DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **00**, 0000

Coloring by Tabu Branch and Bound

FRED GLOVER, MARK PARKER, AND JENNIFER RYAN

20 March, 1995

Abstract. We give an adaptive depth procedure for coloring a graph that combines elements of tabu search and branch and bound. The resulting **tabu branch and bound** method can execute searches of varying degrees of exhaustiveness at different stages and in different regions of the search space, ranging from pure heuristic search to pure tree search at the extremes. The goal is generally to find near optimal colorings efficiently, but with the ability to obtain optimal colorings if desired. We introduce new theoretical results concerning **depth and width** of local optima that motivate our approach, and employ a concept of **color conditioned dependency** to permit shrinking of the graph at appropriate stages. We also employ an associated notion of **node danger** to select a node to be colored and to determine the color assigned.

Computational experiments show our method performs effectively for a variety of problem types, including those taken from real world applications, and notably in graphs that are not excessively dense (as is typical in practice). In general, our approach is significantly better than branch and bound coloring methods which are considered to be state-of-the-art, yet maintains a structure that likewise provides a theoretical ability to assure optimality (controlled by a choice parameter). We demonstrate the effectiveness of this approach by comparing the tabu branch and bound method in both heuristic and exact mode with an efficient implementation of the DSATUR branch and bound algorithm. We find that the tabu branch and bound approach is able to solve problem instances in 1/3 to 1/2400 of the time as DSATUR. Also, when run in a mode to identify near optimal colorings quickly, the tabu branch and bound method terminated with optimal solutions for 15 of 28 graphs taken from the DIMACS Challenge Benchmark (specially constructed to be difficult) for which optimal solutions are known. Additionally, by finding colorings of the same cardinality as lower bounds the method also identified, the method was able to verify the optimality of its solution in 12 of these 15 problems.

1991 *Mathematics Subject Classification.* Primary 90C35, 68R10; Secondary 05C85.

This research was supported in part by AFOSR/ONR Contracts #F49620-90-C and #F49620-92-J-DEF.

This paper is in final form and no version of it will be submitted for publication elsewhere.

1. Introduction

Our goal is to develop an efficient heuristic for near optimal colorings on a general class of graphs with an additional *specific* capability to solve optimally *with verification* when desired. In meeting this objective, we introduce new results on the depth and width of local optima in graph coloring problems. Our final goal is to develop this heuristic within a tabu search metastructure.

It has long been established that branch and bound (tree search) is somewhat limited in its effectiveness as a mechanism for coloring. The studies of Hertz and de Werra [**8**] and Mohr [**10**], for example, yield results using the more flexible memory structures of tabu search that have not been matched by branch and bound procedures. (These results have also been demonstrated superior to those of heuristic approaches such as simulated annealing and genetic algorithms.) One of our goals is to determine whether some of the principles of tabu search can be directly superimposed upon a branch and bound structure to produce a method which is better than branch and bound, but which also offers a theoretical guarantee of finiteness by selecting parameters at boundary values. At issue is the manner in which such a combined method can be usefully organized, and the degree to which the customary effectiveness of tabu search is compromised by creating such a combined approach. We address these considerations by developing new heuristic choice rules that are useful for such a combined method, and by establishing theoretical results for guiding the "search depth" of the procedure. Our results also can be applied with more customary tabu search memory structures, and hence provide an opportunity for investigations outside the branch and bound context which is our present focus.

In order to meet our initial objective, we find it necessary to quickly establish bounds on $\chi(G)$, the chromatic number of our graph G. In this way, we reduce the size of the search space for the tabu branch and bound phase of our heuristic. A lower bound can be obtained by finding a clique in the graph, although finding the maximum clique is theoretically as difficult as the original coloring problem. We present a heuristic for quickly identifying large cliques which works well on many classes of problems. Establishing an upper bound k means finding a coloring on k colors. A sequential coloring algorithm can be implemented efficiently for this purpose. In particular, we develop a dynamic sequential coloring heuristic, called "DANGER", whose ideas provide a generalization of the notions underlying the DSATUR heuristic [**1**].

A fundamental part of our work is embodied in the new results on analysis of the depth and width of local optima for graph coloring formulations. Previously, Ryan [**12**] has analyzed the 0-1 knapsack problem and set covering problem and defined sharp bounds on the depth and width of local minima. Here, these concepts are applied to the graph coloring problem to obtain sharp bounds for the depth and width of local minima. These results can be used by *any* search heuristic which discretizes the solution space. We present only a single instance of their application.

The tabu search metaheuristic has been successfully implemented on numerous combinatorial optimization problems in the recent past (see [**6**] for a brief summary). Hertz and de Werra [**8**] describe an implementation for graph coloring where an infeasible coloring is kept, and the tabu search is used to reduce the number of edges between nodes in the same color class by swapping nodes between color classes. A feasible k coloring is obtained when each color class contains no edge with both endpoints in the color class. Mohr [**10**] explores parallel implementations of this same approach. Tabu search can be implemented in a variety of ways on a given problem. Here, we explore the use of tabu search to control a branch and bound procedure which improves the coloring solution obtained via our dynamic sequential coloring heuristic. We use a different move mechanism from the Hertz and de Werra approach, and we utilize the tabu search framework in a very different manner.

2. DANGER Coloring Heuristic

We develop DANGER to serve as the core of our heuristic. It provides an upper bound during the first phase of our heuristic and is used during the recoloring phase of our tabu branch and bound process. We chose to use a dynamic sequential coloring heuristic rather than an iterative coloring heuristic so that a feasible solution is maintained. This can be especially important for large coloring instances where an exact coloring is (in most cases) impossible to obtain.

The DANGER coloring heuristic is based upon the concept of a dynamic node danger measure which is used to prioritize the uncolored nodes at each iteration. In addition, we also make use of the idea of color conditioned dependency to shrink the size of the graph we are working with. We begin by describing this process.

2.1. Recursive Shrinking.

Consider the following simple idea. Let **max_color** be the largest color index allowed, hence the largest number of colors permitted to be used in coloring of graph G (during the current "phase"). If there is any node i with **degree**(i) $<$ **max_color**, then classify node i as a "dependent" node. Clearly, if we wait to color a dependent node until after all other nodes of G are colored, then we can assign a dependent node any color we want different from the colors of its neighbors, and be assured that we do not exceed the target of using (at most) **max_color** different colors.

Consequently, this suggests a strategy of shrinking the graph by removing all dependent nodes, and coloring the graph that remains. Then the dependent nodes can be reinserted, and the coloring process can be completed.

But we can immediately do better, because once a dependent node is removed, the degree of each of its neighbors is reduced by 1, and hence one or more of its neighbors may become dependent. This conditional dependence gives a basis for further shrinking the graph, and it is only necessary to keep track of the sequence in which nodes are removed in this recursive shrinking, in order to assign colors properly when the graph is once again expanded to full size after coloring the residual graph. (That is, the rule is simply to re-construct the full graph in the sequence that reverses the shrinking sequence.) It is easy to prove that conditional dependence is as valid as first order dependence to assure the graph does not receive too many colors. In fact, the proof is given by the coloring process of the reverse re-construction.

