A Genetic Algorithm to Minimize Chromatic Entropy

Greg Durrett[1], Muriel Médard[2], and Una-May O'Reilly[1]

[1] Computer Science and Artificial Intelligence Laboratory
[2] Research Laboratory for Electronics
Massachusetts Institute of Technology
{gdurrett,medard,unamay}@mit.edu

Abstract. We present an algorithmic approach to solving the problem of chromatic entropy, a combinatorial optimization problem related to graph coloring. This problem is a component in algorithms for optimizing data compression when computing a function of two correlated sources at a receiver. Our genetic algorithm for minimizing chromatic entropy uses an order-based genome inspired by graph coloring genetic algorithms, as well as some problem-specific heuristics. It performs consistently well on synthetic instances, and for an expositional set of functional compression problems, the GA routinely finds a compression scheme that is 20-30% more efficient than that given by a reference compression algorithm.

Key words: chromatic entropy, functional compression, graph coloring

1 Introduction

Chromatic entropy is a combinatorial optimization problem closely related to graph coloring, though it is much less well known because of its relative lack of applications. However, it has recently appeared in information theory as the bottleneck in a particular method of encoding data from two correlated sources. Specifically, a solution to chromatic entropy would allow the implementation of an improved scheme for data compression of correlated sources used as inputs to a function [1].

Chromatic entropy is known to be NP-hard, as shown by Cardinal et al. [2]. Given that this problem cannot be solved efficiently in theory, it is natural to consider finding solutions using genetic algorithms. In this paper, our goal is to introduce the problem and describe a genetic algorithm (GA) tailored to its specifics. In Section 2 we present the background on functional compression necessary to understand the application, then discuss the problem of chromatic entropy. Because our GA borrows heavily from previous algorithms for solving graph coloring, we proceed to briefly survey some of the relevant GA graph coloring literature. In Section 3, we describe the design choices for our GA. Section 4 analyzes the performance of the algorithm on synthetic probabilistic graphs and on graphs derived from certain instances of the functional compression problem, showing that the algorithm performs favorably on both. Section 5 concludes and discusses future work.

2 Background

2.1 Definitions

We use the standard definition of entropy throughout this paper.

Definition 1. *Let X be a discrete random variable with probability density function p that can take values x_i for $1 \leq i \leq n$. The entropy of X is given by*

$$H(X) = -\sum_{i=1}^{n}(p(x_i)\log_2 p(x_i)) \tag{1}$$

Conditional entropy will also be used frequently and is not quite a trivial extension of basic entropy.

Definition 2. *Let X and Y be discrete random variables, with a joint distribution $p(x,y)$, conditional distributions $p(x|y)$ and $p(y|x)$, and marginal distributions $p(x)$ and $p(y)$. If X can take values x_i for $1 \leq i \leq n$ and Y can take values y_j for $1 \leq j \leq m$, we define the conditional entropy $H(X|Y)$ as follows:*

$$H(X|Y) = \sum_{j=1}^{m}(p(y_j)H(X|Y=y_j)) = \sum_{j=1}^{m}\left(p(y_j)\left(-\sum_{i=1}^{n}(p(x_i|y_j)\log_2 p(x_i|y_j))\right)\right) \tag{2}$$

This can be thought of as an average of the conditional entropies $H(X|Y=y_j)$ weighted according to the probability that $Y = y_j$.

In addition to entropy, we require the notion of a coloring in a discrete, edge-node graph.

Definition 3. *Let $G = (V, E)$ be an undirected graph over the vertex set V with edge set E. A* k-coloring *of G is an assignment of "colors" to vertices of G using k colors such that no two adjacent vertices have the same color. More formally, we partition the vertices into k disjoint sets C_i such that if $x_1, x_2 \in C_i$, then $(x_1, x_2) \notin E$.*

Note that throughout this work, the graphs we are using are simple, undirected graphs. Henceforth, any graph not otherwise specified is simple and undirected.

2.2 Functional Compression Basics

Doshi et al. provide in [1] the main information-theoretic results relevant to the efforts of this paper. We provide a brief summary here of the key concepts presented in their paper.

