Enhancing Solution Quality of the Biobjective Graph Coloring Problem Using Hybridization of EA

Rajeev Kumar*
Dept. Computer Sc. & Engg.
IIT Kharagpur
Kharagpur, WB 721302, India
rkumar@cse.iitkgp.ernet.in

Paresh Tolay
Dept. Computer Sc. & Engg.
IIT Kharagpur
Kharagpur, WB 721302, India
paresh.tolay@gmail.com

Siddharth Tiwary
Dept. Computer Sc. & Engg.
IIT Kharagpur
Kharagpur, WB 721302, India
siddharth.iit@gmail.com

ABSTRACT

We consider a formulation of the biobjective soft graph coloring problem so as to simultaneously minimize the number of colors used as well as the number of edges that connect vertices of the same color. We solve this problem using well-known multiobjective evolutionary algorithms (MOEA), and observe that they show good diversity and (local) convergence. Then, we consider and adapt the single objective heuristics to yield a Pareto-front and observe that the quality of solutions obtained by MOEAs is much inferior. We incorporate the problem specific knowledge into representation and reproduction operators, in an incremental way, and get good quality solutions using MOEAs too. The spin-off point we stress with this work is that, for real world applications of *unknown* nature, it is indeed difficult to realize how good/bad the quality of the solutions obtained is.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*Heuristic Methods*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*Graph and tree search strategies*

General Terms

Algorithm, Design, Experimentation.

Keywords

Optimization methods, multi-objective optimization, genetic algorithm, evolutionary algorithm, heuristics, combinatorial optimization, soft graph coloring, Pareto front.

*The author gratefully acknowledges receipt of conference travel support from Google Inc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'08, July 12-16, 2008, Atlanta, Georgia, USA
Copyright 2008 ACM 978-1-60558-130-9/08/07…$5.00.

1. INTRODUCTION

Graph coloring is a classic NP-Hard problem in graph theory, which involves assigning colors to nodes of an undirected graph with as few colors as possible, such that no adjacent nodes share a color. Graph coloring has many applications, ranging from university timetabling to register allocation used in compilers. It has applications in VLSI CAD for logic synthesis, state minimization, routing, cellular networks and aircraft/crew scheduling.

Although graph coloring has widespread applications, even the decision problem of determining whether a given graph can be colored using k colors is NP-Complete [20]. Moreover, approximating the chromatic number of a graph is NP-hard within $|V|^{1-\epsilon}$ for any $\epsilon > 0$, $|V|$ being the number of vertices, unless $NP = P$ (Feige and Kilian [16], Zuckerman [32]). There exist many exact algorithms such as specialized branch-and-bound algorithms (see for example [4, 5]), general integer programming based approaches [25], and modification of Randall-Brown's algorithm by Brélaz [4]. These are effective for small instances of graphs, or on specific classes of graphs. For general graphs, other methods are employed to obtain an approximate chromatic number. Heuristics such as Largest Degree Ordering [31], Smallest Last Ordering [24], Incidence Degree Ordering [1], Iterated Greedy [7], and DSatur [4] have been used to solve graph coloring problem, out of which DSatur is reported to give the best results (see [2, 13]). A Backtracking Correction heuristic has been used by Bhowmick et al. [2]. Approximation algorithms for graph colorings have been suggested by Wigderson [28], Blum [3], and Karger [19].

Evolutionary/genetic algorithms have been widely used for the standard graph coloring problem. Huang et al. [18], Croitoru et al. [6], Galinier and Hao [17], and Drechsler et al. [11] have used these approaches to solve the graph coloring problem. EAs/GAs try to find a coloring of a graph using k colors and try to reduce the number of constraint violations (adjacent nodes having the same color) to zero in the k coloring. For finding the minimum number of colors required to color the graph, k is iteratively decremented till no coloring for some k can be found. Jiaqi Yu and Songnian Yu [29] used parallel genetic algorithm for graph coloring problem in VLSI channel routing. Falkenauer [15] proposed a Grouping GA, which can be used for grouping problems like graph coloring and bin packing. Eiben et al. [13] suggested an adaptive EA for graph coloring, where they dynamically assign weights to vertices, and adapt these vertex weights in such a way that superior results are obtained. They named this ap-

proach the Stepwise Adaptation of Weights (SAW). Marino et al. [23] tried to improve GAs with linear programming.

The two chromosome representations that have been used widely for solving graph coloring using EA are the integer representation [12] also known as the assignment representation [11, 17], and the order based representation [6]. In the assignment approach, each gene of an individual represents a vertex of the graph, and its integral value represents its color. In the order based approach, an individual is a permutation of vertices representing the order in which they are to be colored by a coloring scheme or decoder. A comparison of the two representations has been done by Eiben et al. [12].

