenso Diaz, ed.), Editorial Paraninfo, Madrid, 1996, pp. 105-142.

[30] D. Gale and L.S. Shapley, College admissions and the stability of marriage, *American Mathematical Monthly*, **69** (1962), pp. 6-15.

[31] D. Gusfield and R.W. Irwing, *The stable marriage problem: structure and algorithms*, MIT Press, Cambridge, 1989.

[32] D.E. Knuth, *Marriages stables,* Les Presses de l'Université de Montréal, Montréal, 1976.

[33] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, U.K., 1995.

[34] R. Dorne and J.K. Hao, A new genetic local search algorithm for graph coloring, in *Proceedings of Parallel Problem Solving from Nature (PPSN ´98), Amsterdam, 1998,* published in Lecture Notes in Computer Science 1498, pp. 745-754, (1998).

[35] P. Galinier and J.K. Hao, Hybrid Evolutionary Algorithms for Graph Coloring, *submitted to the Journal of Combinatorial Optimization.*

[36] R.M. Karp, Reducibility among combinatorial problems, in R. E. Miller and J. W. Thatcher, eds., in: *Complexity of Computer Computations*, Plenum Press, New York, 1972, pp. 85-103.

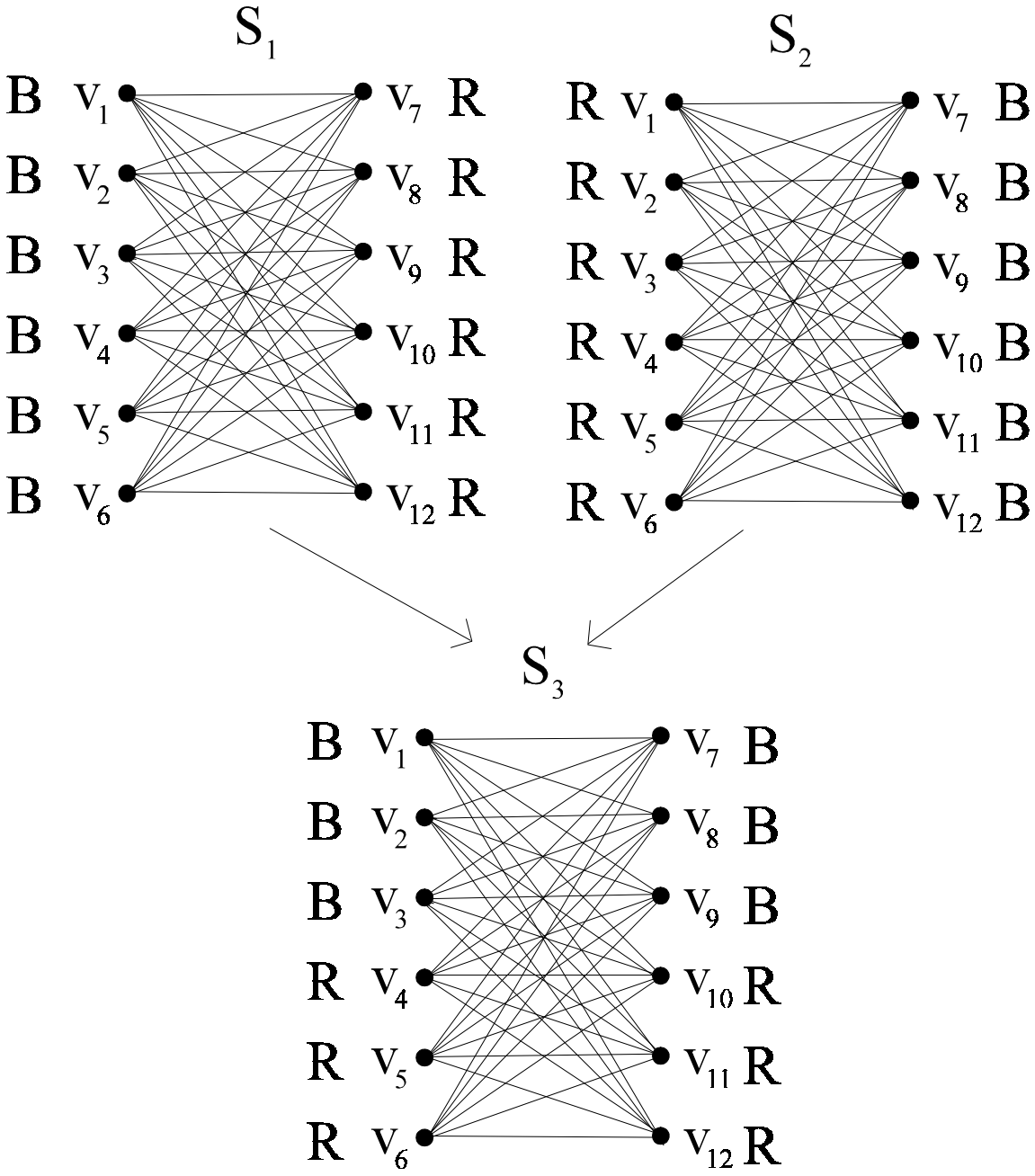


Figure 1: The figure shows two parent colorings S_1 and S_2 and one child coloring S_3 which can be the result by the application of the graph coloring recombination introduced in Ref. [3].For this particular instance of the problem, two feasible colorings (without conflicting edges) can create an output that has half of the edges in conflict.

