Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 381 (2007) 124–133
Theoretical Computer Science
www.elsevier.com/locate/tcs

Some approximation algorithms for the clique partition problem in weighted interval graphs

Mingxia Chen[a], Jianbo Li[b], Jianping Li[c,*], Weidong Li[c], Lusheng Wang[d]

[a] *Department of Science and Technology, Yunnan University, Kunming 650091, PR China*
[b] *School of Management and Economics, Kunming University of Science and Technology, Kunming 650090, PR China*
[c] *Department of Mathematics, Yunnan University, Kunming 650091, PR China*
[d] *Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong, PR China*

Received 28 August 2006; received in revised form 19 March 2007; accepted 13 April 2007

Communicated by D.-Z. Du

Abstract

Interval graphs play important roles in analysis of DNA chains in Benzer [S. Benzer, On the topology of the genetic fine structure, Proceedings of the National Academy of Sciences of the United States of America 45 (1959) 1607–1620], restriction maps of DNA in Waterman and Griggs [M.S. Waterman, J.R. Griggs, Interval graphs and maps of DNA, Bulletin of Mathematical Biology 48 (2) (1986) 189–195] and other related areas. In this paper, we study a new combinatorial optimization problem, named the minimum clique partition problem with constrained bounds, in weighted interval graphs. For a weighted interval graph G and a bound B, partition the weighted intervals of this graph G into the smallest number of cliques, such that each clique, consisting of some intervals whose intersection on a real line is not empty, has its weight not beyond B. We obtain the following results: (1) this problem is *NP*-hard in a strong sense, and it cannot be approximated within a factor $\frac{3}{2} - \varepsilon$ in polynomial time for any $\varepsilon > 0$; (2) we design three approximation algorithms with different constant factors for this problem; (3) for the version where all intervals have the same weights, we design an optimal algorithm to solve the problem in linear time.

Keywords: Weighted interval graphs; Cliques; Approximation algorithms

1. Introduction

An undirected graph $G = (V, E)$ is called an interval graph if, for each vertex $v \in V$, v can be associated to an open interval I_v on the real line, such that any pair of distinct vertices $u, v \in V$ are connected by an edge in E if and only if two related open intervals I_u and I_v satisfy $I_u \cap I_v \neq \emptyset$. Then the family $\{I_v\}_{v \in V}$ is called an interval representation of this interval graph G. For convenience, we also treat these intervals I_v as the vertices of this interval graph G. A graph G is called a weighted interval graph, if there is a function $w : \{I_v\}_{v \in V} \to R^+$, i.e., each vertex v (equivalently interval I_v) has a weight $w(v)$ (or $w(I_v)$).

* Corresponding author. Tel.: +86 871 5032704; fax: +86 871 5033701.
E-mail addresses: mxchen@ynu.edu.cn (M. Chen), lijianbo@public.km.yn.cn (J. Li), jianping@ynu.edu.cn (J. Li), cswangl@cityu.edu.hk (L. Wang).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2007.04.030

Interval graphs have many applications in molecular biology, scheduling of tasks executed, timing of traffic lights, and so on. For example, Benzer [1] invented interval graphs to study the analysis of DNA chains, i.e., the linearity of the chain for higher organisms, and interval graph aids in locating genes along the DNA sequence; Waterman and Griggs [11] utilized interval graphs to study an important representation of DNA called restriction maps; Papadimitriou and Yannakakis [7] utilized interval graphs to study the scheduling interval-order tasks; Roberts [8] utilized interval graphs to study the problem of timing of traffic lights to optimize some criteria such as average waiting time. Other applications can be found in [3,5,9,12].

Since such an interval graph is a special type of perfect graph [5], many research papers have studied some combinatorial optimization problems in interval graphs, such as computing the maximum coloring number, a maximum independent set and a maximum clique in such an interval graph. In this paper, we study the clique partition problem with constrained bounds in weighted interval graphs, where each clique consists of some intervals in G whose intersection on a real line is not empty.

Our problem, named the *Minimum Clique Partition Problem with Constrained bounds in Weighted Interval Graphs* (MCPBI), is stated in the following form:

INSTANCE: A weighted interval graph $G = (V, E; w)$ with intervals $I_1, \ldots, I_n$, having weights $w_1, \ldots, w_n$, and a bound B;

QUESTION: Find a partition of these n intervals into the smallest number of cliques such that each clique has its weight not beyond B.

Kaplan and Shamir [6] studied a problem related to ours. They studied some pathwidth, bandwidth and completion problems for proper interval graphs with small cliques. Bodlaender and Jansen [2] studied the restrictions of graph partition problems in several classes of graphs without weights. They obtained some results: the problem to partition a cograph into bounded cliques (independent sets, respectively) remains *NP*-hard, and the problem to partition an unweighted interval graph into bounded cliques is solvable in linear time by using an algorithm due to Papadimitriou and Yannakakis [7]. However, when we study the clique partition problem with constrained bound B in weighted interval graphs, it will become *NP*-hard to compute such a minimum number of cliques, each clique having a weight not beyond B. As far as we know, there are no approximation algorithms to this new combinatorial optimization problem, and we will design three approximation algorithms with constant factors to the problem, and we will also redesign a new linear time algorithm to solve the problem in the version where all intervals have the same weight. Our new linear time algorithm is completely different from the one due to Bodlaender and Jansen in [2], and the technique they used in [2] heavily depends on a result of scheduling interval-order tasks in [7], but our linear time algorithm in Section 4 will only depend on the new sorting technique defined in Section 3 and the GREEDY method.

