# Approximate Triclique Coloring for Register Allocation

G.Venkatesan* C.Pandu Rangan†

**Abstract**

A graph is said to be a triclique if its vertex set can be partitioned into three cliques i.e if its complement is tripartite. We prove that the coloring problem is NP-complete even when restricted to tricliques whose complements are planar tripartite line graphs. We then proceed to give a factor 1.5-approximation algorithm for the coloring problem on tricliques. In fact, our algorithm works for all graphs $G$ which satisfy $\alpha(G) \leq 3$ and actually guarantees an approximation ratio that is at most $\frac{3}{4} + \frac{n}{4\chi(G)}$. We then indicate how this problem may be useful in the context of the register allocation problem that arises in compiler design.

**Keywords :** Triclique, chromatic number, register allocation graph(RAG), control-flow graph, approximate algorithm.

## 1 Preliminaries

For a graph $G = (V, E)$, a set $X \subseteq V$ is an independent set if no two vertices in $X$ are adjacent in $G$ and it is a clique if every pair of vertices in $X$ are adjacent in $G$. A set $X \subseteq V$ is a vertex-cover if every edge in $E$ has at least one end point in $X$. A graph $G = (V, E)$ is tripartite if $V$ can be partitioned into three independent sets. We call a graph $G = (V, E)$ a **triclique** if its vertex set $V$ can be partitioned into three cliques, i.e if it is the complement of a tripartite graph. Tricliques generalize the notion of *bicliques*, which stands for graphs whose complement is bipartite [8].

For a graph $G$, let $\chi(G)$, $\theta(G)$, $\gamma(G)$, $\omega(G)$ and $\alpha(G)$ denote respectively its vertex-chromatic number (the minimum number of colors required to *properly* color the vertices of $G$), clique cover number (the minimum number of vertex-disjoint cliques required to cover all the vertices of $G$), size of a minimum vertex cover, clique number and size of a largest independent set. It is easy to see that $\theta(G) = \chi(G^c)$ where $G^c$ stands for the complement of $G$. Also $\Delta(G)$ denotes the maximum degree of a vertex in $G$.

Let $I$ be an instance of an optimization (minimization, for definiteness) problem $P$. If $A$ is an algorithm that computes solutions with value $\hat{F}(I)$ and if $F^*(I)$ equals the value of the optimal solution, $A$ is said to be a factor $k$-approximate ($k \geq 1$) algorithm for the problem $P$ if $\left|\hat{F}(I)/F^*(I)\right| \leq k$ for all instances $I$ of $P$. We consider the problem of determining the chromatic number $\chi(G)$ of a triclique $G$. We prove that the problem is NP-complete even if $G^c$ is a planar tripartite line graph with $\Delta = 3$. Recall that the line graph of a graph $G$, denoted $L(G)$, has one vertex for each edge of $G$ and two vertices of $L(G)$ are adjacent if and only if their corresponding edges in $G$ are incident upon a common vertex (of $G$).

In section 3, we give a factor 3/2-approximate algorithm for the coloring problem on tricliques. In section 4, we look at possible applications of this algorithm in the register allocation problem which has a standard graph-theoretic formulation [2, 3].

*Department of Computer Science & Engineering, Indian Institute of Technology, Madras-600 036, INDIA

†Corresponding author. Department of Computer Science & Engineering, Indian Institute of Technology, Madras-600 036, INDIA. Email: rangan@iitm.ernet.in

## 2 NP-completeness

Recall that $\theta(H)$ denotes the minimum number of (vertex-disjoint) cliques required to cover the vertices of $H$. Using a reduction from 3-dimensional matching(3DM), it is easy to show that the triangle cover problem and hence the clique-cover problem is NP-complete on tripartite graphs [6]. However it is possible to prove the following stronger result:

**Theorem 1** *Given an integer $K > 0$, it is NP-complete to determine whether $\theta(H) \leq K$, even for a planar tripartite line graph $H$ with $\Delta(H) = 3$.*

**Sketch of Proof :** Presence in NP is clear. NP-hardness can be proved by a reduction from the vertex-cover problem on a planar cubic graph $G$ [6]. We only sketch the idea, the details may found in [9]. Given a planar cubic graph $G$, construct a graph $G'$ by replacing each edge of $G$ by a path of length 3. Let $H$ be the line graph of $G'$. Then it can be shown that (see [9]) $H$ is a planar tripartite graph with:

$$\theta(H) = m + \gamma(G)$$

where $m$ is the number of edges in $G$. The result follows. □

**Corollary 1** *Given an integer $K > 0$, it is NP-complete to determine whether $\chi(G)$ is at most $K$ even if $G$ is a triclique.*