2.1.1. *Generalization - color conditioned dependency.* We can generalize the basic notion of dependency to give a much more powerful strategy which operates in a graph that is already partially colored. In effect, we look for conditional dependency that is based on the colors currently assigned.

Define **colored**(i) and **uncolored**(i) to be the number of neighbors of node i that are colored and uncolored; hence we have

$$\mathbf{colored}(i) + \mathbf{uncolored}(i) = \mathbf{degree}(i)$$

Let **different_colored**(i) denote the number of different colors assigned to neighbors of node i and define

$$\mathbf{potential\_difference}(i) = \mathbf{different\_colored}(i) + \mathbf{uncolored}(i)$$

In other literature, e.g [**1**], **different_colored**(i) is sometimes called the *saturation degree* of node i. Here we embed this measure as indicated in a new measure that we can exploit rigorously. In particular, **potential_difference**(i) identifies a bound on the value that can be achieved by **different_colored**(i) in any completion of the current partial coloring of G. By extension of our previous observations, whenever we find

$$\mathbf{potential\_difference}(i) < \mathbf{max\_color}$$

then we may classify node i as **dependent by color conditioning**, or **cc-dependent**. Our first result, on which we will base a specialized coloring procedure, is therefore as follows.

THEOREM 2.1. *Let v be the minimum number of colors required to color G, given the current partial coloring. Then all cc-dependent nodes can be eliminated from G, and all nodes that conditionally become cc-dependent as a result of this operation likewise can be removed recursively, without changing the value of v.*

The recursive aspect of this result arises since, similar to the earlier simpler case of conditional dependency, the removal of some of the nodes reduces the degrees of their neighbors and so causes other nodes to become cc-dependent (since reducing **degree**(i) reduces the value of **uncolored**(i), and hence reduces the value of **potential_difference**(i)). This outcome can therefore trigger a chain of changes at various stages of the coloring process.

2.1.2. *Branch and bound effects.* If we just restrict attention to how recursive shrinking can affect branch and bound, we see the outcome can be considerable. Each time a node is removed – rather than colored, for example – then we never have to backtrack to this node and try to assign it colors other than the one first attempted. This has the potential to eliminate large portions of the branch and bound tree. Consequently, a branch and bound method that incorporates this strategy will tend to dominate any such method that does not. Beyond this, the shrinking process is designed to improve the effectiveness of any approach that uses "controlled variable backtracking", as does the method presented here.

This idea is easily implemented. When the graph is initially read in, **uncolored**(i) is set to the number of neighbors for each node i. As a node j is colored, **different_colored**(i) is updated for all neighbors i by looking at whether j's color is available to it. If it is, then j is the only neighbor of i that is colored that color and **different_colored**(i) is updated. In either case, **uncolored**(i) is decremented, **potential_difference**(i) is recalculated and node i is checked for to see if it is **cc-dependent**. If it is, its color is set to 0. While backtracking (linearly), we check to see if a node has color 0. If so, we do not need to try to change its color. Upon completing a coloring, the **cc-dependent** nodes are assigned an actual color.

Now that we have seen how to restrict the size of our graph at each iteration, we present the node and color selection rules for the DANGER heuristic.

2.2. Node Danger. At each iteration of our DANGER heuristic, we must select an uncolored node from the graph as our next node to color. We introduce the simple concept of *node danger* – at step k of our heuristic, we wish to know: *How dangerous is it if we* **do not** *color node* i? Several criteria are used to define this danger measure. A node with a high value of **different_colored**(i) and of **uncolored**(i), if not colored at once, will potentially become more and more difficult to color within the targeted color number bound. Also, if the number of colors available to a node *and* to its uncolored neighbors is large relative to the number of colors available to the node, generally it will be difficult to color that node and its neighbors. Define **avail**(i) to be the number of colors available to node i, and **share**(i) to be the number of colors available to node i that are also available to its uncolored neighbors. At each iteration, these "danger" attributes are combined in the following sense to weight a subset of uncolored nodes:

Define

$$\mathbf{danger}(i) = F(\mathbf{different\_colored}(i)) + k_u \mathbf{uncolored}(i) + k_a \frac{\mathbf{share}(i)}{\mathbf{avail}(i)}$$

where F is a monotonic increasing function and k_u and k_a are nonnegative constants.

Note if F is the identity function and $k_u = 1$ and $k_a = 0$, then **danger**(i) = **potential_difference**(i); that is, the danger represented by node i is the number of different colors that may potentially be assigned to its neighbors. Then

danger$(i) <$ **max_color** discloses that node i is not dangerous enough to worry about, and we may eliminate it by cc-dependency as previously described.

The function F is taken to be different from the identity function because the "real" (or probable) danger of node i depends much more strongly on the number of different colors used by its neighbors than by the value of **uncolored**(i). Various choices of F were considered in our empirical studies. We determined the following form to be particularly promising.

Consider the function $F(y)$. Note y ranges from 0 to **max_color** $- 1$, hence the value $v =$ **max_color** $- y$ ranges in reverse from **max_color** to 1. Let C be an arbitrary positive constant. Then let

$$F(y) = \frac{C}{(\textbf{max\_color} - y)^k}$$

As v gets close to 1, the "criticality of F" becomes more pronounced. That is, the ratio $\frac{F(y+1)}{F(y)}$ increases as y grows. Thus, we are more likely to use the "secondary" parameters of **danger** in selecting a node when **different_colored** is small for all nodes; and as **different_colored** approaches **max_color** these secondary parameters are used only to break ties. The effect of this rule may be further understood by contrasting it with the popular approach embedded in the DSATUR algorithm, which relies chiefly on **different_colored** (or saturation degree) to select nodes, while invoking **uncolored** to break ties. This approach makes no reference to **avail** or **share**, and incorporates no functional relationship as in the danger concept. Consequently, our rule provides a versatility that derives from using fuller information and establishing a parameterized interdependence among components.

The **danger** value is used to choose the next node i^* to color, by picking the node achieving max(**danger**(i) : for i uncolored). This requires node i^* to be a node with the highest, or one of the top two or three highest values of **different_colored**(i). In our implementation, we choose i^* from the set of nodes having one of the three highest values of **different_colored**(i). In this way, we reduce the computational effort at each iteration, and typically evaluate **danger** for less than 10% of all nodes.

In practice, we have found that the following parameter values work well over a large class of graphs: $C = 1.0$, $k = 1.0$ $k_u = 0.025$, and $k_a = 0.33$. Improvements of course are possible if parameters are fine tuned for graphs with particular structures, but our goal has been to provide a coloring procedure that works well across a broad range of graphs without taking advantage of special characteristics. In fact, as noted later, the speed of our method makes it possible to incorporate 5 increments in the parameter k, holding other parameters constant, to achieve additional robustness while maintaining a high level of efficiency.