This paper is divided into the following sections. In Section 2, we prove that the MCPBI problem remains *NP*-hard in a strong sense, by transformation from the 3-PARTITION problem, and that it cannot be approximated within a factor $\frac{3}{2} - \varepsilon$ for any $\varepsilon > 0$, by transformation from the PARTITION problem. In Section 3, three approximation algorithms with different constant factors are designed for this problem. For the case where each interval has the same weight, we redesign an optimal algorithm to solve the problem in linear time in Section 4. We give conclusions and remarks in the last section.

2. Hardness of the MCPBI problem

In this section, we study the hardness of the MCPBI problem, and then we prove that the MCPBI problem cannot be approximated within a factor $\frac{3}{2} - \varepsilon$ in polynomial time for any $\varepsilon > 0$. The *NP*-hardness of the MCPBI problem is proved by transformation from the 3-PARTITION problem.

Theorem 1. *The* MCPBI *problem is NP-hard in a strong sense.*

Proof. We prove the *NP*-hardness of the MCPBI problem by transforming any instance of the 3-PARTITION problem to an instance of the MCPBI problem.

Consider an instance $\mathcal{I}$ of the 3-PARTITION problem: Given the set $S = \{a_1, a_2, \ldots, a_{3k}\}$ of $3k$ integers to satisfy $\frac{B}{4} < a_j < \frac{B}{2}$ for each $1 \leq j \leq 3k$ and $\Sigma_{j=1}^{3k} a_j = kB$, ask whether S can be partitioned into k subsets $S_1, S_2, \ldots, S_k$ such that, for each $i = 1, 2, \ldots, k$, S_i contains exactly three elements of S and $\Sigma_{a \in S_i} a = B$.

We construct a reduction τ from $\mathcal{I}$ of the 3-PARTITION problem to an instance $\tau(\mathcal{I})$ of the MCPBI problem: a weighted interval graph G with intervals $I_1, I_2, \ldots, I_{3k}$, for $j = 1, 2, \ldots, 3k$, each interval I_j having its left endpoint $o(I_j) = j - 1$, right endpoint $d(I_j) = 3k$ on a real line and possessing its weight a_j, and the bound $B = \frac{\Sigma_{j=1}^{3k} a_j}{k}$.

Now, we obtain the following claim.

Claim 1. *There exists a feasible solution to an instance $\mathcal{I}$ of the* 3-PARTITION *problem if and only if the instance $\tau(\mathcal{I})$ of the* MCPBI *problem has its optimal solution with value k.*

In fact, for any feasible solution of an instance $\mathcal{I}$ of the 3-PARTITION problem, the set S is partitioned into k subsets $S_1, S_2, \ldots, S_k$ such that, for $i = 1, 2, \ldots, k$, S_i contains exactly three elements of S and $\sum_{a \in S_i} a = B$. Then we can construct a feasible partition to the instance $\tau(\mathcal{I})$ of the MCPBI problem in the following way: for each $S_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$, select the clique $C_i = \{I_{i_1}, I_{i_2}, I_{i_3}\}$, and then we obtain the partition of these $3k$ intervals into k cliques, each clique having exactly weight B.

Conversely, if the instance $\tau(\mathcal{I})$ of the MCPBI problem has an optimal clique partition $\{C_1, C_2, \ldots, C_k\}$ with the smallest integer k, having $\sum_{a \in C_i} a \leq B$ for each $i = 1, 2, \ldots, k$. By the facts $\Sigma_{j=1}^{3k} a_j = kB$ and $\frac{B}{4} < a_j < \frac{B}{2}$ for each $1 \leq j \leq 3k$, we obtain $\sum_{a \in C_j} a = B$ for each $1 \leq j \leq k$ and then each clique C_j contains exactly three elements from S, i.e., $S_j = \{a_{j_1}, a_{j_2}, a_{j_3}\}$ and $\sum_{a \in C_j} a = B$ for each $1 \leq j \leq k$. So the instance $\mathcal{I}$ of the 3-PARTITION problem has the partition $S_1, S_2, \ldots, S_k$.

Hence, the *NP*-hardness in a strong sense of the MCPBI problem follows the fact that the 3-PARTITION problem is one of the earliest known *NP*-hard problems in a strong sense. This reaches the conclusion of the theorem. ∎

We know that the MCPBI problem is *NP*-hard in a strong sense from Theorem 1; moreover, we obtain the following strong result, which is proved by transformation from the PARTITION problem.

Theorem 2. *For any $\varepsilon > 0$, there is no an approximation algorithm having a factor $\frac{3}{2} - \varepsilon$ for the* MCPBI *problem, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Suppose that there is an approximation algorithm $\mathcal{A}$ having a factor $\frac{3}{2} - \varepsilon$ for the MCPBI problem, then we can show how to solve the PARTITION problem in polynomial times, i.e., deciding if there is a way to partition n nonnegative numbers $a_1, a_2, \ldots, a_n$ into two sets, each adding up to $\frac{1}{2}\Sigma_{i=1}^{n} a_i$.

For an instance $\mathcal{I}$ of the PARTITION problem consisting of n nonnegative numbers $a_1, a_2, \ldots, a_n$, we can construct a reduction τ from $\mathcal{I}$ of the PARTITION problem to an instance $\tau(\mathcal{I})$ of the MCPBI problem: an interval graph G with intervals $I_1, I_2, \ldots, I_n$, for $j = 1, 2, \ldots, n$, each interval I_j having its left endpoint $o(I_j) = j - 1$, right endpoint $d(I_j) = n$ on a real line and possessing its weight a_j, and the bound $B = \frac{1}{2}\Sigma_{i=1}^{n} a_i$.