Consider two correlated sources of data, X and Y, separated from each other and X separated from a decoder. Figure 1 shows the basic setup: we wish to encode the data from X in order to compute a function $f(X,Y)$ without loss at the decoder, and we want this encoding to be as efficient as possible. In the case

where $f(X,Y) = (X,Y)$ (i.e. when we want to be able to recover both X and Y directly), the well-established rate result is the Slepian-Wolf bound described in [3]. The theorem of [3] treats the more general problem when both X and Y are separate from the decoder, and states that X and Y can be sent at rates R_1 and R_2 satisfying $R_1 > H(X|Y)$, $R_2 > H(Y|X)$, and $R_1 + R_2 > H(X,Y)$. In our specific problem, when Y is available at the decoder, the optimal rate for the transmission of X is the conditional entropy of X given Y, $H(X|Y)$. Achieving this rate entails sending X using a more efficient encoding by taking advantage of its correlation with Y, knowing that the decoder can use its knowledge of Y to disambiguate the transmitted data. In the literature, this problem is known as the problem of functional compression with side information.

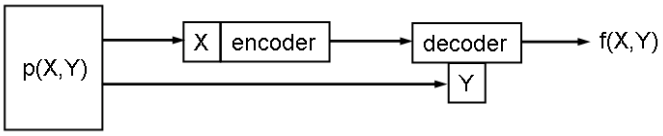


Fig. 1. The functional compression problem when Y is available as side information at the decoder.

This analysis so far has ignored the fact that we are computing $f(X,Y)$ at the decoder, rather than reporting the two values directly. If f is a many-to-one function, we can do significantly better than the Slepian-Wolf bound. As an example taken from [1], consider when X and Y are integers and f is the function $X + Y \pmod 4$. Regardless of how large X and Y might be, it is sufficient to encode only the last two bits of each, rather than the entire integer. The Slepian-Wolf bound only optimizes the rate by taking advantage of the correlation between the two random variables; it does not take advantage of the properties of the function to further increase the efficiency of the encoding.

In general, it is not this simple to use our knowledge of f to improve our encoding. To analyze a given f, we construct something called the confusability graph.

Definition 4. *The confusability graph for X given Y, f, and $p(X,Y)$ is a graph $G = (V,E)$ where V contains a vertex v_x for each value x that X can take and E contains an edge between v_{x_1} and v_{x_2} if x_1 and x_2 are confusable. Two values of X are* confusable *if for some $y \in Y$, $p(x_1,y) > 0$, $p(x_2,y) > 0$, and $f(x_1,y) \neq f(x_2,y)$.*

This definition is slightly easier to grasp if one considers the complement of the graph. If two nodes are not confusable (i.e. there is no edge between them in the confusability graph), then for all possible values y of the random variable Y, either $p(x_1,y) = 0$ or $p(x_2,y) = 0$ (we can disambiguate the two values based on the fact that one of them can never appear with the given y), or $f(x_1,y) = f(x_2,y)$ (they yield the same value under the function, so there is no

need to disambiguate them). See Section 4.1 for an example of using these rules to construct a confusability graph.

By this definition, two nodes that are disconnected on the confusability graph can safely be given the same encoding, since knowing Y and knowing that X takes one of these two values is all that we require to compute the value of f. This extends beyond pairs of nodes to independent sets of nodes. By coloring the confusability graph, we partition it into color classes, each of which is an independent set. To transmit a particular value, we can simply transmit the name of the color class that contains that value, without ambiguity as to what the value of $f(X, Y)$ will be. Using a Slepian-Wolf encoding on the distribution over the color classes yields a valid encoding scheme for the problem. This rate is precisely the conditional chromatic entropy of the confusability graph of X given Y, which we define mathematically in the next section. However, intuitively, this is appropriately analogous to the case when $f(X, Y) = (X, Y)$, for which the rate bound is the conditional entropy $H(X|Y)$. By coloring we have augmented the Slepian-Wolf method to compress data based not only on the correlation between X and Y, but on the properties of the function f, as captured by the confusability graph.