1.1 Biobjective Graph Coloring Problem

A biobjective variant of the standard graph coloring problem is the graph coloring with rejections. Each vertex of the graph is assigned a rejection cost. The objectives here are to color a subset of vertices of the graph with minimum number of colors, and to minimize the total rejection cost of all the vertices that have not been colored. Epstein et al. [14] discussed the offline as well as the online versions of this problem. Another biobjective variant of graph coloring is the soft graph coloring. In soft graph coloring, the adjacent vertices may be assigned the same color, but each such pair of vertices will incur a penalty. The two objectives are to minimize the total number of colors used for coloring the graph, and the total penalty in the graph.

The ANTS Challenge [1] problem by Kestrel Institute deals with applications of soft graph coloring to a resource scheduling problem. The generalization of such a scheduling problem comprises of virtual resources, where each virtual resource consists of multiple physical resources (radars in case of ANTS Challenge). Each virtual resource has to be utilized for some duration in a cyclic manner. It is desirable that all physical resources of a virtual resource be available when it is being utilized. Thus, simultaneously using two virtual resources that share a common physical resource should be avoided. On the other hand, if the time slots in the cycle are increased to minimize conflicts of physical resources, then the duration after which a physical resource can be utilized again increases, which is also undesirable. In the graph abstraction of this problem, vertices represent virtual resources, and virtual resources that share a common physical resource have an edge between their representative vertices. The colors represent time slot for utilization of resources, with resources of the same color being used together. Thus, it is necessary to minimize the number of pairs of neighbors having the same color, which can be done by increasing the number of colors. But the number of colors also have to be minimized so that the waiting time between utilization of a resource and its next utilization can be reduced. This poses an optimization problem with two conflicting objectives, namely the minimization of penalty in the graph, and the minimization of the number of colors used to color the graph. Although ANTS problem uses decentralized algorithms to solve this problem, with three radars to track a target consisting of a virtual resource, there are many instances where such a resource scheduling model can be applied in a centralized way. Such a model can be applied in university timetabling and sensor networks.

[1] `http://ants.kestrel.edu/challenge-problem/index.html` The ANTS Challenge Problem

Although a lot of work has been done on the ANTS Challenge problem, it is for a distributed real-time model, and the problem has been investigated with a fixed number of colors, thus making it a single objective problem. These factors make the solutions for this problem infeasible for the biobjective graph coloring problem. Moreover, hardly any work has been reported in literature treating soft graph coloring as a biobjective problem. The heuristics for the standard single objective graph coloring cannot be used directly for this biobjective problem. Such heuristics would yield a single optimal solution, one in each objective, and may not yield many other equivalent solutions. With the use of ϵ-constrained methods, most other solutions obtained are located near the minimal region of the respective criterion of the Pareto-front, and thus do not form the complete Pareto-front.

However, the design problem considered in this paper is essentially multiobjective in nature. A multiobjective optimizer yields a set of all representative equivalent and diverse solutions; the set of all optimal solutions is the Pareto-front. In this work, we use evolutionary multiobjective optimizer (EMO) to obtain a (near-) optimal Pareto-front. However, black-box optimizers, e.g. EMOs, in solving such hard problems, have their own challenges and difficulties.

Since the problem is *hard* and the Pareto-front is *unknown*, the main issues in such problem instances are: how to assess the convergence, and how to obtain many representative diverse solutions across the Pareto-front. Most of the strategies to assess the convergence need a *reference* solution front which is not known for this problem. Moreover, most diversity preserving strategies attempt to find a *uniformly* distributed solution front, which may not be the case with an unknown problem.

In this work, we use different multiobjective evolutionary algorithms (MOEAs) and get the solution front. Then, in order to validate the solutions, we recast some existing graph coloring heuristics into our biobjective domain and observe that the solutions obtained by MOEAs are superior to those obtained by these heuristics. So, we designed two heuristics that use the solutions obtained from the recast heuristics, and improved upon their quality. We observe that the solutions now obtained are far superior to those of MOEAs. Therefore, we embed the knowledge of heuristics in the solution evolving strategy of MOEAs and extend the solution front towards the actual Pareto-front. The solutions obtained finally by MOEAs are superior to those obtained by heuristics.

The rest of the paper is organized as follows. In Section 2, we present a brief overview of the issues and challenges in solving multiobjective real-world applications and discuss the representation scheme for graph coloring, and the genetic operators and evolutionary algorithms used. In Section 3, we present the heuristics used to validate the solution front obtained from MOEAs. We describe, in Section 4, the use of problem domain knowledge in MOEAs and design a crossover operator for the representation scheme. In Sub-Section 4.2, we change the representation scheme for our problem on the basis of heuristics, and design a problem-specific crossover operator for this representation. We include empirical results in Section 5 along with a comparison with different approaches. Finally, we draw conclusions in Section 6.

2. EA: AS A BLACK-BOX OPTIMIZER

Problem Formulation: Given a simple graph $G = (V, E)$, we need to color the vertices of the graph such that the number of pairs of adjacent vertices assigned the same color is minimized while using as few colors as possible. Assigning the same color to any two adjacent vertices incurs a penalty. Thus, we have to simultaneously minimize both the objectives, namely, the number of colors used and the total penalty incurred for the given graph.

