

Approximating Maximum Weight K -Colorable Subgraphs in Chordal Graphs

Venkatesan T. Chakaravarthy

Sambuddha Roy

IBM India Research Lab, New Delhi.
{vechakra,sambuddha}@in.ibm.com

Abstract

We present a 2-approximation algorithm for the problem of finding the maximum weight K -colorable subgraph in a given chordal graph with node weights. The running time of the algorithm is $O(K(n+m))$, where n and m are the number of vertices and edges in the given graph.

Keywords: Approximation algorithms, chordal graphs, coloring, interval graphs.

1 Introduction

The maximum K -colorable subgraph problem is as follows: the input is a graph $G = (V, E)$ and an integer $K \geq 1$ and the goal is to find a maximum cardinality subset $V' \subseteq V$ such that the subgraph induced by V' is K -colorable. In other words, we wish to color a maximum number of nodes such that no two adjacent vertices receive the same color. When $K = 1$, this is the same as the maximum independent set problem, which cannot be approximated in polynomial time within a factor of $n^{1-\epsilon}$, for any fixed $\epsilon > 0$, unless $\text{NP}=\text{P}$ [13]. Clearly, the hardness result holds for our problem. The problem has been studied by restricting to special graph classes. For the case of interval graphs, Yannakakis and Gavril [12] showed that the problem can be solved in linear time. They also presented a polynomial time algorithm for the weighted version of the problem (see also [4]).

We study the problem restricted to chordal graphs. A graph is *chordal*, if every cycle of length at least four has a *chord*, that is, an edge which is not part of the cycle but connects two vertices in the cycle. Chordal graphs are an important subclass of perfect graphs and generalize interval graphs [7]. The problem finds applications in register allocation [9] (see also [8, 3]). We consider the node-weighted version of the problem, wherein each node is associated with a positive real weight and the goal is to maximize the total weight of the chosen subset V' . Frank [6] designed a linear time algorithm for the case of $K = 1$ (i.e., the maximum weight independent set problem). Yannakakis and Gavril [12] showed that the unweighted problem can be solved in polynomial time for any fixed K ; however, they showed that the problem is NP-hard for arbitrary K .

To the best of our knowledge, approximation algorithms have not been considered even for the unweighted version of the above problem, when K is arbitrary. We present an algorithm for the weighted version of problem with an approximation ratio of 2. The running time of the algorithm is $O(K(n+m))$, where n and m are the number of vertices and edges in the graph.

Our algorithm and analysis are inspired from two prior results. First, Farber [5] gave an alternative linear time algorithm for finding a maximum weight independent set in chordal graphs based on the primal-dual approach; this problem is equivalent to the case of $K = 1$ in our problem. Our algorithm can be seen as extension of his algorithm to the case of arbitrary K . Secondly, we observe that the maximum K -colorable subgraph problem (for arbitrary graphs) reduces to the following problem: the input is a graph $G = (V, E)$ and a partition of V into disjoint sets V_1, V_2, \dots, V_ℓ and the goal is to choose a maximum cardinality set $V' \subseteq V$ such that V' is an independent set in G and additionally, V' includes at most one vertex from each set V_j . The reduction is as follows. Let the input to the maximum K -colorable subgraph problem be $G = (V, E)$ on n vertices, given by v_1, v_2, \dots, v_n . Construct a new graph $G' = (V', E')$ on $N = Kn$ vertices. The vertex set V' is given by $V' = \{(v, c) : v \in V \text{ and } 1 \leq c \leq K\}$. Two vertices (v, c) and (v', c') are adjacent in G' , if $c = c'$ and $(v, v') \in E$. The set V' is partitioned into $\ell = n$ many disjoint sets corresponding to the vertices of G so that $V'_i = \{(v_i, c) : 1 \leq c \leq K\}$. It is clear that solutions of the maximum K -colorable instance corresponds to solutions of the new problem instance, and vice versa. In the weighted version of the new problem, each vertex is associated with a positive real weight and the aim is to maximize the total weight of the chosen set V' . Notice that when each V_i contains exactly one vertex, the problem is the same as the maximum weight independent set problem. Motivated by certain scheduling issues, Berman and DasGupta [2] studied the above problem restricted to interval graphs (under the name *job interval selection problem*–JISP) and presented a 2-approximation algorithm for the weighted version (see also [1]). We extend the analysis of Berman and DasGupta to the more general case of chordal graphs and obtain an approximation algorithm with the same ratio of 2. By combining the above reduction and the extended algorithm, we get the claimed 2-approximation algorithm for the maximum K -colorable subgraph problem on chordal graphs. In the paper, we present the combined algorithm and analysis. While our algorithm can also be analyzed using a primal-dual argument similar to Farber’s, we present a more direct analysis of the algorithm.