2.3 Chromatic Entropy

Armed with this intuition about the method, we can rigorously define chromatic entropy. Chromatic entropy is defined for simple graphs that have an associated probability distribution p over the set of nodes (i.e. $\sum_{v_i \in V} p(v_i) = 1$). We use the following definition from [2]:

Definition 5. *Let $G = (V, E)$ be a graph with a probability distribution p of a random variable X over the vertices and k color classes C_i. Set $p_{C_i} = \sum_{v \in C_i} p(v)$; this represents the probability that a vertex of G chosen according to p falls in the ith color class. The chromatic entropy of G given p and C is*

$$H_G^\chi(C, X) = -\sum_{i=1}^{k} p_{C_i} \log_2(p_{C_i}) \tag{3}$$

Essentially, this is the entropy associated with the probability distribution ϕ over the color classes C_i, where $\phi(C_i) = \sum_{v \in C_i} p(v)$.

Körner showed in [4] that by minimizing this quantity for a high enough "power" of a confusability graph, we can achieve an arbitrarily close approximation to the optimal rate for the case when we are transmitting a single value X to the decoder and the function $f(X)$ depends only on X, which is a simplified version of our compression problem with Y removed. Doshi et al. [1] extended this result to the side information case, where we have a conditional distribution $p(X|Y)$ rather than a univariate distribution. They define a quantity called conditional chromatic entropy, which by analogy with the definition of conditional entropy in Section 2.1, is given to be

$$H_G^\chi(C, X|Y) = \sum_{y \in Y} p(y) H_G^\chi(C, X|Y = y) \tag{4}$$

This conditional chromatic entropy is the optimal rate for encoding of X. It represents a rate optimized by taking advantage of both the correlation between the signals X and Y and the properties of the function.

For a discussion of the relationship between chromatic entropy and standard entropy, we refer the reader to Alon and Orlitsky's work in [5].

Heuristics Given that the sum can be decomposed this way, it is helpful to consider heuristics for minimizing chromatic entropy motivated by the univariate chromatic entropy problem, as this is much easier to analyze than the conditional chromatic entropy problem.

Intuition suggests that reducing the number of colors also reduces chromatic entropy, but while this is a good heuristic, it is not necessary true that the coloring yielding the minimum chromatic entropy is a minimum coloring. This is proven in full depth by Cardinal et al. in [2], but we present here a simple example inspired by their proof. Consider the graph shown on the left in Figure 2. The graph is 3-colorable, and is clearly not 2-colorable by the existence of a 3-clique. Up to symmetries of vertices and color classes, there are only two three-colorings: $\{1, 6\}$, $\{2, 5\}$, $\{3, 4\}$ (shown in Figure 2, middle) and $\{1, 5, 6\}$, $\{2, 4\}$, $\{3\}$ (not shown). The first 3-coloring contains three color classes all with equal probability mass, and $H_G^\chi = 1.5849$. For the second 3-coloring, we can compute the chromatic entropy as follows:

$$H_G^\chi = -0.616 \log_2 0.616 - 0.333 \log_2 0.333 - 0.05 \log_2 0.05 = 1.1745 \qquad (5)$$

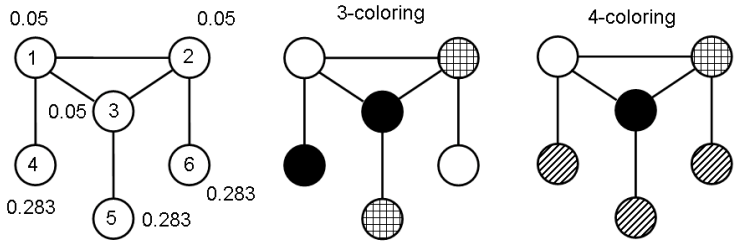


Fig. 2. Example graph to demonstrate that minimum colorings do not always yield minimum chromatic entropy values. One of the possible three-colorings and the optimal four-coloring are also shown.

However, the four-coloring with color classes $\{1\}$, $\{2\}$, $\{3\}$, and $\{4, 5, 6\}$ (Figure 2, right) has $H_G^\chi = 0.8476$. This example demonstrates that, as with the general entropy function, chromatic entropy is minimized when the terms in the entropy sum are as "uneven" as possible. The fourth color class provided "flexibility" to make the distribution ϕ over the color classes more uneven ($[0.85, 0.05, 0.05, 0.05]$ as opposed to $[0.6166, 0.3333, 0.05]$) thereby reducing the entropy.