2.1 Issues and Challenges

Essentially, performance of the multiobjective optimization (MOO) is measured for the following three characteristics:

- Extent: Coverage of the solutions across the front;
- Diversity : Sampling of the solutions across the front;
- Convergence: Distance of the obtained solution-front from the *reference* front.

In the multiobjective scenario, EAs often find effectively a set of mutually competitive solutions without needing much problem-specific information. However, achieving proper diversity in the solution-set while approaching convergence is a challenge in MOO, especially for unknown problems. Many techniques and operators have been proposed to achieve diversity. The commonly used techniques for preventing genetic drift and promoting diversity are: sharing, mating restrictions, density count (crowding), clustering and preselection operators. However, it is a common experience of many researchers that sharing can be beneficial for known problems, but can also prove surprisingly ineffective if parameters are not properly tuned. Also, it is the experience of almost all researchers that proper tuning of sharing parameters is necessary for effective performance.

A common metric for convergence is the distance metric, which finds distance of the obtained solution front from the true Pareto front; this is trivial for known problems. Such a metric is based on a reference front. In real-world search problems, location of the actual Pareto-front, by definition, is unknown. A commonly practiced approach to determine the reference front for unknown problems is to extract the reference front from the best solutions obtained so far, and the reference is incrementally updated with every generation in iterative refinement based algorithms.

For solving unknown problems there is a common concern whether or not the obtained solution set is close to the true Pareto-front. Apparently, it seems that the EA has converged to the Pareto front but conceivably it may have got *stuck* at some sub-optimal point. Such a local minima cannot be detected, for unknown problems, by most of the known metrics because a local front obtained may give excellent numerical values for extent, diversity and convergence [22].

2.2 Algorithms and Operators

From the viewpoint of EMO, the optimization problem attempted in this paper is characterized by the following features:

- No *a priori* knowledge of the solution space is available.
- There exists no information regarding a reference front.
- No experimental result from any polynomial time *good* approximation algorithm is available.

There are many MOEAs and their implementations. Seeing the hardness of the problem, we select those MOEAs, which are steady-state algorithms and use archives that can be updated with the genetic evolutions. For this, among the many such algorithms, we select NSGA-II [8], SPEA2 [8] and PCGA [21]. These algorithms do not need much problem-dependent knowledge.

Chromosome Encoding: The efficiency of the evolutionary search depends how a problem (in this case, a graph coloring) is represented in a chromosome and the reproduction operators work on the encoding. There are many encoding schemes to represent graph coloring - see [6, 12, 17] for a detailed review and comparison. In the first phase, we used the most intuitive encoding scheme among them, the assignment approach [17]. In this approach, each allele in the chromosome represents a vertex of the graph, and is assigned an integral value denoting the color of that vertex for that particular instance of coloring.

Initial Population: We generated the initial population by randomly assigning colors to the vertices in each individual graph. We observed that the genetic operators used become weak in distributing population along the end-point (the left extreme in Fig. 1, where the number of colors used is very small) of the solution-front. This is natural as we assign uniformly distributed random colors, therefore, it is very unlikely that individuals having very few distinct colors are generated. Thus, the initial population is heavily skewed towards the right side, i.e., in the region where number of colors is large.

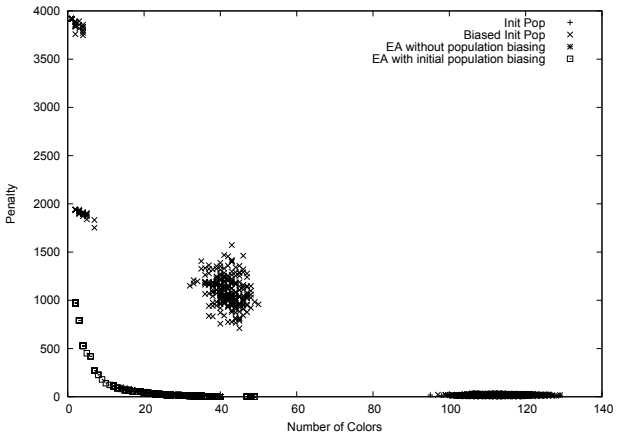

Figure 1: Solutions by an EA for a 184 node graph.

To balance this skew, we injected a few individuals heavily skewed towards the left side, i.e., where number of colors is very small. With such specific chromosomes, we used EAs again, and got solutions having better coverage across the front, as shown in Fig. 1.

Genetic Operators: In this phase, we used simple genetic operators. For crossover, we used the standard one-point crossover. The mutation operator selects random alleles from each chromosome and reassigns them random colors.

3. HEURISTIC ALGORITHMS

Now that we have obtained a solution front, we wish to assess its quality. For this we need some reference front, as no knowledge of the actual solution front is available. We use heuristic algorithms to get a reference front, which is at least as good as the solution front that we obtained from the EA.

Many heuristics have been designed for the classical single-objective graph coloring problem, where the only objective is to minimize the number of colors used to color the graph, see for example Largest Degree First [31], Smallest Degree Last [24], DSatur [4], Iterated Greedy [7] and Backtracking Correction [2]. Most of these heuristics are based on the ordering of vertices for coloring. The ordered vertices are then colored in a greedy manner, using the lowest color available to color a vertex.