2 The 2-Approximation Algorithm

In this section, we present a 2-approximation algorithm for the problem of finding the maximum weight K -colorable subgraph for chordal graphs. The algorithm goes via perfect elimination orderings of chordal graphs.

Definition 2.1 (PEO) *Let $G = (V, E)$ be a graph with n vertices. An ordering τ of the vertices v_1, v_2, \dots, v_n is called a perfect elimination ordering, if for any v_j , the set of neighbors of v_j appearing to the right of v_j in τ form a clique, i.e., for any $1 \leq j \leq n$ and $a, b > j$, if $(v_j, v_a) \in E$ and $(v_j, v_b) \in E$, then $(v_a, v_b) \in E$.*

It is well-known that a graph is chordal if and only if it has a perfect elimination ordering [7]. Moreover, given a chordal graph, such an ordering can be found in linear time using the Lex-BFS algorithm [10] or the maximum cardinality search algorithm [11], i.e., the algorithms run in time $O(n + m)$, where n and m are the number of vertices and edges in the given graph.

2.1 Algorithm

We now describe the claimed 2-approximation algorithm. Let $G = (V, E)$ be the input chordal graph on n vertices and let K be the number of allowed colors. For each $v \in V$, let $w(v)$ denote

<p>Input: $G=(V,E)$, weight function $w : V \rightarrow R^+$, number of colors K.</p> <p>Output: A subset $U \subseteq V$ and a proper K-coloring of U.</p> <p>Begin</p> <p> For $c = 1$ to K</p> <p> For $r = 1$ to n</p> <p> $\mathcal{D}(r, c) = \{\langle r, j \rangle : j < c\} \cup \{\langle i, c \rangle : i < r \text{ and } v_i \in \Gamma(v_r)\}$</p> <p> $Z(r, c) = w(v_r) - \sum_{\langle i, j \rangle \in \mathcal{D}(r, c)} M[i, j]$</p> <p> If $(Z(r, c) > 0)$ then $M[r, c] = Z(r, c)$ else $M[r, c] = 0$.</p> <p> For all $i \leq n$ and $j \leq K$, set $\text{Marked}(i, j) = \text{false}$.</p> <p> For $c = K$ to 1</p> <p> For $r = n$ to 1</p> <p> If $M[r, c] > 0$ and $\text{Marked}(r, c) = \text{false}$ then</p> <p> Assign the color c to vertex v_r and include v_r in U.</p> <p> For all $\langle i, j \rangle \in \mathcal{D}(r, c)$, set $\text{Marked}(i, j) = \text{true}$.</p> <p> Return U along with the assignment of colors.</p> <p>End</p>

Figure 1: The 2-Approximation algorithm

the weight of v . The algorithm outputs a subset $U \subseteq V$ and an assignment of one of the K colors to each vertex in U such that no two adjacent vertices in U get the same color. For a vertex $v \in V$, let $\Gamma(v)$ denote the set of neighbors of v , i.e., $\Gamma(v) = \{v' : (v, v') \in E\}$.