**Proof :** Follows from theorem 1 since $\chi(G) = \theta(G^c)$ and $G^c$ is tripartite. □.

## 3 Approximation Algorithm

We now give a polynomial time 1.5-approximation algorithm for coloring any triclique $G$. In fact our algorithm works for any graph $G$ with $\alpha(G) \leq 3$. The algorithm is based on the following theorem whose proof we sketch for the sake of completeness:

**Theorem 2** *[4] The problem of finding a set of pairwise vertex-disjoint edges and triangles in a graph $G$ to cover the maximum number of vertices of $G$ can be solved in polynomial time.*

**Sketch of Proof:** Since the algorithm is a bit involved, we just explain the idea; full details may be found in [4]. A vertex-disjoint induced subgraph $H = G[S]$ of $G$ is called *critical* if $H$ does not have a perfect packing comprising of vertex-disjoint edges and triangles but for each $v \in S$, $G[S - \{v\}]$ has a perfect matching. The algorithm parallels the classical matching algorithm [5] in that it grows an alternating forest $A$ which is defined relative to a packing $P$ comprising of edges and triangles in $G$.

The nodes of the forest $A$ may be of two types: *real* nodes which correspond to vertices of $G$ and *shrunk* nodes which correspond to *critical* induced subgraphs of $G$. Each tree of $A$ is rooted at some node (which may be a real node or a shrunk node). The roots of the trees are precisely the nodes that are not "covered" by $P$. Every path in $A$ is a $P$-alternating path. A node of $A$ is termed odd or even depending on whether its distance from the root of its tree is odd or even.

The algorithm tries to find an edge joining an even node $v$ of $A$ to a node $w$ that is not an odd node of $A$. If no such node exists, then $P$ is an optimal packing and the algorithm terminates. Depending upon the nature of $w$, the algorithm either grows the forest or shrinks an odd cycle (similar to the shrinking of blossoms in the classical matching algorithm of [5]) or discovers a $P$-augmenting path and removes the edges in $P$ on this path, adds the edges on this path that were not in $P$ to the packing $P$ and updates the set $A$ and a new forest is grown beginning with the next iteration.

The validity of the algorithm follows from the fact that even nodes represent *critical* subgraphs and hence the perfect packing of any subset of them requires the covering of at least as many odd nodes of $A$. So, if in a packing all the odd nodes are already matched with even nodes, then the packing has to be optimal i.e it covers the maximum number of vertices. Since at most $|V(G)|$ augmentations can occur and, between augmentations either the forest is grown or an odd cycle is shrunk, it follows that the algorithm runs in polynomial time. □

We now propose the following algorithm to color a triclique $H = (V, E)$:
Algorithm *Approx-color-Triclique*
Input: A triclique $H = (V, E)$, $|V| = n$.
Output: A proper coloring of $H$.

1. Form $G = H^c$. /* $\omega(G) \leq 3$ */
2. Run the polynomial algorithm in [4] on $G$ to find a packing $P$ comprising a set of pairwise vertex-disjoint edges $e_1, e_2, \ldots, e_l$ and triangles $T_1, T_2, \ldots, T_k$ such that they cover a maximum number of vertices of $G$ i.e such that $3k + 2l$ is as large as possible. Let $v_1, v_2, \ldots, v_m$ be the vertices of $G$ not covered by $P$. /* $3k + 2l + m = n$ */
3. In $H$, give color $i$ to vertices of $T_i$ for $1 \leq i \leq k$, color $k + j$ to vertices of $e_j$ for $1 \leq j \leq l$ and color $k + l + r$ to vertex $v_r$ for $1 \leq r \leq m$. /* Clearly the coloring is proper */
4. Output the coloring of step(3).