Clearly, the answer to the PARTITION problem is 'yes' if and only if the MCPBI problem has exactly two cliques of weight $\frac{1}{2}\Sigma_{i=1}^{n} a_i$.

When we use the approximation algorithm $\mathcal{A}$ on the instance $\tau(\mathcal{I})$, it produces an output m to satisfy $m \leq (\frac{3}{2} - \varepsilon)\mathrm{OPT}$, where OPT is the optimal value to the instance $\tau(\mathcal{I})$. If $\mathrm{OPT} = 2$, then the preceding formula implies $m = 2$, implying that the PARTITION problem has a feasible solution; if $\mathrm{OPT} \geq 3$, then we get $m \geq \mathrm{OPT} \geq 3$, implying that the PARTITION problem has no feasible solution. So the approximation algorithm $\mathcal{A}$ solves the PARTITION problem in polynomial time. But the PARTITION problem remains *NP*-hard [4], a contradiction.

Hence, the theorem holds. ∎

3. Some approximation algorithms for the MCPBI problem

Since the MCPBI problem is *NP*-hard and there is no polynomial-time algorithm to optimally solve it, we will design some approximation algorithms for this problem.

We have known that such an interval graph is a special version of perfect graph, there are some polynomial-time algorithms to compute the maximum coloring number, a maximum cardinality independent set and a maximum clique in such an interval graph [5]. To simply state our approximation algorithms, we will utilize an optimal algorithm due to Tarjan [10] to compute a maximum cardinality independent set in such an interval graph, by utilizing some technique of minimum-cost flow with value 1, and this optimal algorithm runs in time $\mathcal{O}(n)$. For convenience, we denote such an algorithm as MAX-IND.SET.

Before we design approximation algorithms for the MCPBI problem, we give the rules of the sequel sorting. For a weighted interval graph $G = (V, E; w)$ with intervals $I_1, I_2, \ldots, I_n$, denote two integers $o(i)$ and $d(i)$ respectively as the left endpoint and the right endpoint of the interval I_i for each $1 \le i \le n$, located on a real line from left to right, each interval I_i having its weight w_i.

We denote a linear order '$\le$' on G: for any two intervals I_i and I_j of G, denote $I_i \le I_j$ if and only if (1) either $d(i) < d(j)$, or (2) $d(i) = d(j)$ and $o(i) < o(j)$, or (3) $d(i) = d(j)$, $o(i) = o(j)$ and $w_i \le w_j$. We also denote $I_j \ge I_i$ if $I_i \le I_j$ for convenience. In particular, we denote $I_i < I_j$ if $I_i \le I_j$ and $I_i \ne I_j$. Generally, it takes $\mathcal{O}(n \log n)$ steps to sort these n intervals, heavily depending on this linear order '$\le$' according to the rule (3). When either all intervals have the same weights or we do not take care of the weights of these n intervals, the preceding order '$\le$' is also linear, but the sorting time in this case runs in time $\mathcal{O}(n)$, only depending on the choices (1) and (2). We will changeably utilize these two linear orders in the sequel, and their different running times depend on the rule choices. For any subgraph G' of the interval graph G, it is known that the partial order '$\le$' on G' is a linear order [9], too.

Before we state our algorithm Clique-Partition I, we describe a GREEDY method to obtain r disjoint cliques from the interval graph G, depending on a maximum cardinality independent set in G.

Algorithm GREEDY

INPUT: an interval graph $G = (V, E)$ with intervals $I_1, \ldots, I_n$, and a maximum cardinality independent set $\mathcal{I} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$ in G, where $I_{i_1} < \cdots < I_{i_r}$;

OUTPUT: r disjoint cliques $C_1, C_2, \ldots, C_r$, where C_t contains the interval I_{i_t}.

Begin

Step 1 Put $G' = G$;

Step 2 From left to right on the real line, for $t = 1, 2, \ldots, r$, choose a *maximal* clique C_t consisting of I_{i_t} and all other intervals from G', and then let $G' := G' - C_t$, until $G' = \emptyset$.

End of GREEDY.

By utilizing the algorithm GREEDY and the fact that $\mathcal{I} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$ is a maximum cardinality independent set in G, where $I_{i_1} < I_{i_2} < \cdots < I_{i_r}$, we obtain a partition $\{C_1, C_2, \ldots, C_r\}$ of these n intervals $I_1, \ldots, I_n$, where C_t contains the interval I_{i_t} for each $1 \le t \le r$.

Now, we design the first approximation algorithm for the MCPBI problem:

Algorithm Clique-Partition I

INPUT: a weighted interval graph $G = (V, E; w)$ with intervals $I_1, \ldots, I_n$, having weights $w_1, \ldots, w_n$, and a bound B;

OUTPUT: m disjoint cliques consisting of these n intervals, each clique having its weight not beyond B.

Begin

Step 1 Use the algorithm MAX-IND.SET to compute a maximum cardinality independent set $\mathcal{I} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$ in G, where $I_{i_1} < I_{i_2} < \cdots < I_{i_r}$;

Step 2 Use the GREEDY algorithm to obtain r disjoint maximal cliques $C_1, C_2, \ldots, C_r$, where each maximal clique C_t contains the interval I_{i_t} for $1 \le t \le r$ such that $\{C_1, C_2, \ldots, C_r\}$ is a partition of these n intervals;

Step 3 For each clique $C_j = \{I_{j_1}, I_{j_2}, \ldots, I_{j_{m_j}}\}$, where $I_{j_1} \le I_{j_2} \le \cdots \le I_{j_{m_j}}$ only depend on the choices (1) and (2), choose some suitable cliques in the sequels repeated: (a) C_j^1 first contains the interval I_{j_1}; (b) for $2 \le t \le m_j$ and when there are k cliques $C_j^1, \ldots, C_j^k$ containing the intervals $I_{j_1}, I_{j_2}, \ldots, I_{j_{t-1}}$, add the current interval I_{j_t} in a clique $C_j^{k'}$ $(1 \le k' \le k)$ if the total weight sum of I_{j_t} and the intervals in $C_j^{k'}$ is not greater than B, otherwise choose a new clique, named C_j^{k+1}, to contain the current interval I_{j_t};

Step 4 Output all cliques obtained from Step 3.