Let v_1, v_2, \dots, v_n be a perfect elimination ordering for G , denoted by τ^* . Imagine an $n \times K$ matrix M having nK cells. Each column corresponds to a color and the r^{th} row corresponds to the vertex v_r . By $\langle r, c \rangle$, we denote the cell in the r^{th} row and the c^{th} column. The algorithm works in two phases. In the first phase, we scan the cells in the matrix in a suitable order and compute a value $M[r, c] \geq 0$. (The value $M[r, c]$ can be thought of as the profit of assigning the color c to the vertex v_r , given prior color assignments.) In the second phase, we utilize these values to determine the subset U and also the assignment of colors to the vertices in this subset.

Crucial to the algorithm and the analysis is the notion of domination, defined next. We say that a cell $\langle r, c \rangle$ *dominates* a cell $\langle i, j \rangle$, if one of the following two conditions is satisfied: (i) $i = r$ and $j < c$; (ii) $j = c$, $i < r$ and $v_i \in \Gamma(v_r)$. Let $\mathcal{D}(r, c)$ denote the set of all cells dominated by $\langle r, c \rangle$. Also, define $\widehat{\mathcal{D}}(r, c) = \mathcal{D}(r, c) \cup \{\langle r, c \rangle\}$.

The first phase works as follows. We scan the K columns from left to right and within each column, we process the n cells from top to bottom. Such an ordering of the cells ensures that all the cells dominated by a cell $\langle r, c \rangle$ are processed before the cell $\langle r, c \rangle$ is processed. Compute

$$Z(r, c) = w(v_r) - \sum_{\langle i, j \rangle \in \mathcal{D}(r, c)} M[i, j].$$

If $Z(r, c) > 0$, then set $M[r, c] = Z(r, c)$; otherwise, set $M[r, c] = 0$.

The second phase works as follows. Process all the cells in the reverse order, namely scan the columns from right to left and within each column, scan the cells in the column from bottom to top. Consider a cell $\langle r, c \rangle$. If $M[r, c] > 0$ and the cell is not marked, then assign the color c to v_r and include v_r in U , and mark all the cells dominated by $\langle r, c \rangle$. The algorithm is presented in Figure 1.

2.2 Analysis

In this section, we analyze the algorithm presented in the previous section. It is easy to see that the algorithm outputs a proper K -coloring of the vertices in U , namely no two adjacent vertices in U are assigned the same color and a vertex is assigned at most one color. We next discuss the approximation ratio achieved by the algorithm and then consider its time complexity.

Our goal is to show that the algorithm has an approximation ratio of 2. For a set of cells D , let $Val(D)$ denote the sum of the values in the cells in D . Let VAL denote the total value of all the cells in the matrix after the first phase, i.e., $VAL = \sum_{i=1}^n \sum_{j=1}^K M[i, j]$. For a subset of vertices $V' \subseteq V$, let $w(V')$ denote sum of weights of the vertices contained in V' .

Let X any subset of V such that the subgraph induced by X is K -colorable. We shall prove two claims regarding the given solution X and the solution U computed by the algorithm: (i) $2 \times VAL \geq w(X)$; (ii) $w(U) \geq VAL$. It follows that $2w(U) \geq w(X)$. Since X was an arbitrary feasible solution, we have that U is a 2-approximate solution.

Lemma 2.2 $2 \times VAL \geq w(X)$.

Proof: Let $X = \{x_1, x_2, \dots, x_\ell\}$ be the given solution and c_1, c_2, \dots, c_ℓ be a K -coloring of the vertices in X . Denote this coloring by C . We can visualize the given solution X and its coloring C by placing ℓ pebbles on the matrix M . Let r_1, r_2, \dots, r_ℓ denote the rows corresponding to the vertices x_1, x_2, \dots, x_ℓ , respectively. For $1 \leq i \leq \ell$, place a pebble p_i in the cell $M[r_i, c_i]$. Note that each of the rows r_1, r_2, \dots, r_ℓ contains exactly one pebbles and the others rows do not contain any pebble. Thus, any row in M contains at most one pebble. On the other hand, for any column c in M , if there are pebbles on $M[i, c]$ and $M[j, c]$ then v_i and v_j are non-adjacent in G .