Recently, Hussein et al. [1] combined two heuristics for obtaining a better ordering of vertices, by using the second heuristic for tie-breaking in determining the order. They combined Largest Degree Ordering (LDO) and Incidence Degree Ordering (IDO), using the latter for tie-breaking to obtain a new heuristic, and combined Saturation Degree Ordering (SDO or DSatur) and Largest Degree Ordering, using LDO for tie-breaking to obtain another heuristic. Their results suggest the latter combination outperforms the other heuristics used individually and in combination.

These heuristics are designed for the single-objective graph coloring problem, and thus generate a single solution. To map these heuristics to the biobjective domain, we use them to obtain a coloring for the graph, but put an upper bound on the total number of colors used. This process is repeated with the upper bound on the number of colors used iterating over integers starting from one to the value which yields a solution with zero penalty. This zero-penalty solution corresponds to the extreme case, with the number of colors used being the same as that for the solution for the single objective graph coloring with that heuristic. The greedy coloring scheme had to be accordingly modified, so that for every vertex, the color which incurs the least penalty is chosen.

Algorithm 1 Heuristics in biobjective domain

Input: G = (V, E) - a simple uncolored graph
Output: S - Set of colorings of graph G

1: Initialize $S \leftarrow \phi$, $colour\_limit \leftarrow 1$, $pty \leftarrow$ inf
2: **while** $pty \neq 0$ **do**
3: $G' \leftarrow single\_objective\_heuristic(color\_limit, G)$
4: Add the colored graph G' to S
5: $pty \leftarrow penalty(G')$
6: $color\_limit \leftarrow color\_limit + 1$
7: **end while**
8: Output S

3.1 Adapted Biobjective Heuristics

The heuristic frameworks that we used are listed below, along with their performance analysis (Fig. 3).

- Smallest Degree Last: Smallest Degree Last selects the vertex having the least degree from the graph, and sets this vertex to be colored last. It then remove this vertex from the graph, and the process is repeated on the remaining graph.
- DSatur-LDO: DSatur selects the vertex having the maximum number of differently colored neighbors, as the next vertex to be colored. A tie in saturation degree is resolved by using LDO, in which the vertex with the largest degree is selected. This heuristic outperformed DSatur, IDO, LDO and the other combination suggested by Hussein et al. [1], both in the single-objective domain as reported by Hussein, and in our biobjective domain. It also outperformed the Smallest Degree Last heuristic in our biobjective domain.
- DSatur-IDO-LDO: Following a similar approach, we introduced another heuristic in which we used IDO as the primary tie-breaking criteria, and LDO as the secondary tie-breaking criteria. Thus, in case of a tie in the saturation degree, IDO selects the vertex with the largest colored neighbors for coloring next. As in the case of DSatur, IDO also uses information of the colors of the neighbors, and was thus used before LDO. Unresolved ties are resolved by LDO. This combination outperformed all the other heuristics discussed so far.

Despite improvements over other heuristics, the solution front generated by DSatur-IDO-LDO was still inferior to the EA solution front (Fig. 3). Hence, we tried to improve the performance of DSatur-IDO-LDO. The two improvements we tried are discussed below:

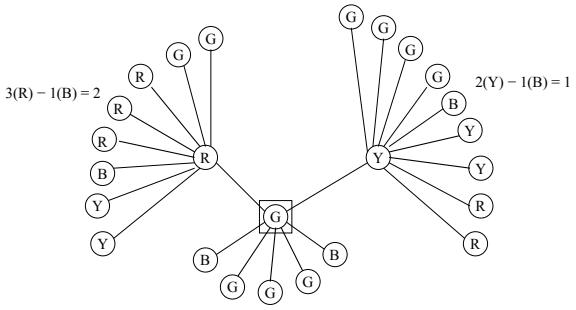

Figure 2: G vertex inside square-box is under consideration; its penalty is minimized by changing its color to R or Y. (a) Penalty adjusting heuristic 1 selects either of R or Y randomly. (b) Penalty adjusting heuristic 2 computes the ranks of R and Y colors and chooses the higher ranked R color.

Penalty Adjusting Heuristic 1

As we can see from Algorithm 1, we start by generating a coloring which uses two colors and stop when we have obtained a zero penalty coloring. These two solutions are the extreme points of the solution-front. This ensures a good extent of the solution-front. Also, as we iterate over all the colors between these extremal points, we get good diversity in the front. But the convergence of this front is inferior to the front we obtained by using EA as a black-box tool. Hence, we focus more on improving the convergence by trying to reduce the penalty once we have obtained a coloring.

In order to reduce the penalty for a given coloring, we scan over all the vertices of the graph and reassign them a color which minimizes the penalty for that vertex. In case we have more than one color which minimizes the penalty

for a vertex, we choose any of these colors randomly. This is described in Algorithm 2. It ensures that the total number of clashes, and hence penalty of the graph, does not increase and has a good chance of decreasing. Indeed, the results show that the front obtained has a better convergence than that obtained by using EA (Fig. 3).