One heuristic inspired by this example is to attempt to find independent sets with high probability mass and establish these as color classes, the idea being to unbalance the distribution as much as possible. There is another sense in which concentrating probability mass is beneficial: Lemma 2 of [2] tells us that, given a choice of color classes to assign a node to, chromatic entropy is increased the least by adding the node to the color class with the highest probability. This follows from the concavity of $f(x) = -x\log(x)$ in the entropy function. These two principles should inform our GA design in order to take full advantage of the problem structure.

Although we now know generally how to best distribute mass within color classes, we still do not know how many color classes to use. Theorem 6 in [2] states that the number of colors in a coloring that minimizes chromatic entropy for a particular graph cannot be bounded by any function of the number of colors in a minimum coloring for that graph, even for very specific, well-behaved types of graphs. Fortunately, there is a sense in which minimizing the number of color classes minimizes the chromatic entropy, as captured by the following lemma.

Lemma 1. *Given a graph G with distribution p and coloring C, if color classes C_1 and C_2 can be merged without violating the rules of coloring, then the coloring $C' = \{C_1 \cup C_2, C_3, ..., C_k\}$ has strictly lower chromatic entropy than the original coloring C.*

The proof follows directly from the definition of chromatic entropy, so we omit it. Because there are only constrained situations under which increasing the number of color classes decreases the chromatic entropy, minimizing the number of color classes is a powerful heuristic for minimizing chromatic entropy.

2.4 Genetic Algorithms for Graph Coloring

The connections between graph coloring and chromatic entropy indicate that a GA for one might be effectively adapted to the other, if the fitness function were to be changed appropriately. We can draw inspiration from the genomes used to solve graph coloring problems. Each individual in the population encodes a coloring, and in this sense chromatic entropy poses an interesting challenge because, for some representations, only a small fraction of the potential individuals encode legal colorings. The most naive representation might be to have a genome of length n for an n-node graph, and have each position store the color for the corresponding node. However, a mutation or crossover operation applied to an individual that represents a legal coloring will frequently produce an individual that represents an illegal coloring. In [6], Dorne and Hao successfully use this genome to solve graph coloring by taking their fitness function to be the number of edges that violate the coloring constraints. Their approach, though interesting, only works for a fixed number of colors, so it is unsuitable for chromatic entropy.

Eiben et al. [7] use a GA to minimize the number of violated coloring constraints, in order to find a valid k-coloring for a fixed k. They use a so-called

"order-based" representation. Each individual in the population is a permutation of the vertices of the graph in question. An individual is "decoded" into a coloring using a greedy algorithm as follows: the nodes are iteratively colored in the order specified by the permutation, and at each iteration, the current node is colored using the lowest-numbered color possible given the partial coloring of the graph. This genome admits a number of sensible mutation and crossover operators and every potential individual corresponds to a valid coloring. The reverse is not true: certain non-minimal colorings cannot be created by this greedy method. However, via a proof we omit for brevity, the optimal solution is assured to be attainable.

Sivanandam et al. [8] also use an order-based representation but their GA uses a different greedy algorithm for decoding. Their decoding introduces many more color classes than the decoding from [7]. For our problem this correlates with chromatic entropy values that are relatively high. We have elected to use the PX crossover of [8].

3 Algorithm Design

3.1 Genome and Objective

We use the order-based representation from [7] and [8] for our genome, as this sidesteps the issue of what to do when an invalid coloring is produced. For decoding, we use the greedy color assignment algorithm of [7] per Section 2.4.

Our objective is to minimize conditional chromatic entropy as defined in Section 2.3, given by

$$H_G^\chi(C, X|Y) = \sum_{y \in Y} p(y) H_G^\chi(C, X|Y = y) \tag{6}$$

Given a coloring (or a permutation that we decode into a coloring), we can evaluate this function directly given that we know the topology of the graph and the joint distribution.

Because we cannot derive a minimum objective value to use as a stopping condition, we run the GA for a fixed number of generations and report the best observed individual after these runs.