End of Clique-Partition I.

Theorem 3. *The algorithm Clique-Partition* I *is an approximation algorithm with factor* 3 *for the* MCPBI *problem, its complexity is* $\mathcal{O}(n)$.

Proof. For each $1 \le j \le r$, let Out_j be the set of cliques produced by Step 3 in the algorithm Clique-Partition I on each clique C_j and denote $\mathrm{OUT} = \bigcup_{j=1}^{r} \mathrm{Out}_j$. Then, such $|\mathrm{Out}_j|$ cliques must contain at least $|\mathrm{Out}_j| - 1$ cliques whose weights are greater than $\frac{B}{2}$, otherwise $|\mathrm{Out}_j|$ will be decreased. Thus we have

$$\begin{aligned}\sum_{i=1}^{m} w_i &> \frac{B}{2}(|\mathrm{Out}_1| - 1) + \frac{B}{2}(|\mathrm{Out}_2| - 1) + \cdots + \frac{B}{2}(|\mathrm{Out}_r| - 1)\\ &= \frac{B}{2}(|\mathrm{Out}_1| + |\mathrm{Out}_2| + \cdots + |\mathrm{Out}_r| - r)\\ &= \frac{B}{2}(|\mathrm{OUT}| - r)\end{aligned}$$

implying

$$|\mathrm{OUT}| < \frac{2\sum_{i=1}^{m} w_i}{B} + r \le 2|\mathrm{OPT}| + |\mathrm{OPT}| = 3|\mathrm{OPT}|$$

where the second inequality depends on the two facts that the optimal solution has two lower bounds $\frac{\sum_{i=1}^{m} w_i}{B}$ and r, i.e., $|\mathrm{OPT}| \ge \frac{\sum_{i=1}^{m} w_i}{B}$ and $|\mathrm{OPT}| \ge r$. So the algorithm Clique-Partition I has a factor 3.

Now, we analyze the complexity of the algorithm Clique-Partition I: (1) Step 1 needs $\mathcal{O}(n)$ steps to compute a maximum cardinality independent set in such an interval graph in Tarjan [10]; (2) by using the GREEDY algorithm, Step 2 needs $\mathcal{O}(n)$ steps to find such a clique partition of these n intervals; (3) since each interval must be chosen in a clique and the index sorting in this case needs $\mathcal{O}(n)$ steps only depending on the rules (1) and (2), by treating a constant number of steps to put each interval in some clique, the steps in Step 3 totally need time in $\mathcal{O}(m_1) + \mathcal{O}(m_2) + \cdots + \mathcal{O}(m_r)$, i.e., at most $\mathcal{O}(n)$. Hence, the whole algorithm needs a running time in $\mathcal{O}(n)$.

This establishes the conclusion of the theorem. ∎

When we modify Step 3 in the algorithm Clique-Partition I, i.e., we sort all intervals in each clique C_j according to the similar preceding rules (1)–(3) before we choose suitable cliques, where $1 \le j \le r$, we obtain a better algorithm with a factor $\frac{5}{2}$.

Algorithm Clique-Partition II

INPUT: a weighted interval graph $G = (V, E; w)$ with intervals $I_1, \ldots, I_n$, having weights $w_1, \ldots, w_n$, and a bound B;

OUTPUT: m disjoint cliques consisting of these n intervals, each clique having its weight not beyond B.

Begin

Step 1 Use the algorithm MAX-IND.SET to compute a maximum cardinality independent set $\mathcal{I} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$ in G, for convenience, $I_{i_1} < I_{i_2} < \cdots < I_{i_r}$;

Step 2 From left to right on the real line, utilize the GREEDY algorithm to obtain r disjoint cliques $C_1, C_2, \ldots, C_r$, where C_t contains the interval I_{i_t} for each $1 \le t \le r$, such that $\{C_1, C_2, \ldots, C_r\}$ is a partition of these n intervals;

Step 3 For each clique $C_j = \{I_{j_1}, I_{j_2}, \ldots, I_{j_{m_j}}\}$, where $1 \le j \le r$ and $I_{i_j} \in C_j$, sort these m_j intervals in C_j into decrease order according to their weights, i.e., $w(I_{j_1}) \ge w(I_{j_2}) \ge \cdots \ge w(I_{j_{m_j}})$ for convenience, then choose some suitable cliques in the sequel repeated: (a) C_j^1 first contains the interval I_{j_1}; (b) for $2 \le t \le m_j$ and when there are k cliques $C_j^1, \ldots, C_j^k$ containing the intervals $I_{j_1}, I_{j_2}, \ldots, I_{j_{t-1}}$, add the current interval I_{j_t} in a clique $C_j^{k'}$ $(1 \le k' \le k)$ if the total weight sum of I_{j_t} and the intervals in $C_j^{k'}$ is not greater than B, otherwise choose a new clique, named C_j^{k+1}, to contain the current interval I_{j_t};

Step 4 Output all cliques obtained from Step 3.

End of Clique-Partition II.

Theorem 4. *The algorithm Clique-Partition* II *is an approximation algorithm with a factor* $\frac{5}{2}$ *for the* MCPBI *problem, its complexity is* $\mathcal{O}(n \log n)$.