Algorithm 2 Penalty Adjusting Heuristic 1

Input: G = (V, E) - a simple uncolored graph
Output: Set S of 2-tuples (k, p), where k is the number of colors used to color G and p is the penalty incurred for that coloring

1: Initialize $k \leftarrow 2$, $p \leftarrow \infty$, $S \leftarrow \phi$
2: **while** $p \neq 0$ **do**
3: Obtain a coloring $f_k(G)$ using the DSatur-IDO-LDO heuristic, where $f_k : V(G) \rightarrow C_k$ is a coloring of G using k colors with $|C_k| = k$.
4: **for** all vertices $u \epsilon V(G)$ **do**
5: **if** $\exists v \epsilon Neighbor(u)$, $color(u) = color(v)$ **then**
6: Find a set of colors $NC \subset C_k$ such that each color in NC causes the minimum number of conflicts with the colors of vertices in $Neighbor(u)$
7: Assign any random color $c \epsilon NC$ to the vertex u
8: **end if**
9: **end for**
10: $p \leftarrow$ total penalty in coloring $f_k(G)$
11: Add (k, p) to set S
12: $k \leftarrow k + 1$
13: **end while**
14: Output S

Penalty Adjusting Heuristic 2

Here we propose a small modification to the Penalty Adjusting Heuristic 1 proposed above. In Algorithm 2, line 7 states that we select and assign any random color from the set NC to the vertex u under consideration. Instead of selecting a random color, we apply another heuristic to select a color that is more likely to reduce the total penalty of the graph. This heuristic for color selection is described in Algorithm 3 and demonstrated in Fig. 2.

We observe that this does not significantly improve upon the convergence obtained from Penalty Adjusting Heuristic 1, but is still much better in terms of convergence than the other heuristics (Fig. 3).

4. EA: USING PROBLEM KNOWLEDGE

Now we have obtained reference fronts for our biobjectve problem using both EA and heuristics. So we aim to improve upon these reference fronts by incorporating specific problem domain knowledge in the EA. We do this in two stages. In the first stage, we re-design the evolutionary operators for the representation used in Section 2. Next, we change the representation scheme, to an order-based representation and design evolutionary operators better suited for this representation.

4.1 Problem Knowledge in Evo. Operators

Here we use the same assignment based chromosome representation as used in Section 2, where a color is assigned to each vertex. The initial population is also biased in the same way as in Section 2. We design a crossover operator that uses problem domain knowledge.

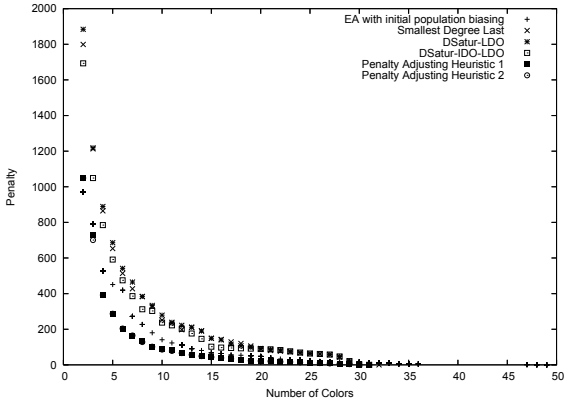

Figure 3: Heuristics compared with EA.

Algorithm 3 Penalty Adjusting Heuristic 2 :: Modifications over Heuristic 1 for selecting colors

Input: G = (V, E) - a simple uncolored graph, a vertex u, and set NC of colors
Output: Selected color for vertex u

1: **if** NC consists of only one color **then**
2: Assign that color to vertex u
3: **else**
4: **for** $\forall$ color $c \epsilon NC$ **do**
5: $color\_rank(c) \leftarrow 0$
6: **for** $\forall v \epsilon Neighbor(u)$ **do**
7: **if** $color(v) = c$ **then**
8: Construct a set F_k containing sizes of color classes for all colors in C_k, with each class containing the vertices in $Neighbor(v) - \{u\}$
9: $color\_rank(c) \leftarrow colour\_rank(c) +$ (Number of vertices $w \epsilon Neighbor(v) - \{u\}$ having color c) $-$ (size of minimum color class in F_k)
10: **end if**
11: **end for**
12: **end for**
13: Assign a color c to vertex u, such that $c \epsilon NC$ and $colour\_rank(c) \leq color\_rank(c') \forall c' \epsilon NC$
14: **end if**
15: Output c

Genetic Operators: The crossover operator that we used is an adaptation of the Greedy Partition Crossover (GPX) designed by Galinier and Hao [17]. The GPX was designed for the standard single objective graph coloring problem, and tries to achieve a k-coloring of the graph, when the number of colors k is fixed a priori. In GPX, the two parents are considered alternately, and vertices in the color class with the maximal number of unassigned vertices in the considered parent are assigned an unused common color in the child, and deleted from both the parents. Galinier and Hao try to obtain the minimum value of k for which a k-coloring is obtained.