We now focus our efforts on selecting a color for i^*.

2.3. Color Danger. Once we have chosen a node to color, we want to select a color that is least likely to be required by neighboring nodes at later iterations. We present the concept of *color danger*, which is similar in form to the node danger idea just presented. Loosely speaking, we will say that a color is *attractive to* a specified node if our rules would judge it to be a good choice for that node. Then we will consider a color to be *dangerous* to use for coloring node n if this color is:

- attractive to an uncolored node with a large value of **different_colored**,
- attractive to an uncolored node with many uncolored neighbors,
- infrequently used (we prefer to use colors that are already used extensively).

For a particular color c, define **diff_neighbors**(c) to be the maximum number of different colored neighbors, over all uncolored nodes having c available as a color. Let n_c be the node achieving this maximum. Let **num**(c) denote the number of times that c has been used. We choose to use the color c that minimizes the quantity

$$\frac{k_1}{(\mathbf{max\_color} - \mathbf{diff\_neighbors}(c))^{k_2}} + k_3\mathbf{uncolored}(n_c) - k_4\mathbf{num}(c))$$

where k_1, k_2, k_3, and k_4 are nonnegative constants.

Again, this formulation is similar to that defined for node danger, where our functional choice for **diff_neighbors** requires that our color danger value increase rapidly as **diff_neighbors** approaches **max_color**.

In practice, we have found that the following parameter values work well over a large class of graphs: $k_1 = 1.0$, $k_2 = 1.0$ $k_3 = 0.5$, and $k_4 = 0.025$. As described later, these parameters are maintained fixed throughout the application of our procedure with the exception of k_2.

We use the DANGER heuristic to dynamically and sequentially color a graph, making use of new results which form the crux of our tabu branch and bound strategy to improve upon this coloring.

3. Depth and Width of Local Optima

We can view a solution to our coloring problem as a local minimum in the topology defined by our move mechanism. We develop a theory to enable us to determine how far we must move away from our current solution in order to improve upon it, and apply this theory within the tabu branch and bound phase of our heuristic to demonstrate its use in practice. Once we have obtained a valid coloring on k colors, we attempt to improve upon it, by setting $lower\_bnd \leq k_{new} < k$. We then backtrack to a prescribed level in the branch and bound tree (i.e. uncolor a set of previously colored nodes) and resume our search down another branch of the tree (i.e. continue coloring nodes). The theoretical results derived in this section provide an algorithm for guiding this movement through the branch and bound tree. We emphasize that *any* heuristic which discretizes the solution space can make use of this information, or information derived in a similar manner for a different move mechanism. We begin by defining the depth and width of a local minimum for a general problem.

3.1. The Depth and Width of a Local Minimum. Consider a combinatorial optimization problem $\min\{v(x)|x \in X \subseteq R^n\}$, where v is a real valued function and X is a finite set. Heuristics such as simulated annealing [**9**] and tabu search [**4**] search X by "moving" from one element to another.

The structure of the solution space depends on the move mechanism being used. Clearly, in order for any heuristic to be effective, the move mechanism should ensure that each member of X is reachable from any other. A *local minimum* of X is an $x \in X$ such that no $y \in X$ with $v(y) < v(x)$ is reachable from x without first passing through $z \in X$ with $v(z) \geq v(x)$. A *global minimum* of X is an $x \in X$ such that there is no $y \in X$ with $v(y) < v(x)$. Let GL denote the set of global minima, and let L denote the set of local minima which are *not* global minima. The definition of the *depth* of a local minimum below follows [**7**] and [**2**]. The definition of the *width* of a local minimum was introduced in [**12**].

A solution $x \in L$ has *depth* $d(x)$, if $d(x)$ is the minimum d such that some $y \in X$ with $v(y) < v(x)$ can be reached from x without passing through any $z \in X$ with $v(z) - v(x) > d$, i.e., it is the minimum distance uphill that must be traveled in order to escape from the local minimum x. Let $d^* = \max_{x \in L}\{d(x)\}$. In [**7**], Hajek gives a cooling schedule for simulated annealing that is guaranteed to converge in probability to an element of GL. The choice of temperatures in the cooling schedule depends on d^*. Moreover, Chiang and Chow ([**2**] and [**3**]) show that the rate of convergence also depends on d^*. Tabu search can likewise exploit d^*, though by reference to a somewhat different mechanism called strategic oscillation (which does not rely on a monotonic "cooling" operation). More particularly, tabu search can also exploit the notion of width, as follows.

An $x \in L$ has *width* $w(x)$, if $w(x)$ is the minimum w such that some $y \in X$ with $v(y) < v(x)$ can be reached from x without making more than w nonimproving moves. Let $w^* = \max_{x \in L}\{w(x)\}$. The width gives an estimate of the the necessary length of a tabu list in a tabu search. A "flat" topography, i.e., one where the width is large in relation to the depth will impose difficulties for an objective driven search. Tabu search therefore proposes the use of additional measures such as *influence* in such situations [**6**]. Estimates of depth and width for different neighborhood structures can provide further insight and guidance when selecting a neighborhood to be used in such a search.

The following terms will be used in the next sections. An independent set is a set of nonadjacent nodes. A matching is a set of edges M so that no two edges in M share an endpoint. We let $\chi(G)$ denote the fewest number of colors that can be used to color the nodes of graph G and let $\alpha(G)$ denote the size of the largest independent set of G.

3.2. Application to Tabu Branch and Bound. Once a coloring is obtained, we attempt to improve upon it by keeping a fixed number of colors and iterating between improving and nonimproving moves until either we find a feasible coloring or determine that this is not "readily" possible. Therefore, a trial solution will have some, but not necessarily all, of the nodes colored. A "move" involves uncoloring a colored node, or coloring an uncolored node. In our combinatorial optimization setting, the value of a trial solution (to be minimized) is the number of uncolored nodes. The optimal value is 0 if and only if the graph G has a feasible coloring using k colors.

For this topology, which we refer to henceforth as the *fixed number of colors topology*, the depth and width are equivalent. This can be seen by noting that the depth of a local minimum is the distance in terms of objective function that we must move in order to escape the influence of the minimum. Since our objective is measured in terms of the number of nodes colored, this tells us the number of nodes which must be uncolored in order to escape the local minimum. This is equivalent to the definition of width of a local minimum, since this is the number of nonimproving moves necessary to escape the influence of the minimum. Recall that a nonimproving move corresponds to the uncoloring of a colored node. We now derive the value of d^*, and hence the value of w^*.

3.3. Theoretical Results. In any legal graph coloring, the nodes of any one color will always form an independent set. Our solution technique can be considered a set covering approach where we want to cover the nodes with independent sets. Thus we keep a set of k independent sets, and cover as many nodes as possible.