**Observation :** The above algorithm runs in polynomial time.

We now perform the efficacy analysis of the coloring output by the above algorithm:

**Theorem 3** *Let $H$ be a graph with $|V(H)| = n$ and $\alpha(H) \leq 3$. If the above algorithm uses $\hat{F}(H)$ colors to properly color $H$, then*

$$\left|\frac{\hat{F}(H)}{\chi(H)}\right| \leq \frac{3}{4} + \frac{n}{4\chi(H)}$$

**Proof :** Let $G$ be the complement of $H$, then $\omega(G) \leq 3$. Let the minimum number clique cover of $G$ use $k^*$ triangles (3-cliques), $l^*$ edges (2-cliques) and $m^*$ vertices (1-cliques) of $G$, we have $3k^* + 2l^* + m^* = n$ and $\chi(H) = \theta(G) = k^* + l^* + m^* = n - 2k^* - l^*$.

By the nature of the packing $P$ obtained by step(2) of the algorithm, we have:

$$\begin{aligned}
3k^* + 2l^* &\leq 3k + 2l \\
\Rightarrow 6k^* + 4l^* &\leq 6k + 4l \\
\Rightarrow 6k^* + 3l^* &\leq 8k + 4l \\
\Rightarrow 4n - 8k - 4l &\leq 4n - 6k^* - 3l^* \\
\Rightarrow n - 2k - l &\leq \frac{3(n - 2k^* - l^*)}{4} + \frac{n}{4} \qquad (1)
\end{aligned}$$

Now,

$$\frac{\hat{F}(H)}{\chi(H)} = \frac{k + l + m}{k^* + l^* + m^*} = \frac{n - 2k - l}{n - 2k^* - l^*} \leq \frac{3}{4} + \frac{n}{4\chi(H)} \text{ using } (1) \; \square$$

**Corollary 2** *The algorithm Approx-color-Triclique is a factor 3/2 approximation algorithm for the coloring problem on Tricliques.*

**Proof :** Follows from theorem 3 using the facts that a Triclique has $\alpha \leq 3$ and the fact that for any graph $G$, $\chi(G) \geq |V(G)|/\alpha(G)$. □

Note that theorem 3 implies that the algorithm performs much better than a factor of 3/2 for cases where the chromatic number of the graph to be colored is large.

**Remark:** In fact, the above algorithm can be run on an arbitrary graph $G$ and will guarantee an approximation factor of:

$$\frac{\alpha(G)}{2(\alpha(G)-1)} + \frac{|V(G)|(\alpha(G)-2)}{2\chi(G)(\alpha(G)-1)}.$$

This fact may be proved in a manner similar to the proof of theorem 3. Note that this ratio is at most $\frac{\alpha}{2}$.

# 4 Coloring Tricliques and Register Allocation

Register allocation is a very important task for a compiler. The register allocation problem is the problem of designing a strategy for a compiler to decide which variables to store in what registers [1]. In the standard graph-theoretic formulation of this problem [2, 3], we first consider the **control-flow graph** of a program. In this graph the nodes are the instructions of the program and there is an arc from node $i$ to node $j$ if it is possible for instruction $j$ to be executed right after instruction $i$ in some execution of the program.

A variable used in a program is *live* in the range of instructions between its definition and its possible uses. In the graph-theoretic formulation the live range of a variable is a connected subgraph of the control-flow graph. Our goal is to find the minimum number of registers necessary to store all the variables for their entire live range subject to the condition that any two variables that are both live at some common instruction are not stored in the same register. This can be formulated as the clean theoretical question of determining the chromatic number of the *register allocation graph (RAG)* where the vertices are the variables and there is an between vertex $i$ and vertex $j$ if the live ranges of $i$ and $j$ intersect [7].

The control-flow graphs of good *structured* programs have small vertex-connectivity. In [7], Kannan and Proebsting define a program to be structured if the control-flow graph is series-parallel. In their model, each vertex in the control-flow graph represents a clique in the RAG corresponding to those variables that are live at that vertex. Note that this notion is more conservative[1] (i.e this method builds RAGs with more edges and therefore tends to use up more registers) than Chaitin's notion [3] according to which two values interfere if and only if, at any point in the routine, they are both live, they are both available and they are *different*. For example, if we have a code segment like:

$$\begin{array}{l} i = \cdots \\ j = i \\ \vdots \end{array}$$

then $i$ and $j$ do not interfere (as far as we can tell) because they have the same value. But $i$ and $j$ interfere as per the definition we use since their live ranges overlap. However, here we stick to the more conservative (and therefore easier) notion due to [7].