Proof. For each $1 \le j \le r$, let Out_j be the set of cliques produced by Step 3 of the algorithm Clique-Partition II on each clique C_j, and denote $\mathrm{OUT} = \bigcup_{j=1}^{r} \mathrm{Out}_j$.

Then, we obtain the following result which plays an important role to ensure the factor $\frac{5}{2}$ for the algorithm Clique-Partition II.

Claim 2. *For each set* Out_j*, where* $1 \le j \le r$*, either these* $|\mathrm{Out}_j|$ *cliques in* Out_j *must contain at least* $|\mathrm{Out}_j| - 1$ *cliques each of which has weight greater than* $\frac{2B}{3}$ *or all intervals in the clique* C_j *must be covered by at least* $|\mathrm{Out}_j|$ *cliques from the set of an optimal solution.*

In fact, suppose that there exists some $1 \le j \le r$ *such that* Out_j *contains at least two cliques* C_j^1 *and* C_j^2 *whose weights are both less than or equal to* $\frac{2B}{3}$*, without loss of generality,* C_j^1 *and* C_j^2 *are the last two cliques whose weights are both less than or equal to* $\frac{2B}{3}$*, and* C_j^1 *is produced before* C_j^2*.*

When the clique C_j^2 *contains at least two intervals, then at least one interval has its weight less than or equal to* $\frac{B}{3}$*. According to our algorithm, this light interval must be added into some clique before* C_j^2*, for example, is added into the clique* C_j^1*, a contradiction. When the clique* C_j^2 *contains only one interval, by utilizing our algorithm, each interval in the clique* C_j *has its weight more than* $\frac{B}{3}$*, and for such an instance, at most two intervals in the clique* C_j *can be covered by any clique from the set of the optimal solution; by utilizing our algorithm, this shows that at least* $|\mathrm{Out}_j|$ *cliques from the set of an optimal solution are needed to cover all intervals in the clique* C_j*.* ∎

By Claim 2, all intervals from the clique C_j must be covered by at least $\frac{2}{3}(|\mathrm{Out}_j| - 1)$ cliques, in either case, from an optimal partition OPT. Since the r cliques $C_1, C_2, \ldots, C_r$ produced in the processes repeated by the algorithm GREEDY are maximal in the interval graphs G's, so $|\mathrm{Out}_j|$ cliques from the optimal solution that cover different C_j must be different. Thus we conclude

$$\begin{aligned} |\mathrm{OPT}| &> \frac{2}{3}(|\mathrm{Out}_1| - 1) + \frac{2}{3}(|\mathrm{Out}_2| - 1) + \cdots + \frac{2}{3}(|\mathrm{Out}_r| - 1) \\ &= \frac{2}{3}(|\mathrm{Out}_1| + |\mathrm{Out}_2| + \cdots + |\mathrm{Out}_r| - r) \\ &= \frac{2}{3}(|\mathrm{OUT}| - r) \end{aligned}$$

implying

$$|\mathrm{OUT}| < \frac{3}{2}|\mathrm{OPT}| + r \le \frac{3}{2}|\mathrm{OPT}| + |\mathrm{OPT}| = \frac{5}{2}|\mathrm{OPT}|$$

where the second inequality depends on the facts that r is the cardinality of the independent set $\mathcal{I} = \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$ in G and the optimal solution has a lower bound r, i.e., $|\mathrm{OPT}| \ge r$. So the algorithm Clique-Partition II has a factor $\frac{5}{2}$.

The complexity of the algorithm Clique-Partition II comes from the following analysis: (1) Step 1 needs $\mathcal{O}(n)$ steps to compute a maximum cardinality independent set in such an interval graph in Tarjan [10]; (2) by using the GREEDY algorithm, Step 2 needs $\mathcal{O}(n)$ steps to find such a clique partition of these n intervals; (3) since each interval must be chosen in a clique and the index sorting in this case only depends on the time $\mathcal{O}(m_j \log\ m_j)$ according to their weights in the clique C_j, by treating a constant number of steps to put each interval in some clique, the steps in Step 3 need a total time in $\mathcal{O}(m_1 \log\ m_1) + \mathcal{O}(m_2 \log\ m_2) + \cdots + \mathcal{O}(m_r \log\ m_r)$, i.e., at most $\mathcal{O}(n \log\ n)$. Hence, the whole algorithm needs a running time in $\mathcal{O}(n \log n)$.

This establishes the conclusion of the theorem. ∎

Finally, we design the third approximation algorithm for the MCPBI problem that has a better factor than before, and it has a running time $\mathcal{O}(n^2)$. For convenience, the interval is called a *heavy* interval if this interval has its weight greater than $\frac{B}{2}$, otherwise the interval is called a *light* interval.

The ideas to design a new algorithm are to choose each heavy interval in a clique and then choose the light intervals as the preceding algorithms; in particular when Step 3 in the following algorithm executes once, we choose a smallest interval according to the rules (1) and (2) which is also a light interval. Here an interval $I_{\min}$ is called a *smallest* one if $I_{\min} \le I$ holds for each light interval I in the current interval graph G which only contains light intervals, depending on the linear order '$\le$' according to the rules (1) and (2) [5]. When we sort all intervals in the original interval graph

G on the linear order '$\leq$' according to the rules (1) and (2), it runs in time $\mathcal{O}(n)$. Similarly, when we sort all light intervals in the original interval graph G on the linear order '$\leq$' according to the rules (1) and (3), it runs in time $\mathcal{O}(n \log\ n)$. Then we can choose a smallest interval $I_{\min}$ by a constant time from the current interval graph (sorted) once, depending on the preceding index sorting.