Let $C_1, C_2, \dots C_k$ denote the k independent sets used by the coloring x. (So C_i is colored with color i.) Similarly, let $C'_1, C'_2, \dots, C'_k$ denote the k independents sets used by the coloring x'. We "exchange" C_i and C'_i by uncoloring all of the nodes in $C_i \setminus C'_i$ that are *still colored with color* i, and then coloring the nodes in $C'_i \setminus C_i$ with color i. After $\ell - 1$ steps of coloring and uncoloring, the number of nodes that have been uncolored but not recolored is $\sum_{i=1}^{\ell-1} (|C_i \cap (C'_i \cup \dots \cup C'_k)| - |C'_i \cap (C_1 \cup \dots \cup C_i)|)$. The increase in the number of nodes uncolored will reach a maximum after ℓ sets of uncoloring have been done, and only $\ell - 1$ recolorings, for some ℓ. Define the following:

$$f(\ell) \equiv |(C'_\ell \cup \dots \cup C'_k) \cap (C_1 \cup \dots \cup C_\ell)|.$$

After the ℓth step of uncoloring, (and before the ℓth step of recoloring), the number of nodes uncolored but not recolored is $f(\ell) - |C_\ell \cap C'_\ell|$. (Note that there is no need to uncolor the nodes in $C_\ell \cap C'_\ell$.)

The value of $f(\ell)$ is clearly affected by the ordering of the C_i's and the C'_i's. The following lemmas show how to order these sets to achieve the bound of Theorem 3.1.

Lemma 3.1. *After the $(\ell-1)$st uncoloring sequence, the set $C'_{\ell-1}$ can be selected (possibly after reindexing) from $\{C'_{\ell-1},\dots,C'_k\}$ so that*

$$A \equiv |C'_{\ell-1}\cap(C_1\cup\ldots\cup C_{\ell-1})| \geq \frac{f(\ell-1)}{k+2-\ell}.$$

Proof. Since $C'_{\ell-1}\cap(C_1\cup\ldots\cup C_{\ell-1})\cup\ldots\cup C'_k\cap(C_1\cup\ldots\cup C_{\ell-1})$ is the same as $(C'_{\ell-1}\cup\ldots\cup C'_k)\cap(C_1\cup\ldots\cup C_{\ell-1})$, at least one of the $k+2-\ell$ sets in $\{C'_{\ell-1}\cap(C_1\cup\ldots\cup C_{\ell-1}),\dots,C'_k\cap(C_1\cup\ldots\cup C_{\ell-1})\}$ must cover at least $\frac{|(C'_{\ell-1}\cup\ldots\cup C'_k)\cap(C_1\cup\ldots\cup C_{\ell-1})|}{k+2-\ell}$ nodes. $\square$

Lemma 3.2. *After the $(\ell-1)$st recoloring sequence, the ℓth set to uncolor can be chosen from $\{C_\ell,\dots,C_k\}$ so that*

$$B \equiv C_\ell\cap(C'_\ell\cup\ldots\cup C'_k) \leq \frac{A}{k+1-\ell}+\alpha(G)-\frac{f(\ell-1)}{k+1-\ell}.$$

Proof. Since $(C_\ell\cap(C'_\ell\cup\ldots\cup C'_k))\cup\ldots\cup(C_k\cap(C'_\ell\cup\ldots\cup C'_k))$ is the same as $(C_\ell\cup\ldots\cup C_k)\cap(C'_\ell\cup\ldots\cup C'_k)$, at least one of the $k+1-\ell$ disjoint sets in $\{(C_\ell\cap(C'_\ell\cup\ldots\cup C'_k))\ldots(C_k\cap(C'_\ell\cup\ldots\cup C'_k))\}$ must cover at most $\frac{|(C_\ell\cup\ldots\cup C_k)\cap(C'_\ell\cup\ldots\cup C'_k)|}{k+1-\ell}$ nodes. So without loss of generality we can assume that

$$C_\ell\cap(C'_\ell\cup\ldots\cup C'_k) \leq \frac{|(C_\ell\cup\ldots\cup C_k)\cap(C'_\ell\cup\ldots\cup C'_k)|}{k+1-\ell}.$$

Let $C''_i = C'_i\cap(C_1\cup\cdots\cup C_k)$. Then the above expression is equal to

$$\begin{aligned}
&\frac{|C''_{\ell-1}\cup\ldots\cup C''_k|}{k+1-\ell}-\frac{|(C'_{\ell-1}\cup\ldots\cup C'_k)\cap(C_1\cup\ldots\cup C_{\ell-1})|}{k+1-\ell}\\
&\quad-\frac{|C'_{\ell-1}\cap(C_\ell\cup\ldots\cup C_k)|}{k+1-\ell}.\\
&=\frac{|C'_{\ell-1}\cap(C_1\cup\ldots\cup C_{\ell-1})|}{k+1-\ell}+\frac{|C''_\ell\cup\ldots\cup C''_k|}{k+1-\ell}\\
&\quad-\frac{f(\ell-1)}{k+1-\ell}\\
&\leq\frac{A}{k+1-\ell}+\alpha(G)-\frac{f(\ell-1)}{k+1-\ell},
\end{aligned}$$

where the last inequality holds because each C'_i is an independent set and so $|C''_\ell\cup\cdots\cup C''_k|\leq\alpha(G)(k+1-\ell)$. $\square$

In the following theorem, "asymptotically sharp" means that the bound is sharp except possibly for an $O(1)$ term.

Theorem 3.1. *For the topology we have defined for the graph coloring problem,*

$$d^* \leq \frac{\alpha(G)}{k}\left(\frac{(k+1)^2}{4}\right).$$

and this is an asymptotically sharp bound.

Proof. Consider the sequence of coloring and uncoloring moves described in Lemma 3.1 and Lemma 3.2 above. As noted, the largest difference between the number of nodes originally colored, and the number of nodes currently colored, will be $f(\ell) - |C_\ell \cup C'_\ell|$ for some ℓ. We will prove that

$$f(\ell) \leq \frac{\alpha(G)}{k}\left((k+1)\ell - \ell^2\right).$$

The value on the right hand side achieves its maximum when $\ell = \frac{k+1}{2}$ so that

$$d(x) \leq \frac{\alpha(G)}{k}\left(\frac{(k+1)^2}{4}\right).$$

Thus since $|C_\ell \cup C'_\ell| \geq 0$ the bound on $f(\ell)$ implies the theorem. The proof is by induction on ℓ. The bound is clearly valid when $\ell = 1$. So suppose that $f(\ell-1) \leq \frac{\alpha(G)}{k}\left((k+1)(\ell-1) - (\ell-1)^2\right)$. It is easy to see that $f(\ell) = f(\ell-1) - A + B$, where A and B are as defined in Lemma 3.1 and Lemma 3.2. By Lemma 3.2, we have then $f(\ell) \leq \frac{k-\ell}{k+1-\ell} f(\ell-1) - \frac{k-\ell}{k+1-\ell} A + \alpha(G)$. By Lemma 3.1, $f(\ell) \leq \frac{k-\ell}{k+2-\ell} f(\ell-1) + \alpha(G)$. By induction, this is bounded by $(\frac{k-\ell}{k+2-\ell})(\frac{\alpha(G)}{k})\left((k+1)(\ell-1) - (\ell-1)^2\right) + \alpha(G)$. Simplifying gives the bound of the theorem.