3.2 Mutation

For our mutation operator, we use the swap operator as described in Eiben et al. [7], which randomly swaps two adjacent vertices. We fix a mutation rate inversely proportional to the number of the vertices in the graph so as to swap a constant number of pairs of vertices in expectation. This parameter was tuned experimentally for each class of graphs that we considered. We also investigated varying the distance of the swap: rather than swapping two adjacent nodes, which may hardly change the overall coloring, we considered all pairs of nodes that are separated by a distance d, for fixed d. Experiments confirmed our intuition that

increasing d increased the amount of change the mutation operator induced. Higher swap distances were more likely to cause shakeups in the color classes, and they generally increase the number of color classes. However, this difference did not propagate to a difference in the performance of the GA. The performance difference between mutation operators with different swap distances is dwarfed by the variation among GA trials. Therefore, we fixed the swap distance at one and proceeded to examine crossover operators.

3.3 Crossover

We considered one crossover operator, the PX operator described in [8]. This operator produces a child by reordering a subset of the nodes in the first parent according to their order in the second parent. In our experiments, we observed that using PX appears to improve performance slightly, though again, the variance between individual samples is much more significant than this effect. However, the added computational cost of this operator is fairly small, so in light of the potential performance improvement, we incorporated the PX operator into our GA.

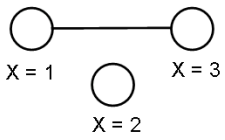
4 Results

4.1 Performance on Sample Problem Instances

In order to study the effectiveness of the overall functional compression scheme, we must start with a joint distribution and function of two variables, create the confusability graph, and compare the encoding from functional compression with the basic Slepian-Wolf encoding of the two correlated sources. Figure 3 shows a simple example of constructing a confusability graph from a table of function and probability values.

f, p	$Y=1$	$Y=2$	$Y=3$
$X=1$	$0, \frac{1}{7}$	$1, \frac{1}{7}$	$1, \frac{1}{7}$
$X=2$	$0, 0$	$1, \frac{1}{7}$	$1, \frac{1}{7}$
$X=3$	$1, \frac{1}{7}$	$1, \frac{1}{7}$	$0, 0$

(a)



(b)

Fig. 3. Example function f and joint distribution p over two random variables X and Y, each of which takes values in the set $\{1, 2, 3\}$. The figure shows the confusability graph of X given Y. There is an edge between $X = 1$ and $X = 3$ because they are confusable when $Y = 1$, but no other pair of X values is confusable.

Intuitively, the algorithm will be much more effective when there are relatively few edges in the graph and many nodes can be given the same coloring. We generated joint distributions and functions that would yield confusability graphs with varying edge densities. The density of a confusability graph becomes quite high if the function takes a wide range of values and most combinations of X and Y occur with positive probability. This fact follows from the definition of the confusability graph. Therefore, we used $\{0,1\}$-valued functions and joint distributions with many "holes" (i.e. (x,y) pairs that occur with probability 0).

We randomly generate each joint distribution as follows: first, we set a parameter $\delta \in [0,1]$ to indicate the probability that a given entry in the table should be nonzero, then we choose each entry to be zero or nonzero according to this probability, and finally we normalize the table such that all nonzero entries are equiprobable and sum to one. If δ is set to a higher value, the graph will be denser, since the joint distribution is less helpful in allowing us to disambiguate values. The function table was generated in a similar fashion, with a parameter $\epsilon \in [0,1]$ chosen to be the probability that a given $f(x,y)$ would be 1. The most dense graph possible is achieved by $\epsilon = 0.5$, since a biased choice of function values will cause more of the (X,Y) pairs to be equal. By adjusting these parameters, we were able to create graphs of essentially any density for which we know $f(x,y)$ and $p(x,y)$.

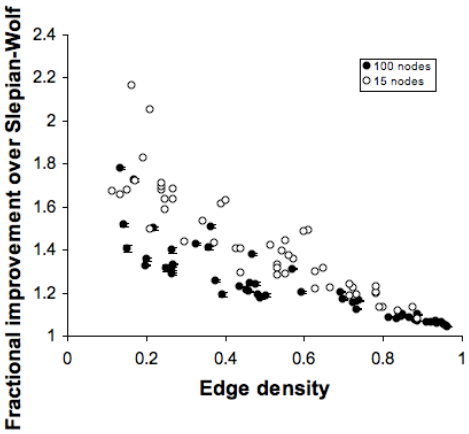


Fig. 4. Graph of the performance of the GA relative to the Slepian-Wolf bound for fifty 15-node and fifty 100-node graphs of varying edge densities, with 5 trials run on each graph. The y-axis shows the averaged values of $\frac{H_G^\chi(C,X|Y)}{H(X|Y)}$ for each graph, the fractional improvement over the Slepian-Wolf bound given by the best coloring C discovered by the GA. Error bars representing the standard deviation are shown for the 100-node graphs; in the 15-node case, the GA discovered identical (presumably optimal) values in all trials for all but two of the graphs.