Our third approximation algorithm for the MCPBI problem is designed:

Algorithm Clique-Partition III

INPUT: a weighted interval graph $G = (V, E; w)$ with intervals $I_1, \ldots, I_n$, having weights $w_1, \ldots, w_n$, and a bound B;

OUTPUT: m disjoint cliques consisting of these n intervals, each clique having its weight not beyond B.

Begin

Step 1 For each heavy interval I_i, let a clique C_i only contain such a heavy interval I_i; and after removing all heavy intervals from G, the current interval graph G only contains light intervals;
(/*We only consider all light intervals in the current interval graph G below/*)

Step 2 Sort all light intervals in the current interval graph G according to the preceding rules (1) and (2), without loss of generality, the light intervals in G are sorted as $I_1, \ldots, I_{n'}$ depending on the rules (1) and (2);

Step 3 According to the linear order '$\leq$' on G depending on the rules (1) and (2), choose a smallest element, $I_{\min}$, in the current graph G as the alive interval; and then find the *maximal* clique C from the current graph G to contain such an alive interval $I_{\min}$, without loss of generality, all intervals in such a maximal clique C are sorted as $I_i, I_{i+1}, \ldots, I_j$, where $I_i = I_{\min}$;

Step 4 Use a similar method to Step 3 in the algorithm Clique-Partition I, and then obtain m_i disjoint cliques C_{i_1}, $C_{i_2}, \ldots, C_{i_{m_i}}$ from the current alive clique C, simultaneously, the cliques $C_{i_1}, C_{i_2}, \ldots, C_{i_{m_i-1}}$ must have weights greater than $\frac{B}{2}$;

Step 4.1: If the last clique $C_{i_{m_i}}$ has its weight greater than $\frac{B}{2}$, then put $G := G - \bigcup_{t=1}^{m_i} C_{i_t}$ and produce the m_i cliques $C_{i_1}, C_{i_2}, \ldots, C_{i_{m_i}}$;

Step 4.2: If $m_i \geq 2$ and the last clique $C_{i_{m_i}}$ has its weight not greater than $\frac{B}{2}$, then put $G := G - \bigcup_{t=1}^{m_i-1} C_{i_t}$ and produce the $m_i - 1$ cliques $C_{i_1}, C_{i_2}, \ldots, C_{i_{m_i-1}}$;

Step 4.3: If $m_i = 1$ and the clique $C_{i_{m_i}}$ $(=C)$ has its weight not greater than $\frac{B}{2}$, then put $G := G - C$ and produce the clique C;

Step 5 Continue to execute Step 3 until $G = \emptyset$;

Step 6 Output all cliques obtained from Steps 1 and 4.

End of Clique-Partition III.

We note the following facts: (1) when Step 4.1 or 4.3 executes once, Step 4 will exactly produce the m_i cliques $C_{i_1}, C_{i_2}, \ldots, C_{i_{m_i}}$ for this time; (2) when Step 4.2 executes once, Step 4 will only produce the $m_i - 1$ cliques C_{i_1}, $C_{i_2}, \ldots, C_{i_{m_i-1}}$ for this time.

Now, we provide a proof of the correctness for the algorithm Clique-Partition III and its running complexity.

Theorem 5. *The algorithm Clique-Partition* III *is an approximation algorithm with factor* 2 *for the* MCPBI *problem, its complexity is* $\mathcal{O}(n^2)$.

Proof. We may assume that the output cliques are ordered as $C_1^0, \ldots, C_{j_0}^0, C_1^1, \ldots, C_{j_1}^1, C_1^2, \ldots, C_{j_2}^2, \ldots, C_1^t, \ldots, C_{j_t}^t$, where $C_1^0, \ldots, C_{j_0}^0$ are sequentially produced at Step 1, each clique having its weight greater than $\frac{B}{2}$, and $C_{j_1}^1$, $C_{j_2}^2, \ldots, C_{j_t}^t$ are sequentially produced at Step 4.3, each clique having its weight not greater than $\frac{B}{2}$ except the last clique $C_{j_t}^t$ (we note that the clique $C_{j_t}^t$ has its weight greater than $\frac{B}{2}$ when $C_{j_t}^t$ is the last clique produced before the algorithm stops at Step 4.1, otherwise the clique $C_{j_t}^t$ has its weight not greater than $\frac{B}{2}$), and the other cliques are sequentially produced at Steps 4.1 or 4.2, each clique having its weight greater than $\frac{B}{2}$. So the number of output cliques is $m = j_0 + j_1 + j_2 + \cdots + j_t$, and these m cliques are disjoint by the choices in our algorithm.

By the choice at Step 4.3 of our algorithm, any two cliques $C_{j_k}^k$ and $C_{j_{k'}}^{k'}$ from $\{C_{j_1}^1, C_{j_2}^2, \ldots, C_{j_t}^t\}$ can not be covered simultaneously by a clique from any optimal solution OPT, otherwise at least one of $C_{j_k}^k$ and $C_{j_{k'}}^{k'}$ is not a maximal clique, contradicting the choice at Step 4.3. Since the other cliques produced at Step 4.1 or 4.2 or Step 1 have

weight greater than $\frac{B}{2}$, then it needs at least $\lceil \frac{j_1}{2} \rceil$ cliques from any optimal solution OPT to cover the cliques $C_1^1, \ldots, C_{j_1}^1$, and it needs at least $\lceil \frac{j_2}{2} \rceil$ cliques from any optimal solution OPT to cover the cliques $C_1^2, \ldots, C_{j_2}^2$, and so on.