For $1 \leq d \leq K$, associate with the pebble p_d the cells contained in $\widehat{\mathcal{D}}(r_d, c_d)$. We make two claims: (a) any cell $\langle r, c \rangle$ is associated with at most two pebbles; (b) for any pebble p_d , $Val(\widehat{\mathcal{D}}(r_d, c_d)) \geq w(x_d)$. The lemma follows from the two claims. We proceed to prove the two claims.

Let us prove claim (a). Consider any cell $\langle r, c \rangle$. Clearly, the cell can be associated with only pebbles placed in row r or column c . It is easy to see that the cell $\langle r, c \rangle$ is associated with at most one pebble placed in the row r , since a row cannot contain two pebbles. We shall prove that the cell can be associated with at most one pebble from the column c . We consider two cases: Suppose some pebble p_d is placed on the cell $\langle r, c \rangle$ itself. Now if $\langle r, c \rangle$ is associated with some other pebble p_i placed in the same column c , it means that the corresponding vertices v_r and x_i are adjacent; this would contradict the fact that C is a proper coloring. Thus the cell $\langle r, c \rangle$ cannot be associated with any other pebble placed in column c . Now suppose, no pebble is placed on the cell $\langle r, c \rangle$. By contradiction, assume that the cell $\langle r, c \rangle$ is associated with two pebbles p_i and p_j placed in column c . Then, $i > r$ and $j > r$ and moreover, $v_r \in \Gamma(v_i)$ and $v_r \in \Gamma(v_j)$. By the perfect elimination ordering property of τ^* , it follows that v_i and v_j are adjacent. This contradicts the hypothesis that C is a proper coloring of vertices in X .

We now turn to claim (b). Note that for any $\langle r, c \rangle$, $Val(\widehat{\mathcal{D}}(r, c)) = Val(\mathcal{D}(r, c)) + M[r, c]$. Consider any pebble p_d . The construction of the algorithm implies the following two facts: (i) if $Z(r_d, c_d) > 0$, then $Val(\widehat{\mathcal{D}}(r_d, c_d)) = Val(\mathcal{D}(r_d, c_d)) + M[r_d, c_d] = w(x_d)$; (ii) if $Z(r_d, c_d) \leq 0$, then $Val(\widehat{\mathcal{D}}(r_d, c_d)) \geq Val(\mathcal{D}(r_d, c_d)) \geq w(x_d)$. It follows that the claim is true in both the cases. \square

Lemma 2.3 $w(U) \geq VAL$.

Proof: The proof has a flavor similar to that of Lemma 2.2. Let the solution computed by the algorithm be $U = \{u_1, u_2, \dots, u_s\}$. Let c_1, c_2, \dots, c_s be the colors assigned by the algorithm to the vertices u_1, u_2, \dots, u_s , respectively. Let r_1, r_2, \dots, r_s be the rows corresponding to the vertices u_1, u_2, \dots, u_s , respectively. We can imagine the solution U and its coloring by placing s pebbles on the matrix. For each $1 \leq d \leq s$, place a pebble p_d on the cell $\langle r_d, c_d \rangle$. For each pebble p_d , associate the cells in $\widehat{\mathcal{D}}(r_d, c_d)$ with the pebble p_d . We make two claims: (a) for any pebble p_d , $w(u_d) \geq \text{Val}(\widehat{\mathcal{D}}(r_d, c_d))$; (b) for any cell $\langle r, c \rangle$, if $M[r, c] > 0$ then the cell is associated with at least one pebble. Notice that in computing VAL, we can ignore the cells having value=0. Thus, the lemma follows from the two claims.