To see the sharpness of this bound, consider the example of Figure 1. The k horizontal circles will denote the sets $C_1, \ldots, C_k$. The k vertical circles denote the sets $C'_1, \ldots, C'_k$. All pairs of nodes not contained in a circle are adjacent. Note that x' colors one node that is not colored by x. (This node is colored by C'_r but "missed" by C_s. Note that all circles contain k nodes except for the circle corresponding to C_s which contains $k-1$. So $\alpha(G) = k$. In order to determine the depth of x we must consider various ways to order the sets when performing the set exchanges. We consider all possible choices of ordering, and the value of $f(\ell) - |C_\ell \cap C'_\ell|$ for all possible ℓ.

Case 1. The sets are ordered so that $s < r$:

Case 1a. $\ell < s$: In this case $f(\ell) = \ell(k+1-\ell)$ and $|C_\ell \cap C'_\ell| = 1$. So $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k+1-\ell) - 1$.

Case 1b. $s \leq \ell \leq r$: In this case $f(\ell) = \ell(k+1-\ell) - 1$ but we still have $|C_\ell \cap C'_\ell| = 1$ since $s \neq r$. So $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k+1-\ell) - 2$.

Case 1c. $r < \ell$: In this case we again have $f(\ell) = \ell(k+1-\ell)$ and $|C_\ell \cap C'_\ell| = 1$. So $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k+1-\ell) - 1$.

Case 2. The sets are ordered so that $s = r$:

Case 2a. $\ell \neq s$: In this case $f(\ell) = \ell(k+1-\ell)$ and $|C_\ell \cap C'_\ell| = 1$. So $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k+1-\ell) - 1$.

Case2b. $\ell = s = r$: In this case $f(\ell) = \ell(k+1-\ell)$ -1 and $|C_\ell \cap C'_\ell| = 0$. So $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k+1-\ell) - 1$.

Case 3. $s > r$: A similar analysis shows that in this case, no matter what ℓ is, $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k+1-\ell) - 1$.

Thus it is clear that the maximum $f(\ell) - |C_\ell \cap C'_\ell|$ can be minimized by choosing $s = 1$ and $r = k$ so that for all ℓ, $s \leq \ell \leq r$ and $f(\ell) - |C_\ell \cap C'_\ell| = \ell(k + 1 - \ell) - 2$. This in turn is maximized when $\ell = (k + 1)/2$ giving a depth of $(k + 1)^2/4 - 2$. Since $\alpha(G) = k$ in this example, we see that our bound is asymptotically sharp. $\square$

FIGURE 1. Sharp example for Theorem 3.1

We now show how this result is used in the tabu branch and bound phase of our algorithm.

4. Tabu Branch and Bound Phase

Once an initial feasible coloring is obtained, we seek to improve upon it in the *tabu branch and bound* phase of our heuristic. At an extreme level, this phase takes the form of a complete tree search. As a heuristic, we have experimented with using the d^* bound derived in the previous section to guide the tabu branch and bound. This result is based upon knowing what set of nodes to uncolor and then recolor to achieve the *most direct route* out of the local minimum. In general, since this information is not known in advance, we assume there exists a sequence of nodes to uncolor. This allows us to take advantage of the result in a backtracking algorithm. We emphasize that this is only one of many possible ways to use this information.

We incorporate the d^* parameter into a tabu search framework as follows. First, note that in addition to obtaining a feasible coloring from the dynamic sequential coloring phase of our heuristic, we also construct a node coloring sequence. This represents a path through a more general branch and bound tree. If we look at uncoloring a subsequence of these ordered nodes, forcing a change, and recoloring, we are exploring a different node sequence or path in the branch and bound tree. The number of nodes which must be uncolored is a function $F(d^*)$ of d^*. $F(d^*)$ would be the identity function if we knew the optimal path back out. However, we do not have this information, and with near certainty will **not** take the optimal path out of the minimum. Define a *backtrack node* to be the earliest node of the current sequence that is uncolored by a backtrack step (hence, the node we "backtrack to"). We seek a range for $F(d^*)$ over which we search for a backtrack node. A node can be a backtrack node if it was not cc-dependent and it has a color available to it in addition to the color currently assigned to it. During a pass where we seek to color the graph with **current_color** colors, and n nodes are presently assigned, we attempt to find a backtrack node that lies in position $n - \mathit{up\_bnd} * d^*$ down to $n - \mathit{lo\_bnd} * d^*$ of the sequence of colored nodes. In other words, we say it is *tabu* to backtrack to a node in position p of this sequence, where $n \geq p > n - \mathit{up\_bnd} * d^*$ or $n - \mathit{lo\_bnd} * d^* > p \geq \mathit{min\_colors}$, where $\mathit{min\_colors}$ is the lower bound on the graph's chromatic number. We allow an *aspiration criterion* to override the tabu status of the first range. This is satisfied if we find no acceptable backtrack node in a position p satisfying $n - \mathit{up\_bnd} * d^* \geq p \geq n - \mathit{lo\_bnd} * d^*$. In practice we have found values of 1 for *up_bnd* and 3 for *lo_bnd* to work well in quickly finding good colorings. As the value of *lo_bnd* increases, the algorithm's behavior approaches that of a traditional branch and bound algorithm.

Thus, the method enhances a traditional depth first search of the branch and bound tree by backtracking to "good" areas for escaping from our local minimum. From the previous section, we have

$$d^* \leq \frac{\alpha}{\textbf{current\_color}} \frac{(\textbf{current\_color} + 1)^2}{4}$$

where **current_color** is the number of colors currently used, and α is the size of the largest independent set. In our implementation, we approximate α by the maximum number of nodes currently colored by any one color.

5. Lower Bound Identification

As the graph is being read in the degree of each vertex is calculated. A large clique will *usually* contain a vertex of large degree. Therefore, we take all vertices having degree among the top three and create a subset of vertices to create cliques on. For each of these root vertices in the set, we attempt to heuristically find a maximal clique containing that vertex. For each neighbor, a greedy algorithm attempts to find the largest clique containing the neighbor and the root vertex. The maximum cardinality clique found is used as the lower bound on the chromatic number for the graph. This heuristic is relatively fast, and tends to find large cliques.