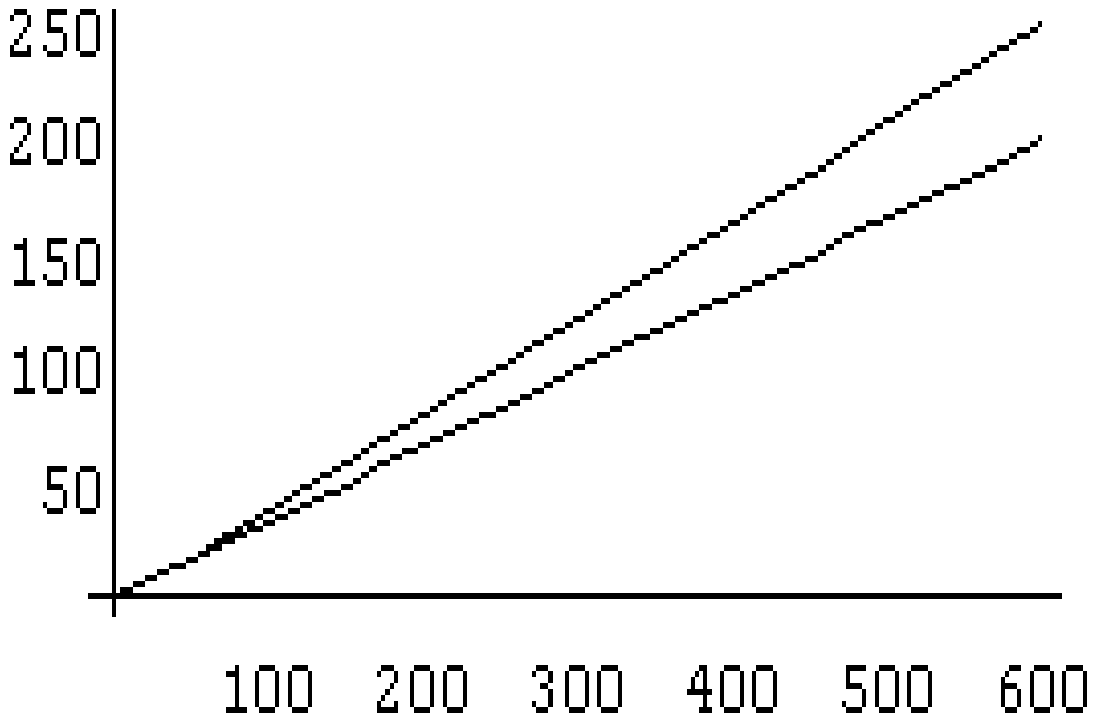


Figure 2

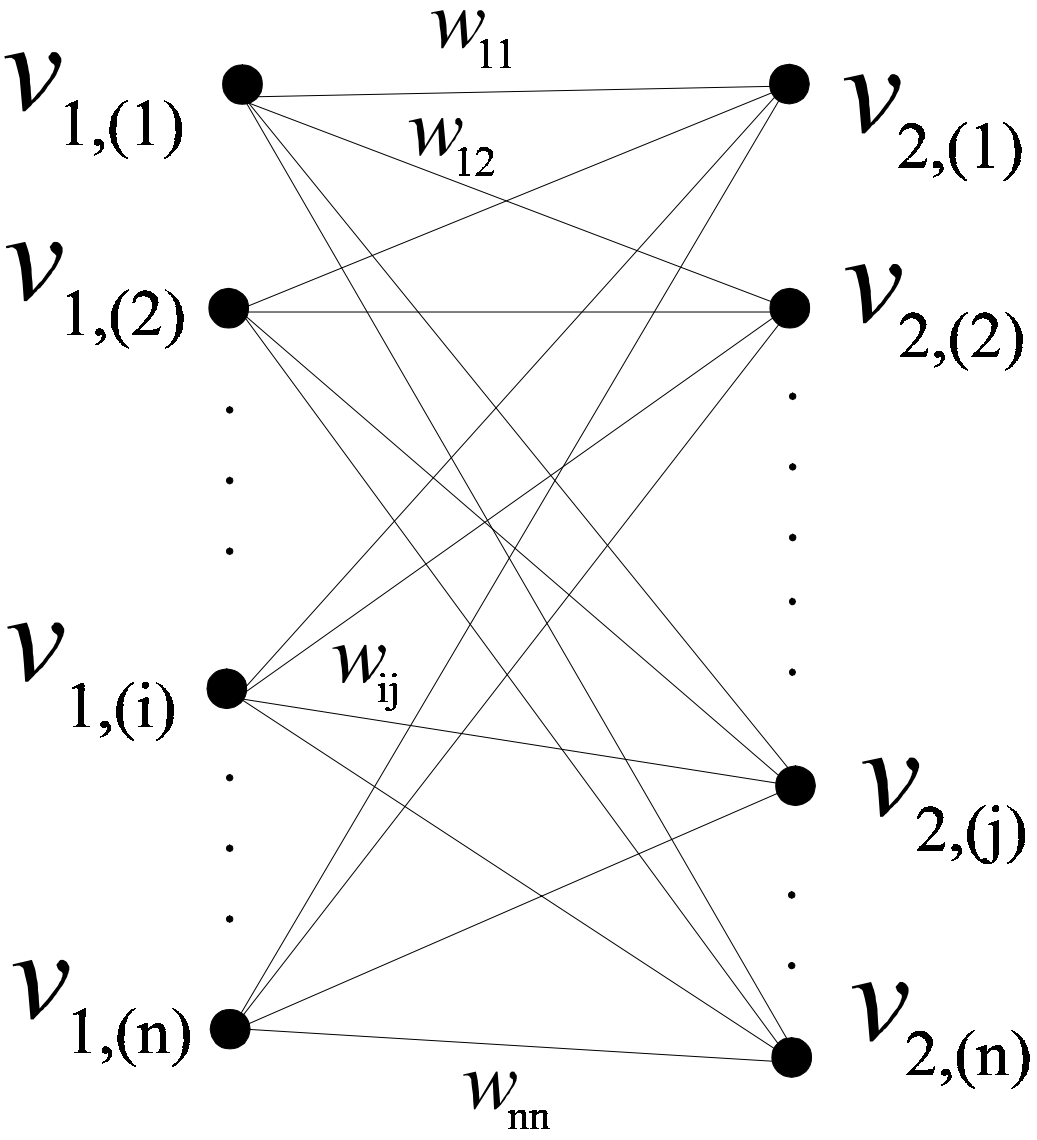


Figure 3: An auxiliary graph used for reindexing the colors before crossing two parent colorings. It is a $K_{q,q}$, a bipartite