Clearly at most two vertices need to be removed to separate a series-parallel graph and hence if the program is structured, its RAG will be 2-clique-separable, i.e it has two cliques whose removal separates it. This fact along with the optimal coloring algorithm for bicliques (based on matching) has been used in [7] to obtain a factor 2 approximation algorithm for the register allocation problem restricted to structured

[1] We thank an anonymous referee for this observation.

programs. Kannan and Proebsting in [7] suggested a broader class of graphs than series-parallel graphs to handle programming language constructs like the **if**..**then**..**else** statement with short-circuit evaluation, whose control-flow graph is not series-parallel. To the flow graph of such constructs and other ones like a **while**..**do** loop with a break statement, one may associate in addition to the usual entry and exit nodes, a *break* node which represents *short-circuiting* out of the loop to some other point of the program. This means that we relax the upper limit on the connectivity of the control-flow graph from two (series-parallel graphs have connectivity at most two) to three. For such a control-flow graph, the RAG will be 3-clique-separable and we believe that our approximate triclique coloring algorithm will form the first step towards handling such RAGs and thereby obtaining good approximation algorithms for the register allocation problem for programs with more general flow graphs than series-parallel graphs.

Finally, we note one possible use of our approximate triclique coloring algorithm in the factor 2 approximate algorithm for register allocation for structured programs presented in [7]. Let $G = (V, E)$ be a series-parallel control-flow graph with *terminals* $A$ and $B$. The algorithm presented in [7] initially colors the variables passing through the terminals $A$ and $B$ by using the algorithm that optimally colors a biclique and passes these colors as forbidden colors to the appropriate components in the series-parallel decomposition. This is all right if the top-level decomposition is a parallel decomposition. However, if the top level decomposition is a series decomposition at node $C$, then instead of coloring the biclique formed by the variables passing through $A$ and $C$ separately and then reassigning the color of any variable passing through $C$ that conflicts with a color assigned to a variable passing through $B$ as done in [7], we may color the triclique formed by the variables passing through $A$, $B$ or $C$ by our approximate algorithm. In practice, this modification will reduce the number of colors used by the algorithm.

The reason why we are unable to reduce the approximation ration of 2 achieved in [7] is that the algorithm in [7] requires a triclique coloring algorithm where two of the three cliques have already been assigned colors. In such a case the $3/2$ ratio achieved by our algorithm no longer holds. It would thus be interesting to discover a triclique coloring algorithm that is tailor-made to reduce the approximation ratio guaranteed by the algorithm in [7].

## Acknowledgments

We would like to thank Sampath Kannan for having mentioned the problem to us. We would like to thank anonymous referees for their constructive criticism and insightful comments. In fact, it was one of the referees who suggested that it would be interesting to determine what kind of approximation factors are obtained as a function of the chromatic number of the graph to be colored. This led to the improved bound of theorem 3. Finally, we are indebted to David Gries for his numerous suggestions that greatly improved the presentation of this paper.

## References

[1] A.V.Aho, J.Hopcroft and J.D.Ullman, Compilers: Principles, Techniques and Tools, *Addison-Wesley, Reading, Massachusetts*, 1986.

[2] P.Briggs, K.D.Cooper, Ken Kennedy and Linda Torczon, Coloring Heuristics for register allocation, *Proc. of the SIGPLAN '89 Conference on Programming Language Design and Implementation*, (1989), pp 275-284.

[3] G.J.Chaitin, Register allocation and spilling via graph coloring, *Proc. of the ACM SIGPLAN '82 Symposium on Compiler Construction*, (1982), pp 98-101.

[4] G.Cornuejols, D.Hartvigsen and W.Pulleyblank, Packing subgraphs of a graph, *Operations Research Letters*, 1(1982), pp 139-143.

[5] J.Edmonds, Paths, Trees and Flowers, *Canadian J. Math.*, 17(1965), pp 449-469.

[6] M.R.Garey and D.S.Johnson, Computers and Intractability : A Guide to the theory of NP-completeness, *W.H.Freeman and Co., San Francisco*, (1979).

[7] S.Kannan and Todd Proebsting, Register Allocation in Structured Programs, *Proc. of the sixth ACM-SIAM Symposium on Discrete Algorithms*, (1995), pp 360-368.

[8] J.Naor, M.Naor and A.Schaffer, Fast Parallel Algorithms for Chordal Graphs, $19^{th}$ *ACM STOC, New York*, (1987), pp 355-364.

[9] G.Venkatesan and C.Pandu Rangan, Approximate Triclique Coloring for Register Allocation, *Technical Report*, Department of Computer Science, Indian Institute of Technology, Madras, (1996).