6. The Algorithm

By using the DANGER heuristic to quickly find feasible colorings, and by heuristically identifying large cliques, we obtain an upper and lower bound on the chromatic number of a graph G. We further tighten the upper bound by applying the DANGER algorithm over a small range of parameter values. The node danger formula:

$$\mathbf{danger}(i) = \frac{C}{(\mathbf{max\_color} - y)^k} + k_u \mathbf{uncolored}(i) + k_a \frac{\mathbf{share}(i)}{\mathbf{avail}(i)}$$

uses the following baseline parameter values:

$$C = 1.0, k = 1.0, k_u = 0.025, \text{ and } k_a = 0.33,$$

and the color danger formula:

$$\mathbf{danger}(j) = \frac{k_1}{(\mathbf{max\_color} - \mathbf{diff\_neighbors}(c))^{k_2}} + k_3 \mathbf{uncolored}(n_c) - k_4 \mathbf{num}(c))$$

uses the following baseline parameter values:

$$k_1 = 1.0, k_2 = 1.0, k_3 = 0.5, \text{ and } k_4 = 0.025.$$

From this starting point, we take advantage of the guidelines embodied in the theoretical results of Section 3.3, making use of the strategies described in Section 4 and Section 5. Here we provide an overview of how these pieces fit together to form the tabu branch and bound algorithm.

DANGER Heuristic

(i) *Define lower bound.* Heuristically identify a large clique in the graph.
(ii) Iteratively examine alternate parameter values for the constants in the **danger** node selection and color selection rules.
(iii) For each set of parameter values, color G by the choice rule(s) specified, without a target for **max_color** and without allowing backtracking. (Hence the coloring process simply assigns a color to each node once, and then stops.)
(iv) Identify the parameter values that produced the smallest number of colors in Step 3, and denote this number of colors by U (giving an upper bound for **max_color**). Let L denote the lower bound for **max_color** derived from clique information.
(v) Select a starting value of **max_color** as a function of L and U.

In practice, we use five iterations of steps 2 and 3. The value of the exponent for both the node danger and color danger (k and k_2) are increased by 1 between each run. We also choose the starting value of **max_color** (from step 5 above) to be $U-1$.

Tabu Branch and Bound

(i) **Initialization:** Use the parameters that gave the best coloring in the initial DANGER phase. If **max_color** $= U$ stop.

(ii) **Destructive Phase I:** Backtrack to the first occurrence of color **max_color**$+1$ – call this the n^{th} node.

(iii) **Destructive Phase II:** Backtrack $up\_bnd * d^*$ more nodes, continue backtracking until an eligible node is found or either node $n - lo\_bnd * d^*$ or the maximum backtrack depth is reached. (The maximum backtrack depth is:

$$L + \text{ number of cc-dependent nodes from the initial clique coloring).}$$

If the maximum backtrack depth is reached, set $L =$ **max_color**, **max_color** $=$ **max_color** $+1$ and go to step 1. If node $n - lo\_bnd * d^*$ is reached, then we override the tabu status and search in the range from $n - up\_bnd * d^*$ to $n-1$. If nothing is found in this range, set $L =$ **max_color**, **max_color** $=$ **max_color** $+1$ and go to step 1.

(iv) **Constructive Phase:** Color with the DANGER heuristic, using a target of **max_color** colors.
If no feasible coloring is found in this target, go to step 2.
If a feasible coloring is found, set $U =$ **max_color**, and select a value of **max_color** as a function of L and U and go to step 1.

In practice we update **max_color** with $U-1$ and *lo_bnd* and *up_bnd* are set to 4 and 1 respectively.

7. Results

The tabu branch and bound code was compiled using the Pascal compiler “pc” on a DEC Alpha 3000 computer. Additional runs were made on a Balance Sequent 21000 computer, where the code was compiled using the DYNIX Pascal compiler. All computer times listed in this section are CPU time measurements taken using the UNIX time command, and are expressed in seconds as the sum of user and system time for the execution.

In order for the tabu branch and bound algorithm to work effectively, it must be able to quickly produce good colorings and it must backtrack efficiently. The tabu branch and bound uses the dynamic sequential coloring heuristic DANGER to solve its coloring subproblems. In order to demonstrate the utility of DANGER, a set of comparisons were made between DANGER and the highly efficient version of the DSATUR algorithm coded by Morgenstern [**11**]. We tested the algorithms on a test bed of random problems ranging in size from 100 nodes to 1000 nodes and having edge probabilities from 25 to 75%. Five different problems of each size were created. The results are presented below.

Problem	DANGER		DSATUR	
	Avg. Solution	Avg. Time	Avg. Solution	Avg. Time
$G_{(100,0.25)}$	10.0	0.26	10.2	0.17
$G_{(100,0.50)}$	17.8	0.34	18.4	0.37
$G_{(100,0.75)}$	29.4	0.40	29.8	0.57
$G_{(200,0.25)}$	16.0	0.79	16.4	0.66
$G_{(200,0.50)}$	29.8	1.01	31.2	1.31
$G_{(200,0.75)}$	50.6	1.36	51.2	1.99
$G_{(500,0.25)}$	32.8	3.80	32.8	4.21
$G_{(500,0.50)}$	63.5	6.79	65.0	8.46
$G_{(500,0.75)}$	110.4	8.92	112.4	13.18
$G_{(1000,0.50)}$	113.0	26.26	115.4	37.05

TABLE 1. Comparison of the DANGER and DSATUR sequential coloring algorithms

The DSATUR algorithm was run with no backtracking and a target coloring of 150 to operate as a heuristic approach according to the design specifications of [**11**]. The outcome yields an upper bound on the chromatic number of the graph. It should be noted that this version of Morgenstern's code was optimized for planar graphs. These refinements were focused on heuristic constructive moves rather than on deep backtracking.

To exploit its speed to achieve greater robustness, the DANGER heuristic was run by varying two of its parameters to create 5 different settings for each problem. The best coloring found and the **total** run time are used to calculate the averages presented below. The parameter sets are those identified in Section 6. The value of the exponents in both the node danger and color danger formulae (k and k_2 respectively) are increased by 1 between each of the 5 runs. This has the effect of modifying the criticality of the exponential function over a range that covers a large number of graphs. This set of runs was made on the DEC Alpha 3000.

We see that even without the benefit of the special data structures and code optimizations incorporated into Morgenstern's code, our DANGER approach finds solutions whose quality dominates those obtained by DSATUR, yielding generally better solutions for problems of all sizes and densities except one, where the average outcomes were the same. The DANGER approach also requires less run time except in two cases, consisting of the smallest graph and the least dense graph on 200 nodes. As problem size and density increases, the differences become more pronounced.