complete graph. Each vertex of a given partition represents a given color in one of the parents.

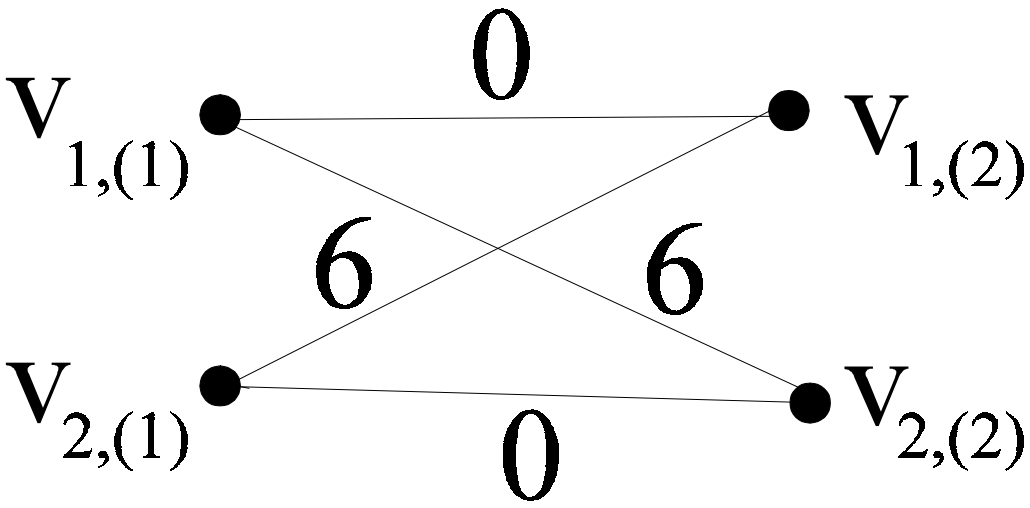


Figure 4: The auxiliary graph $K_{2,2}$ which can be created for the case posed in Figure 1.