In our biobjective problem, we extract only those vertices from the largest color partition in the parent under consideration, which incur minimum penalty among all vertices in that partition, and assign these vertices an unused com-

mon color in the child. When any other vertices from this color partition of the parent are selected in a later iteration, they are assigned the same color as that assigned earlier to other vertices from this partition. This Penalty based Color Partitioning crossover (PCPX) algorithm is described in Algorithm 4. We used the polynomial mutation operator suggested by Deb [8]. Real values of colors obtained after mutation were rounded off to the nearest integer.

We observe that the solution front obtained using Penalty based Color Partitioning crossover has slightly better convergence than EA used as a black-box tool for large values of number of colors. That is, when more colors are available, it gives a lower penalty coloring than by EA. For small number of colors, EA gives lower penalty coloring. Nevertheless, the best results from heuristics (obtained by using Penalty adjustment) have better convergence than both of them.

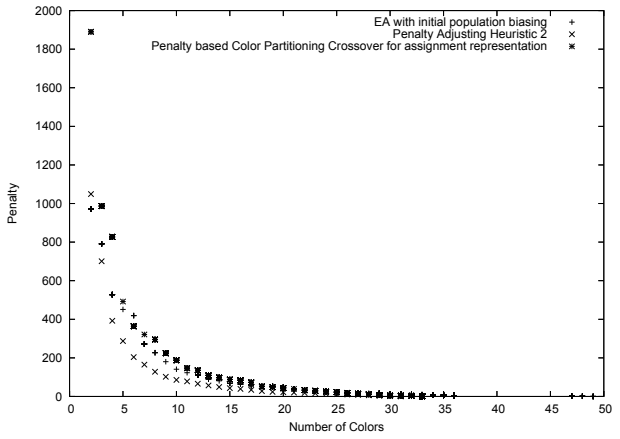

Figure 4: The Penalty based Color Partitioning Crossover (PCPX) compared with EA and the best heuristic results on the same 184 node graph.

Algorithm 4 Penalty based Color Partitioning crossover (PCPX)

Input: Parents $P1$ and $P2$
Output: T - Offspring

```
Initialize selected parent P = P1
while any vertex in T still not assigned a color do
    Obtain largest color partition C from P
    Extract set of vertices minP from C which incur minimum penalty among vertices in C
    if color of C not mapped to a color in T then
        Assign a new color mapping in T for this color of C
    end if
    Assign mapped color of C to vertices of minP in T
    Remove all vertices of minP from both parents P1 and P2
    Swap selected parent P from P1 to P2 or from P2 to P1
end while
Output T
```

4.2 Order Based Representation Scheme

Most of the heuristics for graph coloring are based on the ordering of vertices. We incorporate this in our chromosome encoding scheme to make it an order based representation. This representation scheme has been widely used for solving graph coloring problems using EA (see [6]) and has been generally found to give good results in the single objective domain. As an ordering is essentially a permutation of vertices, we designed some crossover operators for permutations while also incorporating problem domain knowledge.

Chromosome Encoding: The order-based encoding was used. The value assigned to the i^{th} allele in the chromosome indicates the vertex that will be the i^{th} vertex to be colored. There is another allele that stores an integer representing the maximum number of colors k that can be used to color the graph. This allele is necessary to decide the upper bound on the number of colors to be used to color the graph.

Initial Population: The initial population consists of individuals having random permutations of the vertices.

Coloring Scheme: We need to color the vertices in the order provided by the chromosome in order to compute the total penalty incurred for that ordering of vertices. We use the greedy coloring scheme for coloring. Each vertex is assigned the color which incurs the least penalty in the current coloring of the graph, and the number of colors used does not exceed the maximum number of colors k allowed for that individual.

Genetic Operators: We first used the order-based crossover described in [6] and named as Permutation One Point Crossover (POP) [26]. Next, we incorporate problem knowledge in the crossover by using ideas from some of the heuristics we used. The Largest Degree Ordering (LDO) heuristic discussed in Section 3 suggests that the vertices having large degree should be colored first. Our Degree Based Crossover operator gives a preference to vertices having large degree in the ordering sequence. This Degree Based Crossover (DBX) operator is given in Algorithm 5. We used the swap mutation [27] on the ordering of vertices.

Algorithm 5 Degree Based Crossover (DBX)

Input: Parents $P1$ and $P2$
Output: T - Offspring

```
Choose a random mating point m
Assign vertices having order 1 to m in P1 the same order in T, that is, copy vertices from 1 to m from P1 to T
for all vertices u in P2 starting from order 1 in P2 do
    if u not assigned an order in T then
        if there exists some vertex v in the ordering in T having degree lesser than degree of u then
            Find the vertex v having minimum order in T and with degree less than degree of u
            Increment by one the order of all vertices having current order greater than or equal to that of v
            Assign the original order of v to u
        else
            Assign the lowest available order to u
        end if
    end if
end for
Output T
```

The allele representing the number of colors k that can be used for coloring does not take part in the Degree Based Crossover. As this is a single element, we cannot use simple

one-point crossover operator. For this, we used the Simulated Binary Crossover (SBX) [9] and polynomial mutation proposed by Deb et al. [8].