The second phase of testing involved examining the effectiveness of the tabu branch and bound in directing the search towards promising areas of the solution space. The d^* parameter described in Section 3 is used to guide the tabu branch and bound heuristic. In order to demonstrate the effectiveness of using this d^*

Problem	TBB with d^*		TBB exhaustive		DSATUR	
	Solution	Time	Solution	Time	Solution	Time
$G_{(20,0.50)}$	6	0.13	6	0.16	6	0.12
$G_{(30,0.50)}$	8	0.12	8	0.12	8	17.69
$G_{(40,0.50)}$	8	0.18	8	0.20	8	47.99
$G_{(50,0.50)}$	10	0.28	9	0.70	9	2403.22
$G_{(60,0.50)}$	11	0.50	10	37.40	10	3712.56
$G_{(70,0.50)}$	13	0.74	12	3108.04	12	9735.92

TABLE 2. Comparison between TBB with d^*, TBB exhaustive search, and DSATUR

parameter, we compare results of this implementation with a full backtracking version of the algorithm, which yields a depth first search of the branch and bound tree. In order to place these results in perspective, we also applied the DSATUR algorithm with backtracking. Since a target color number must be given as input to the DSATUR algorithm, these runs were made last and we selected as its target color number $k - 1$, where k is the chromatic number obtained from the full backtrack version of the tabu branch and bound. This gives DSATUR an important advanced start advantage, and the time reported is *only* the time to verify that no $k-1$–coloring exists. Each of the three algorithms was run on a set of random problems and the results are presented below. The best feasible coloring, and the total number of CPU seconds spent working on the problem are reported. Five iterations of DANGER (with parameter sets as previously defined) were used for the initial coloring, both for the tabu branch and bound method in exhaustive search mode (TBB exhaustive) approach and for the tabu branch and bound method with the adaptive depth parameter (TBB with d^*).

The same parameter sets were likewise used by these two methods for both the initial phase and the backtracking phase. Hence, the only difference in the algorithms was the tabu branch and bound versus unlimited depth first search branch and bound. These runs were also made on the DEC Alpha 3000.

Note that both the TBB with d^* and TBB exhaustive solve a more difficult problem than DSATUR. They must find an initial lower bound, find an initial coloring, improve that coloring, and either prove it optimal or not find a better coloring with the d^* restrictions in place. DSATUR attempts to find a coloring only on $k - 1$ colors. Yet in all problems but the first, which is a 20 node problem, both of the TBB algorithms works significantly faster than the DSATUR algorithm. We also note that the solution time increases much more slowly as a function of problem size for the TBB approaches versus the DSATUR approach. We only begin to see the advantage of utilizing the d^* parameter in the branch and bound phase of the TBB as the problems get larger. As expected, we trade verified optimality for the ability to identify the "quickest" improvements to the

Problem	TBB with d^*		TBB exhaustive	
	Solution	Time	Solution	Time
le450_15b	15	935.6	17	(130132.5)
R250.5	66	339.6	68	(83041.8)
DSJC500.5	65	(96440.1)	65	(163954.3)

Table 3. Comparison between TBB with d^* and TBB exhaustive search

current coloring.

To further explore what happens as problem size increases, we look at the following problems from the DIMACS challenge test bed. Time results contained in parenthesis were time limit terminated rather than algorithm terminated. The best feasible coloring and the total number of CPU seconds spent working on the problem are reported. The feasible colorings found by exhaustive search are in fact the same solutions found during the initial DANGER coloring phase, that is, the depth first search was overwhelmed by the combinatorial complexity of the problem. These results were obtained on the Sequent Balance 21000.

We see that for the leighton graph and the geometric random graph (le450_15b and R250.5 respectively), the TBB with d^* offers a significant improvement in much less time over the colorings obtained by the TBB exhaustive method.

The third graph is a specially constructed graph which has an optimal coloring of 25, but has other characteristics (density) of a random graph on 500 vertices with 50% edge probabilities. In this case, the TBB with d^* has as much trouble as TBB exhaustive. Neither improved the initial coloring (of 65) and terminated after approximately 26 and 45 hours respectively. This particular graph appears to belong to an exceptional category.

Additional test results with a larger range of problems from the DIMACS test bed are shown in Table 4. These runs were made on the DEC alpha 3000. In this table, "clique" is the cardinality of the largest clique found heuristically and is used as a lower bound on chromatic number by the algorithm. "Soln." is the best feasible coloring found. "Time" is the total cpu time, where the algorithm terminates either by exhausting available alternatives determined by the choice of d^*, or by meeting a one hour time limit that was imposed by the user. In the latter case, the time is enclosed in parenthesis. If the solution is optimal, a "Y" is in the optimal column. If the solution is known to be non-optimal, an "N" is in this column, otherwise a "U" for unknown is in the optimal column. The column headed "DANGER" gives the initial upper bound on the color number found by the DANGER heuristic. The "best known low. bound" heading indicates either the size of the largest clique found for this problem or the chromatic number for this problem. Thus, the value in this column is a lower bound on the chromatic number of the graph. In many instances, the lower bound is simply the largest clique identified by this algorithm.

The problems are separated into different classes according to their construction. The first set of graphs is leighton graphs, each 15-colorable by construction. Two of the problems are rather sparse, and the second two are dense. Note the significant decrease in performance for the denser graphs.

The second set of graphs represents real life problems. The first eight are register allocation problems, and are easily solved by our method. The next two are class scheduling problems, which are more difficult and seem to have characteristics similar to random graphs.

The third set of graphs consists of geometric random graphs and their complements (those ending with a "c"). The structure of these problems, both for the original graphs and their complements, is exploited quite effectively by our tabu branch and bound algorithm.

The fourth set of graphs constitutes a family of specially constructed graphs having characteristics of a $G_{(x,y)}$ random graph, but a feasible coloring much smaller than the expected color number for such a graph. The initial colorings obtained by the DANGER algorithm appeared to be reasonably good, and the TBB with d^* found it very difficult to improve upon them. In this case the color numbers are only slightly better than those obtained by the DSATUR algorithm.

The last class of difficult graphs are "flat" graphs specially constructed on 300 vertices to have guaranteed color numbers and "hidden" cliques.

The TBB with d^* method terminated with an optimal solution on 15 of 28 difficult graphs for which optimal solutions are known. Additionally, by identifying a clique of the same cardinality as its coloring, the heuristic was able to identify the optimality of its solution in 12 of these 15 problems. It performed especially well on the register allocation problems and geometric random graphs. The evidence of these efforts indicates that the d^* theory has merit in application to dynamically variable backtracking. In general, we are able to quickly backtrack to areas that allow coloring improvements. This appears to be especially true for graphs of lower density. At the same time, our experiments suggest that tying a tabu search approach to a branch and bound structure can sometimes constrain the operation of the tabu search component too rigidly. In particular, it appears that if the density of a graph reaches a "critical" level, the combinatorics of backtracking are too difficult to overcome even using the d^* information. Nevertheless, our results support the general finding that for many problems, including those taken from real world applications, the coupling of tabu search with branch and bound produces a very effective method, and performs significantly better than a tailored branch and bound procedure on its own.