For each $1 \leq k \leq t$, denote $\varepsilon(j_k) = 1$ if j_k is odd and $\varepsilon(j_k) = 0$ otherwise. When $\varepsilon(j_k) = 0$, i.e., j_k is even, it needs at least $\frac{j_k + \varepsilon(j_k)}{2}$ cliques from any optimal solution OPT to cover the j_k cliques $C_1^k, \ldots, C_{j_k}^k$; and when $\varepsilon(j_k) = 1$, i.e., j_k is odd, it needs at least $\frac{j_k - 1}{2}$ cliques from any optimal solution OPT to cover the $j_k - 1$ cliques $C_1^k, \ldots, C_{j_k - 1}^k$, and simultaneously it needs at least one clique from any optimal solution OPT to cover both the clique $C_{j_k}^k$ and any clique from $C_1^0, \ldots, C_{j_0}^0$. Since all cliques produced at Step 3 are maximal in the current graphs, so the cliques from the optimal solution that cover these different cliques must be different, too. This shows that

$$\begin{aligned} |\mathrm{OPT}| &\geq \max\left\{\frac{j_0 - \sum_{k=1}^{t} \varepsilon(j_k)}{2}, 0\right\} + \sum_{k=1}^{t} \frac{j_k + \varepsilon(j_k)}{2} \\ &\geq \frac{j_0 + j_1 + \cdots + j_t}{2} = \frac{m}{2} \end{aligned}$$

which implies $m \leq 2|\mathrm{OPT}|$. So the algorithm Clique-Partition III has a factor 2 for the MCPBI problem.

Now, we analyze the complexity of the algorithm Clique-Partition III: (1) Step 1 needs at most $2n$ steps to find all heavy intervals to construct the cliques, each having weight greater than $\frac{B}{2}$; (2) Step 2 needs at most $2n$ steps to sort the light intervals, depending on the rules (1) and (2); (3) for the current interval graph G executed once, it needs a constant time to choose a smallest interval $I_{\min}$ from the current interval graph G after index sorting, so it totally needs a time at most in $\mathcal{O}(n)$; (4) since each interval must be chosen in a clique, so the steps during Steps 3 and 4, except the steps to find a smallest interval from the current interval graph G, totally needs a time in $\mathcal{O}(n^2)$ similarly to Step 3 in the algorithm Clique-Partition I. Hence, the whole algorithm needs a running time $\mathcal{O}(n^2)$.

This establishes the conclusion of the theorem. ∎

4. A linear algorithm for the special version of the MCPBI problem

In this section, we study the special version of the MCPBI problem, where all intervals have the same weight 1. When we utilize the algorithms Clique-Partition I, II or III on this special interval graph, we obtain a feasible solution whose value is not greater than 3, $\frac{5}{2}$ or 2 times that of the optimal solution. But when we modify the algorithm Clique-Partition III in some ways, we can indeed redesign an optimal algorithm for the special version of the MCPBI problem in linear time.

Our method to redesign an optimal algorithm in linear time depends on the following ideas: (1) sort all intervals depending on the rules (1) and (2); (2) choose a suitable *maximal* clique C for each circulation; (3) choose some cliques, each having its weight not greater than B, from the intervals of C by distinguishing the cardinality of such suitable maximal clique C; (4) repeatedly execute Steps 2 and 3 until $G = \emptyset$.

Since the preceding partial order '$\leq$' is a linear order on the original interval graph G, then this partial order '$\leq$' is also a linear order on the current interval subgraph of G. We sort all intervals in the time $\mathcal{O}(n)$, and then choose a smallest interval $I_{\min}$ as before in constant time at each choice in the current interval subgraph of G. Again, execute this process repeatedly until this subgraph becomes empty.

Our linear optimal algorithm is stated in the following way:

Algorithm: Clique-Partition IV
INPUT: a interval graph $G = (V, E; w)$ with intervals $I_1, \ldots, I_n$, and a bound B;
OUTPUT: m disjoint cliques consisting of these n intervals, each clique containing intervals not beyond B.
Begin
Step 1 Sort all intervals of G according to the preceding rules (1) and (2) of the linear order '$\leq$' on G;
Step 2 Choose a smallest interval $I_{\min}$ in G as the alive interval; and find the maximal clique C from G to contain such an alive interval $I_{\min}$, and then sort all intervals in C according to the preceding rules (1) and (2), without loss of generality, all intervals in C are sorted as $I_{i_1}, I_{i_2}, \ldots, I_{i_r}$, where $I_{i_1} = I_{\min}$;
Step 3 For the current alive clique $C = \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$, choose the new cliques from C, depending on the following choice regulations:
Step 3.1 If $r < B$, i.e., $|C| < B$, then output the alive clique C only containing these r intervals; and put

$G := G - \{I_{i_1}, I_{i_2}, \ldots, I_{i_r}\}$;
Step 3.2 If $r \geq B$, set $r = sB + r_0$, where $s = \lfloor \frac{r}{B} \rfloor$ and $0 \leq r_0 < B$, then output the s cliques $C_1 = \{I_{i_1}, \ldots, I_{i_B}\}$, $C_2 = \{I_{i_{B+1}}, \ldots, I_{i_{2B}}\}$, $\ldots$, $C_s = \{I_{i_{(s-1)B+1}}, \ldots, I_{i_{sB}}\}$; and $G := G - \{I_{i_1}, \ldots, I_{i_B}, \ldots, I_{i_{(s-1)B+1}}, \ldots, I_{i_{sB}}\}$;
Step 4 Continue to execute Step 2 until $G = \emptyset$;
Step 5 Output all cliques at Step 3.
End of Clique-Partition IV.