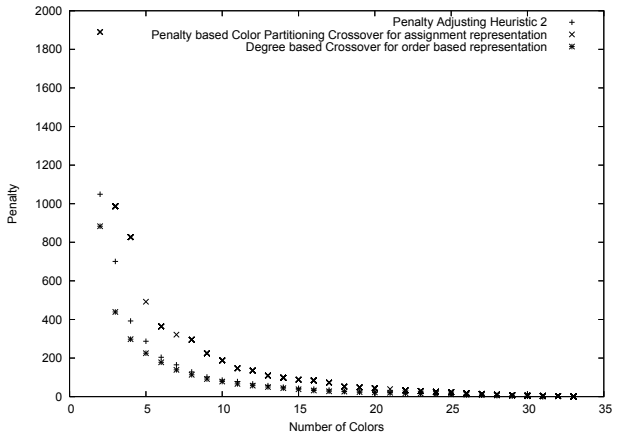

Figure 5: Degree based crossover for order based representation compared with best heuristic results and assignment based representation

The order-based representation scheme vastly improved the solution quality. Incorporating problem knowledge in the crossover further improved the results, which are shown in Fig. 5 along with the results of the best heuristic and the assignment based EA.

5. RESULTS AND DISCUSSIONS

We tested our heuristics and evolutionary algorithms for the biobjective graph coloring problem on the benchmark data taken from the DIMACS Graph Coloring Challenge[2]. The DIMACS data is a collection of test data sets obtained from various sources for a variety of problems requiring graph coloring. We considered graphs of varying densities having 120, 128, 184, 450, 496 and 864 nodes. Here, we discuss the results obtained for coloring the mulsol.i.3.col graph having 184 nodes and 3916 edges. The algorithms that we used give a similar performance for the other graphs as well . We observe the results after 3000 generations, starting with an initial population of 1000 individuals. We keep the crossover probability as 0.8 and mutation probability as 0.01.

We quantitatively evaluate the solution fronts obtained from the algorithms we have used. We compute C-measure [30], convergence [10], spread [8] and hypervolume (S measure) [30] for the basic MOEA, best heuristic result, and the two hybridized MOEAs. The *convergence* metric measures the convergence of the obtained solution set against a reference set. In the case of unknown problems, approximation sets of all the considered algorithms are combined and non-dominated approximation set is computed to act as a reference set. A lower value of the metric indicates better convergence. Ideally, this should be zero. The *spread* metric also uses a reference set to measure the average distribution of points in obtained solution set and the distance between the extreme solutions in obtained solution set in comparison to extreme solutions in reference set. Here also, a lower value of the metric indicates better spread. The *hypervolume* metric provides the volume of search space covered by the obtained set of solutions with reference to a reference point. Hence, a higher value indicates superiority of solution (better solution). The computed values shown in the Table 1 are for a typical case. A total of 10 runs were made on each test graph for the EAs, and the results shown are for one representative run. It was observed that for each of the benchmark test graphs, the minimum number of colors required for exact coloring was equal to the best known coloring for the graph, when using the order based representation with DBX.

[2]`http://mat.gsia.cmu.edu/COLOR/instances.html` has the graph coloring instances

Table 1: C-measure, Convergence, Spread and Hypervolume (S-measure) metric values for a graph of 184 nodes and 3916 edges.

Algorithm	Basic EA	Best Heur	PCPX	DBX
C-measure	1.0000	0.5667	1.0000	0.0000
Convergence	0.0902	0.0218	0.0596	0.0000
Spread	1.8111	1.2903	1.5987	1.3329
Hypervolume	90938	92336	89324	93059

As can be seen from Table 1, the order-based representation used with a Degree based Crossover gives the best convergence, for both the C-measure and convergence metrics. The heuristic C-measure and convergence are better than both the basic EA and Penalty based Color Partitioning Crossover for the assignment based representation, which were both completely dominated by the order based representation. The heuristic gives the best spread, as we iterate over all values of the number of colors k used for coloring. Thus, the solutions are uniformly spaced between the heuristics. The EA with order based representation using DBX crossover follows the heuristic in the spread metric. The solutions from all EA based algorithms tend to give slightly poorer spread than most heuristics as they generate solutions stochastically, and some solutions along the front may not be generated. For example, in our case, there may arise gaps in the solution front for some runs of the algorithm, where the soft coloring of the graph for some value k of colors is not generated in the solution set. In the hypervolume measure, the EA with order-based representation with DBX crossover again performs the best, with the heuristic following closely behind.