To prove claim (a), consider any pebble p_d . Since the algorithm assigned the color c_d to the vertex v_d , we have $Z(r_d, c_d) > 0$. This means that $w(u_d) = \text{Val}(\widehat{\mathcal{D}}(r_d, c_d))$.

We now prove claim (b). Take any cell $\langle r, c \rangle$ having value $M[r, c] > 0$. Consider the processing of this cell in the second phase. If the cell is not marked, then we assign the color c to r . This means that a pebble is placed in the cell $\langle r, c \rangle$, with which the cell gets associated. On the other hand, if the cell is marked, then it is clearly associated with some pebble (for instance, the pebble that caused the cell to get marked for the first time). \square

Time Complexity: In the first phase of the algorithm, in order to compute the quantity $Z(r, c)$, we need to look-up the values of the cells contained in $\mathcal{D}(r, c)$. Size of $\mathcal{D}(r, c)$ is bounded as $|\mathcal{D}(r, c)| \leq K + |\Gamma(v_r)|$. Via a more careful implementation, the factor K in the number of look-ups can be eliminated. With each cell $\langle r, c \rangle$, we maintain the quantity $C(r, c)$ given by $C(r, c) = \sum_{j=1}^{c-1} M[r, j]$. Then, for any cell $\langle r, c \rangle$, $Z(r, c)$ can be computed as

$$Z(r, c) = w(v_r) - C(r, c - 1) - \sum_{i: i < r \text{ and } v_i \in \Gamma(v_r)} M[i, c].$$

This way, the number of lookups for cell $\langle r, c \rangle$ can be reduced to $1 + |\Gamma(v_r)|$. Clearly, the dominating factor is $|\Gamma(v_r)|$. Summing up this over all the cells yields:

$$\sum_{j=1}^K \sum_{i=1}^n |\Gamma(v_i)| = K \left[\sum_{i=1}^n |\Gamma(v_i)| \right] = 2Km,$$

where $m = |E|$ is the number of edges in the graph. Processing the Kn cells of the matrix M takes time $O(Kn)$. Thus, the total time taken by the first phase is $O(K(n + m))$. A similar argument shows that the second phase can also be implemented in time $O(K(n + m))$. So, the overall time complexity of the algorithm is $O(K(n + m))$. \square

We have proved the following theorem.

Theorem 2.4 *There exists a 2-approximation algorithm for the maximum weight K -colorable subgraph problem for chordal graphs. The running time of the algorithm is $O(K(n + m))$, where n and m are the number of vertices and edges in the given graph and K is the number of colors allowed.*

As noted in the introduction, when $K = 1$, the problem is the same as the problem of finding maximum weight independent set. Notice that in Lemma 2.2, when $K = 1$, a cell can be associated with at most one pebble of the given solution X . This means that the lemma can be refined as $\text{VAL} \geq w(X)$. As a result, we see that the algorithm produces an optimal solution. Thus, we get a linear time optimal algorithm for the problem of finding maximum weight independent set in chordal graphs. The resulting algorithm is equivalent to Farber's algorithm [5] for this problem.

References

- [1] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Noar, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.
- [2] P. Berman and B. DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4(3):307–323, 2000.
- [3] P. Brisk, F. Dabiri, J. Macbeth, and M. Sarrafzadeh. Polynomial time graph coloring register allocation. In *14th International Workshop on Logic and Synthesis*, 2005.
- [4] M. Carlisle and E. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3):225–235, 1995.
- [5] M. Farber. *Applications of Linear Programming Duality to Problems Involving Independence and Domination*. PhD thesis, Rutgers University, 1982.
- [6] A. Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *5th British Combinatorial Conference*, 1976.
- [7] M. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.
- [8] S. Hack. Interference graphs of programs in SSA form. Technical report, Universitat Karlsruhe, 2005.
- [9] F. Pereira and J. Palsberg. Register allocation via coloring of chordal graphs. In *3rd Asian Symposium on Programming Languages and Systems*, 2005.
- [10] D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.
- [11] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.
- [12] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.
- [13] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, 2006.