Second DIMACS Challenge

Coloring Benchmark Results

GENERAL INFORMATION Authors: Fred Glover, Mark Parker and Jennifer Ryan

Title: Coloring by Tabu Branch and Bound

Name of Algorithm: Tabu Branch and Bound

Brief Description of Algorithm:

Heuristic: Dynamic sequential coloring algorithm combined with tabu branch and bound. Has capability to run in exact mode – when tabu branch and bound is relaxed to complete branch and bound. *Type of Machine:* DEC Alpha 3000

Compiler and flags used: pc -o

MACHINE BENCHMARKS

User time for instances:

`r100.5`	`r200.5`	`r300.5`	`r400.5`	`r500.5`
0.02	0.41	3.55	22.47	86.91

ALGORITHM BENCHMARKS

Authors' Comments: Our implementation is written in pascal, and reads files in the ASCII format rather than the more efficient binary format. Due to storage problems, a number of problems were not run. These are denoted with a “DNR” in the table.

Our algorithm will continue to look for improving colorings until it determines that none are available due to the tabu branch and bound restrictions. The times presented here are the complete run time until termination, and the solution is the best feasible coloring found at time of termination. The times to actually find the best feasible coloring were not determined, but in general most of the cpu time is spent looking for a coloring which would improve upon the solution presented. In most instances, the best feasible coloring is found within the first 1 to 10% of the total computer time. A time limit of 3600 seconds was specified for each run.

Results on Benchmark Instances

Name	Runs (Fail)	Time Avg	Solution Avg
DSJC125.5.col	1	160.31	20
DSJC250.5.col	1	3600.00	35
DSJC500.5.col	1	3600.00	65
DSJC1000.5.col	1	3600.00	117
C2000.5.col	DNR		
C4000.5.col	DNR		
R125.1.col	1	0.38	5
R125.1c.col	1	0.97	46
R125.5.col	1	0.70	36
R250.1.col	1	0.25	8
R250.1c.col	1	48.54	65
R250.5.col	1	61.67	66
DSJR500.1.col	1	0.48	12
DSJR500.1c.col	1	3600.00	87
DSJR500.5.col	1	413.21	126
R1000.1.col	1	1.71	20
R1000.1c.col	1	3600.00	105
R1000.5.col	1	3600.00	248
flat300_20_0.col	1	3600.00	39
flat300_26_0.col	1	3600.00	41
flat300_28_0.col	1	3600.00	41
flat1000_50_0.col	DNR		
flat1000_60_0.col	DNR		
flat1000_76_0.col	DNR		
latin_square_10.col	1	3600.00	130
le450_15a.col	1	18.65	16
le450_15b.col	1	29.72	15
le450_15c.col	1	3600.00	23
le450_15d.col	1	3600.00	23
mulsol.i.1.col	1	0.33	49
school1.col	1	94.83	29
school1_nsh.col	1	32.63	26

References

1. D. Brélaz, *New Methods to Color the Vertices of a Graph*, Communications of the ACM **22:4** (1979), 251-256.
2. T. S. Chiang and T. Chow, *On the Convergence Rate of Annealing Processes*, SIAM Journal of Control and Optimization **26** (1988), 1455-1470.
3. ______ *A Limit Theorem for a Class of Inhomogeneous Markov Processes*, Annals of Probability **17** (1989), 1438-1502.
4. F. Glover, *Tabu Search, Part 1*, ORSA Journal on Computing **1** (1989), 190-206.
5. ______ *Tabu Thresholding: Improved Search by Nonmonotonic Trajectories*, ORSA Journal of Computing (to appear).
6. F. Glover and M. Laguna, *Tabu Search*, Modern Heuristic Techniques for Combinatorial Problems (C. Reeves, ed.), Blackwell Scientific Publications, Oxford, 1993.
7. B. Hajek, *Cooling Schedules for Optimal Annealing*, Mathematics of Operations Research **13:2** (1988), 311-329.
8. A. Hertz and D. De Werra, *The Tabu Search Metaheuristic: How we used it*, Annals of Mathematics and Artificial Intelligence **1** (1990),111-121.
9. S. Kirkpatrick , C. D. Gelatt, and M. P. Becchi, *Optimization by Simulated Annealing*, Science **220** (1983), 621-680.
10. Th. Mohr, *Parallel Tabu Search Algorithms for the Graph Coloring Problem*, preprint (1990).
11. C. Morgenstern , *Improved Implementations of Dynamic Sequential Coloring Algorithms*, TCU Tech Report: CoSc-91-4 (1991).
12. J. Ryan , *The Depth and Width of Local Minima in Discrete Solution Spaces*, Discrete Applied Math (to appear).

U.S. West Chair in Systems Science, Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, Colorado 80309-0419

E-mail address: fred.glover@colorado.edu

Mathematics Department, Campus Box 170, University of Colorado at Denver, Post Office Box 173364, Denver, Colorado 80217-3364

E-mail address: mparker@carbon.cudenver.edu

Mathematics Department, Campus Box 170, University of Colorado at Denver, Post Office Box 173364, Denver, Colorado 80217-3364

Current address: US WEST Technologies, 4001 Discovery Drive, Boulder, Colorado 80303

E-mail address: jryan@uswat.uswest.com

Problem	Clique	Soln.	Time	Optimal	DANGER	Best Known Low. Bound
le450_15a	15	16	18.65	N	17	15
le450_15b	15	15	29.72	Y	16	15
le450_15c	15	23	(3600.00)	N	23	15
le450_15d	15	23	(3600.00)	N	23	15
mulsol.i.1	49	49	0.33	Y	49	49
mulsol.i.2	31	31	0.29	Y	31	31
mulsol.i.3	31	31	0.26	Y	31	31
mulsol.i.4	31	31	0.25	Y	31	31
mulsol.i.5	31	31	0.31	Y	31	31
zeroin.i.1	49	49	0.37	Y	49	49
zeroin.i.2	30	30	0.32	Y	30	30
zeroin.i.3	30	30	0.34	Y	30	30
school1	14	29	94.83	N	29	14
school1_nsh	12	26	32.63	N	26	14
R125.1	5	5	0.38	Y	5	5
R125.1c	44	46	0.97	Y	46	46
R125.5	33	36	0.70	Y	37	36
R250.1	8	8	0.25	Y	8	8
R250.1c	55	65	48.54	N	65	64
R250.5	64	66	61.67	N	67	65
DSJR500.1	12	12	0.48	Y	12	12
DSJR500.1c	65	87	(3600.00)	U	87	85
DSJR500.5	108	126	413.21	U	129	124
R1000.1	18	20	1.71	Y	20	20
R1000.1c	67	105	(3600.00)	U	105	67
R1000.5	191	248	(3600.00)	U	251	191
DSJC125.5	9	20	160.31	N	22	9
DSJC250.5	10	35	(3600.00)	N	37	15
DSJC500.5	11	65	(3600.00)	N	65	25
flat300_20_0	9	39	(3600.00)	N	41	20
flat300_26_0	9	41	(3600.00)	N	42	26
flat300_28_0	10	41	(3600.00)	N	42	28

TABLE 4. Results for TBB with d^* on a subset of the DIMACS test bed