6. CONCLUSIONS

We observe that MOEAs perform poorly when used without any problem knowledge. Incorporating problem knowledge in the crossover (PCPX) slightly improved the performance. But the heuristics outperform both of them in all the metrics that we used. This indicates that the problem knowledge must be introduced in some other aspects of the EA strategy. Changing the chromosome representation scheme to an order based representation drastically improved the performance. Further improvement was achieved by modeling the crossover (DBX) on heuristics. Moreover, the poor performance of the assignment based representation may be due to the symmetry in the solution space, where coloring may be different but the color classes may still be the same, leading to the same objective values. This symmetry tends to increase the size of the search space exponentially.

Hybridizing the EA by changing the representation to an order based representation made use of a very common technique used by graph coloring heuristic algorithms. Also, designing a problem specific crossover operator based on some of the heuristics, gave superior results than both the heuristics and MOEAs.

7. REFERENCES

[1] H. Al-Omari and K. E. Sabri. New graph coloring algorithms. *Journal of Mathematics and Statistics*, pages 739–741, 2006.

[2] S. Bhowmick and P. Hovland. A backtracking correction heuristic for improving performance of graph coloring algorithms. *2nd Int. Workshop on Combinatorial Scientific Computing*, 2005.

[3] A. Blum. New approximation algorithms for graph coloring. *J. ACM*, 41(3):470–516, 1994.

[4] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, 1979.

[5] M. Caramia and P. Dell'Olmo. Bounding vertex coloring by truncated multistage branch and bound. *Netw.*, 44(4):231–242, 2004.

[6] C. Croitoru, H. Luchian, O. Gheorghies, and A. Apetrei. A new genetic graph coloring heuristic. *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, 63-74, 2002.

[7] J. Culberson. Iterated greedy graph coloring and the difficulty landscape. Technical Report TR 92-07, 1992.

[8] K. Deb. *Multiobjective Optimization Using Evolutionary Algorithms*. Chichester, UK: Wiley, 2001.

[9] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[10] K. Deb and S. Jain. Running performance metrics for evolutionary multiobjective optimization. *In SEAL, Singapore*, pages 13–20, November 2002.

[11] N. Drechsler, W. Günther, and R. Drechsler. Efficient graph coloring by evolutionary algorithms. In *Proceedings of the 6th International Conference on Computational Intelligence, Theory and Applications*, pages 30–39, London, UK, 1999. Springer-Verlag.

[12] A. Eiben and J. van der Hauw. Graph coloring with adaptive genetic algorithms. *Technical Report TR 96-11, Leiden University*, August 1996.

[13] A. E. Eiben, J. K. V. D. Hauw, and J. I. V. Hemert. Graph coloring with adaptive evolutionary algorithms. *Journal of Heuristics*, 4(1):25–46, 1998.

[14] L. Epstein, A. Levin, and G. J. Woeginger. Graph coloring with rejection. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 364–375, London, UK, 2006. Springer-Verlag.

[15] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics, 2:5–30*, 1996.

[16] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. Syst. Sci.*, 57(2):187–199, 1998.

[17] P. Galinier and J.-K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[18] F. Huang and G. Chen. A symmetry-breaking approach of the graph coloring problem with gas. *Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on*, 2:717–719 Vol.2, 26-28 May 2004.

[19] D. R. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[20] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[21] R. Kumar and P. Rockett. Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution: A Pareto Converging Genetic Algorithm. *Evolutionary Computation*, 10(3):283–314, Fall 2002.

[22] R. Kumar, P. K. Singh, A. P. Singhal, and A. Bhartia. Evolutionary and heuristic algorithms for 0-1 knapsack problem. In *10th Online World Conf. Soft Computing & Indutrial Applications, Applications of Soft Computing: Recent Trends*, 2005.

[23] A. Marino, A. Prugel-Bennett, and C. Glass. Improving graph colouring with linear programming and genetic algorithms. *In K. Miettinen, M. M. Makela, and J. Toivanen (Eds.), Proceedings of EUROGEN99, Jyvaskyla, Finland*, 113–118, 1999.

[24] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.

[25] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8:344–354, 1996.

[26] C. Mumford. New order-based crossovers for the graph coloring problem. In *Parallel Problem Solving from Nature - PPSN IX*, pages 880–889, Reykjavik, Iceland, September 2006. LNCS 4193, Springer.

[27] E. Ryan, R. M. A. Azad, and C. Ryan. On the performance of genetic operators and the random key representation. In *Genetic Programming, Lecture Notes in Computer Science*, pages 162–173, Berlin/Heidelberg, 2004. Springer.

[28] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *J. ACM*, 30(4):729–735, 1983.

[29] J. Yu and S. Yu. A novel parallel genetic algorithm for the graph coloring problem in vlsi channel routing. *Natural Computation, 2007. ICNC 2007. Third International Conference on*, 4, 101-105, 2007.

[30] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Parallel Problem Solving from Nature - PPSN V*, pages 292–301, Berlin/Heidelberg, 292–301, 1998. Springer.

[31] J. Zoellner and C. Beall. A breakthrough in spectrum conserving frequency assignment technology. *IEEE Trans Electromag Compat*, EMC-19:313–319, 1977.

[32] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007.