In Figure 4, we display our results on graphs of varying densities created using this scheme, using for each graph a GA of 200 individuals iterated for 50 generations. We compare the encoding rate achieved by the GA (i.e. the lowest chromatic entropy value we could find) to the Slepian-Wolf bound, which can be thought of as the chromatic entropy of the graph where every node is colored differently. For high densities, it is difficult to garner much improvement, as we need to use many relatively small color classes when coloring the graph. However, for low densities, we can improve on the Slepian-Wolf bound by 30% or more, which translates to a 30% reduction in the amount of data that needs to be transmitted in the original problem.

4.2 Performance on Random Graphs

Though these graphs do represent a simple class of problem instances that are theoretically appropriate, our particular method of constructing functions and joint distributions might be somehow restricting the range of instances we are considering. We would like to be able to demonstrate our algorithm's performance in a general setting and argue that it will do well at minimizing chromatic entropy for any given graph. To do this, we will construct graphs of all densities with completely random structures (i.e. all edges are have an equal, fixed probability of being included), and then experiment with a range of probability distributions on the nodes. Although these graphs will generally have no particular interpretation in the context of functional compression in which edge structure is linked to the joint distribution, it does allow us to convince ourselves of our algorithm's viability for applications other than confusability graphs.

We generated random 100-node graphs of various densities using Culberson's graph generator, described in [9]. In the generation of these graphs, edges are included or not included independently and with a fixed probability, so graphs produced in this manner will have no special structure. We then attached sensible joint distributions to these graphs. We induce an ordering on the n vertices and define a random variable X that takes values over the vertices $v_1, ..., v_n$. We also define a random variables Y that takes values $y_1, ..., y_n$ not associated with the vertices. Our joint distribution over these is as follows:

$$P(x_i, y_j) \propto \exp\left\{-\frac{1}{2}\left[i-\frac{n}{2}, j-\frac{n}{2}\right]\begin{bmatrix} s & -cs \\ -cs & s \end{bmatrix}\left[i-\frac{n}{2}, j-\frac{n}{2}\right]^T\right\} \qquad (7)$$

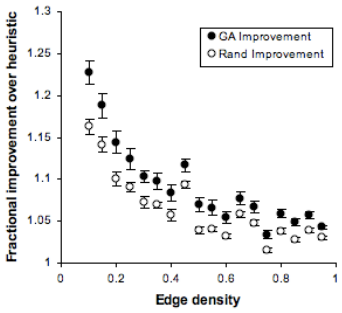
$s > 0$ and $c \in (0, 1)$ in the inverse covariance matrix term are constants so that we can tune the degree of variance and covariance in the parameters, respectively. The motivation behind this was to create distributions with varying degrees of correlation between X_i and Y_j in order to evaluate the GA on a variety of instances.

We will compare the GA performance to a heuristic and random search. The heuristic is a deterministic method that is independent of the any ordering of the nodes in the graph. Its goal is to maximize probability mass on high-entropy color classes as much as possible because this frequently minimizes chromatic

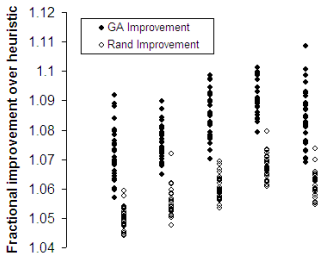
entropy. It greedily colors the nodes in the graph in descending order of marginal probability mass $P(X)$. To each node, it assigns the color whose color class has the highest current weight in terms of $P(X)$ and that admits a legal coloring. The random search examines as many random individuals in the search space as the GA evaluated for fitness over an entire run, and returns the best among these.