Theorem 6. *The algorithm Clique-Partition* IV *is a linear optimal algorithm for the special version of the* MCPBI *problem, where all intervals have the same weight* 1.

Proof. We may assume that the output cliques are ordered as $C_1^1, \ldots, C_{j_1}^1, C_1^2, \ldots, C_{j_2}^2, \ldots, C_1^t, \ldots, C_{j_t}^t$, where $C_{j_1}^1$, $C_{j_2}^2, \ldots, C_{j_t}^t$ are sequentially produced at Step 3.1 for the case $r < B$, each clique having its weight less than B except the last clique $C_{j_t}^t$ (we note that the clique $C_{j_t}^t$ exactly has its weight B when $C_{j_t}^t$ is the last clique produced before the algorithm stops at Step 3.2), and the other cliques are sequentially produced at Step 3.2 for the case $r \geq B$ in the sequential circulations, each clique having exactly weight B. So the number of output cliques is $m = j_1 + j_2 + \cdots + j_t$, and these m cliques are disjoint by the choices in the algorithm.

By the choice at Step 3.1 of our algorithm, any two cliques $C_{j_k}^k$ and $C_{j_{k'}}^{k'}$ from $\{C_{j_1}^1, C_{j_2}^2, \ldots, C_{j_t}^t\}$ can not be covered simultaneously by a clique from any optimal solution OPT, otherwise at least one of $C_{j_k}^k$ and $C_{j_{k'}}^{k'}$ is not a maximal clique, contradicting the choice at Step 3.1.

Since the other cliques produced at Step 3.2 have the same weight B, then it needs at least j_1 cliques from any optimal solution OPT to cover the cliques $C_1^1, \ldots, C_{j_1}^1$, and it needs $j_2 + 1$ cliques from any optimal solution OPT to cover the cliques $C_{j_1}^1, C_1^2, \ldots, C_{j_2}^2$, and so on. Then, for any optimal solution OPT, we must have $|\mathrm{OPT}| \geq j_1 + j_2 + \cdots + j_t = m$, implying $|\mathrm{OPT}| = m$.

Hence, the output cliques $C_1^1, \ldots, C_{j_1}^1, C_1^2, \ldots, C_{j_2}^2, \ldots, C_1^t, \ldots, C_{j_t}^t$ are the elements of an optimal solution OPT to the special version of the MCPBI problem.

Now, we analyze the complexity of the algorithm Clique-Partition IV: (1) Step 1 needs $2n$ steps to sort the n intervals; (2) for the current interval graph G at each circulation, it needs a constant time to choose a smallest interval $I_{\min}$ from the current interval graph G, so it totally needs at most a time in $\mathcal{O}(n)$ to find such smallest intervals; (3) each interval must be chosen in a clique, so the steps during Steps 2 and 3, except the steps to find such a smallest interval from the current interval graph G, totally needs a time in $\mathcal{O}(n)$. Hence, the whole algorithm needs a running time in $\mathcal{O}(n)$, i.e., the whole algorithm runs in linear time.

This establishes the conclusion of the theorem. ∎

5. Conclusion

In this paper, we have studied the minimum clique partition problem with constrained weight in interval graphs, and we have proved that this problem is *NP*-hard and it cannot be approximated within a factor $\frac{3}{2} - \varepsilon$ in polynomial time for any $\varepsilon > 0$. Then we have designed three approximation algorithms with different constant factors for this problem and redesigned an optimal algorithm in linear time to solve the problem for the version where all intervals have the same weights.

For further work, we will design an approximation algorithm for the MCPBI problem with a factor $\frac{3}{2}$, which will show the tight factor $\frac{3}{2}$, by adding the result of Theorem 2. On the other hand, we will design some approximation algorithms within a factor 2 to possess lower complexity.

Acknowledgements

The authors are very grateful to two anonymous referees whose helpful suggestions and comments have led to a substantially improved presentation of the paper.

Mingxia Chen, Jianping Li and Weidong Li are supported by the National Natural Science Foundation of China [No. 10561009,10271103] and the Natural Science Foundation of Yunnan Province [No. 2006F0016M]. Jianping Li is also partially supported by the SRF for ROCS, SEM, China.

Lusheng Wang is supported by a grant from City University of Hong Kong [No. 7001996].

References

[1] S. Benzer, On the topology of the genetic fine structure, Proceedings of the National Academy of Sciences of the United States of America 45 (1959) 1607–1620.
[2] Hans L. Bodlaender, K. Jansen, Restrictions of graph partition problems: Part I, Theoretical Computer Science 148 (1995) 93–109.
[3] M. Carlisle, E. Lloyd, On the k-coloring of intervals, Discrete Applied Mathematics 59 (1995) 225–235.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of *NP*-Completeness, W.H. Freeman, San Francisco, 1979.
[5] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, London, Toronto, 1980.
[6] H. Kaplan, R. Shamir, Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques, SIAM Journal on Computing 25 (3) (1996) 540–561.
[7] C.H. Papadimitriou, M. Yannakakis, Scheduling interval-order tasks, SIAM Journal on Computing 8 (3) (1979) 405–409.
[8] F.S. Roberts, Graph Theory and its Applications to the Problem of Society, in: CBMS-NSF Monograph, vol. 29, SIAM Publications, 1978.
[9] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer, 2002.
[10] R. Tarjan, Data Structures and Network Algorithms, SIAM, Philadelphia, PA, 1983.
[11] M.S. Waterman, J.R. Griggs, Interval graphs and maps of DNA, Bulletin of Mathematical Biology 48 (2) (1986) 189–195.
[12] D.B. West, Introduction to Graph Theory, second ed., Prentice-Hall, Inc., 2001.