Experiments showed that the performance of the GA relative to the heuristic and random search was independent of the covariance matrix, so we fix $s = 4$ and $c = 0.5$. We know from Section 4.1 that density is an important factor for the performance of the algorithm.

We fixed the number of individuals in the population at 200 and the number of generations at 50, to maintain consistency with the experiment in Section 4.1. The results are shown in Figure 5. Both the GA and random search consistently outperform the heuristic. In addition, we see that the GA consistently does modestly better than random search, with Wilcoxon sign-rank tests yielding p-values of less than 2×10^{-6} for every graph under investigaton. Given the small marginal computational cost to implement the machinery of the GA, we prefer the GA to the random search.



(a) Averaged GA vs averaged RAND on graphs of varying density



(b) Thirty trials of GA vs RAND on each of five graphs of density 0.5

Fig. 5. (a) shows the performance of the GA and RAND relative to the heuristic for 100-node random graphs of varying edge densities. Each point represents the average performance of the given method on the graph averaged over 30 trials, with error bars showing the standard deviation. The y-axis shows $\frac{H_G^X(C_{\text{Result}}, X|Y)}{H_G^X(C_{\text{Heuristic}}, X|Y)}$ for each graph, the fractional improvement in chromatic entropy for the coloring found by the method over the chromatic entropy of the heuristic coloring. Wilcoxon sign-rank tests established the statistical significance with a p-value of less than 2×10^{-6} for each point. (b) compares GA and RAND more closely to show that GA almost always performs better than RAND across multiple trials and multiple graphs of the same density, and therefore we are justified in using averages on one graph to contrast the two.

5 Conclusion

In this paper, we implemented a GA to minimize conditional chromatic entropy for a given graph with a joint probability distribution over its vertices. The GA employs an order-based representation combined with a greedy coloring heuristic. We applied this algorithm to the problem of functional compression with side information, wherein the minimization of the conditional chromatic entropy of a confusability graph supports approximation of the optimal encoding rate. This scheme allowed us to improve Slepian-Wolf encodings by around 30% for relatively sparse confusability graphs, and furthermore, the algorithm routinely outperformed random search and a very inexpensive heuristic.

Future work along a theoretical avenue to pursue is fitness landscape analysis. Chromatic entropy is an interesting problem in that there is something of a continuous nature to it as a result of having a probability distribution over the vertices. Changes in the color class assignments of very low probability vertices have correspondingly small effects in the overall chromatic entropy calculation. This could yield a fitness landscape that is atypical of combinatorial optimization problems, which could in turn suggest even better algorithms to solve it efficiently.

References

1. Doshi, V., Shah, D., Medard, M., Jaggi, S.: Distributed functional compression through graph coloring. In: Data Compression Conference, 2007. DCC '07. (March 2007) 93–102
2. Cardinal, J., Fiorini, S., Van Assche, G.: A graph coloring problem with applications to data compression (2004)
3. Slepian, D., Wolf, J.: Noiseless coding of correlated information sources. Information Theory, IEEE Transactions on **19**(4) (1973) 471–480
4. Körner, J.: Coding of an information source having ambiguous alphabet and the entropy of graphs. In: 6th Prague Conference on Information Theory. (1973) 411–425
5. Alon, N., Orlitsky, A.: Source coding and graph entropies. IEEE Trans. Inform. Theory **42** (1995) 1329–1339
6. Dorne, R., Hao, J.K.: A new genetic local search algorithm for graph coloring. In: In Parallel Problem Solving from Nature - PPSN V, 5th International Conference, volume 1498 of LNCS, Springer-Verlag (1998) 745–754
7. Eiben, A.E., Van Der Hauw, J.K., Van Hemert, J.I.: Graph coloring with adaptive evolutionary algorithms. Journal of Heuristics **4**(1) (1998) 25–46
8. Sivanandam, S.N., Sumathi, S., Hamsapriya, T.: A hybrid parallel genetic algorithm approach for graph coloring. Int. J. Know.-Based Intell. Eng. Syst. **9**(3) (2005) 249–259
9. Culberson, J.C., Luo, F.: Exploring the k-colorable landscape with iterated greedy. In: Dimacs Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society (1995) 245–284