On-line algorithm for the minimal b-clique cover problem in interval graphs

Bernard Kouakou

On-line algorithm for the minimal b-clique cover problem in interval graphs

B. Kouakou[1]

Centre d'Economie de la Sorbonne, CERMSEM, Université Paris 1, 106–112 bd de l'Hôpital, F-75013 Paris, France
kouakou@univ-paris1.fr

Abstract. In this paper we study the on-line b-clique cover problem in interval graphs. In our model the n vertices of the final instance-graph are revealed one-by-one, and we want to find a clique cover with the additional requirement that every clique cannot contain more than b vertices, where b is an integer. We first give a result which can be seen as a reduction of the on-line b-clique cover problem into the on-line clique cover problem. Then, we derive hardness results showing limits of every algorithm solving the on-line b-clique cover problem.

Keywords: on-line algorithm, clique cover problem, interval graphs, competitive ratio.

1 Introduction

In on-line computation, an algorithm has to solve a combinatorial optimization problem with the constraint that the instance is not a priori completely known before the resolution begins. In other words, the data-set is revealed step-by-step and one has, at the end of each step, to irrevocably decide on the final solution dealing with this step. Each decision must be made based of past data without information about the future. An on-line problem is defined by:

- a combinatorial optimization problem;
- a set of rules R describing how the final instance will be revealed;
- a set of rules R' defining what kind of decisions are allowed.

In this paper, we deal with the on-line minimal b-clique cover problem for interval graphs. A clique cover of a graph $G = (V, E)$ is a collection of cliques $W_1, \cdots, W_r$, such that $\bigcup_{i=1}^{r} W_i = V$. A clique cover of the smallest cardinality, r, is called a minimal clique cover; and in this case r is denoted by $k(G)$. The problem of finding, in a graph G, a clique cover of smallest cardinality r is called the minimal clique cover problem, CC. A specific case of this problem consists of minimizing the number of cliques in the cover, under the restriction that every clique must not have more than b nodes, where b is a given integer. This problem is called the b-clique cover problem and denoted by b-CC.
Throughout this study, graphs under consideration are *interval graphs*. Golumbic [9] defined this class of graphs as follows: let F be a family of nonempty sets. The intersection graph of F is obtained by representing each set in F by a vertex and connecting two vertices of F by an edge if and only if their corresponding sets intersect. The intersection graph of a family of intervals on a linear ordered set (like the real line) is called an *interval graph*.
The class of interval graphs have been intensively studied, in particular because of their practical applications (e.g., in memory allocation and in organizing records in databases [1]). Let us consider the following application [9] of interval graphs.

Application [9] *Suppose that $c_1, c_2, \cdots, c_n$ are chemical compounds which must be refrigerated under closely monitored conditions. If compound c_i must be kept at a constant temperature between t_i and t'_i degrees, how many refrigerators will be needed to store all the compounds?*
Let G be an interval graph with vertices $c_1, c_2, \cdots, c_n$ and connect two vertices whenever the temperature intervals of their corresponding compounds intersect. If $c_{i_1}, c_{i_2}, \cdots, c_{i_k}$ is a clique of G, then the intervals $[t_{i_j}, t'_{i_j}]$ corresponding to each compound c_{i_j}, $j = 1, 2, \cdots, k\}$, will have a common point of intersection, say t. A refrigerator set at a temperature of t will be suitable for storing all

of them. An optimal storing will be found by solving the minimal clique cover problem.
If each refrigerator cannot contain more than b compounds, then we have to solve the minimal b-clique cover problem.
Moreover, if these compounds are produced by a manufacturer along the year and one has to regularly (say as soon as compounds are produced) store them in refrigerators, we obtain the on-line version of the minimal b-clique cover problem. The off-line version of the minimal clique cover is solved in polynomial time for chordal graphs [9,5], consequently for interval graphs. This same problem is also solved in polynomial time for circular arc graphs (given their representations as families of arcs) [6], for comparability graphs [8] and for the more general class of "perfect graphs", [10]. Whereas it is NP-hard for general graphs, see [4].

For the off-line b-clique cover problem, we have the following results:

1. If $b = 2$, the problem reduces to finding a maximum cardinality matching in the graph $G = (V, E)$. Jack Edmond [3] derives an efficient algorithms which is based on augmenting paths for constructing matchings. Given a (partial) matching M in a graph G, an augmenting path P is a path of edges where every odd-numbered edge (including the first and last edge) is not in M, while every even-numbered edge is. The best algorithm known for general matching runs in $O(\sqrt{|V|}|E|)$ [19].
2. For $b \in \{3, \cdots, n\}$, this problem is NP-Hard for general graphs [2]. On the other hand, Finke et al. solved in [7] the minimum b-clique cover (b-CC) in polynomial time for interval graphs. Using notations of [7], we recall that a b-clique in a graph G is a clique Q of G with size $|Q| \leq b$. A b-clique cover of the graph $G = (V, E)$ is a family $\mathcal{B}$ of b-cliques that cover V. Its size is $|\mathcal{B}|$. Let $B(G, b)$ denote the set of all b-cliques covers of G, the optimum objective value of the problem (b-CC) is:
$$k_b(G) = min_{\mathcal{B} \in B(G,b)} |\mathcal{B}| .$$

In this paper, we study an on-line model of the minimum b-clique cover problem. We denote it by b-LCC. Section 1 is devoted to generalities on competitive analysis and also presents our on-line model. In section 2 we derive a competitive ratio for the n-steps version of b-LCC where intervals are presented one-by-one. Finally, section 3 brings to the fore an hardness result showing limits of every algorithm which solves b-LCC.

1.1 On-line models and competitive analysis

We denote by LCC the on-line version of the minimal clique cover problem and recall that b-LCC denotes the on-line minimal b-clique cover problem. b-LCC can be defined by the quadruplet (b-CC, R, R') where R and R' describe how the final instance is revealed and how the solution is constructed. Given a problem, we can pass from an on-line version to another by changing only the rules R and R'. For b-LCC we consider the following model:
R: the interval graph $G = (V, E)$ is revealed in n steps, where $n = card(V)$. At each step, a vertex (an interval) of G is revealed together with the edges linking this vertex and already revealed vertices. Consequently, all the vertices revealed from the first step to a step i form the graph $G[V_i]$, i.e., the subgraph induced by $v_1 \cdots, v_i$.
R': at each step, one irrevocably decides whether the new vertex is introduced in the solution.
The quality of an on-line algorithm is expressed by the competitive ratio defined below.

Definition 1. *Let us consider a minimization on-line problem P, an instance σ of P and an algorithm $\boldsymbol{LA}$ for P. Furthermore, we consider a function $c_{\boldsymbol{LA}}(n)$ (n denotes the order of the instance). Let $\boldsymbol{LA}(\sigma)$ denote the value of the solution of P computed by the algorithm LA and $\sigma^\star$ the solution returned by an optimal algorithm knowing beforehand the final instance. The algorithm $\boldsymbol{LA}$ is said to guarantee the competitive ratio $c_{\boldsymbol{A}}(n)$ if, for any instance σ of P :*
$c_{\boldsymbol{LA}}(n) \times \boldsymbol{LA}(\sigma) \leq \sigma^\star \quad (c_{\boldsymbol{LA}}(n) \times \sigma^\star \leq \boldsymbol{LA}(\sigma)$ *for a maximization problem*).

The minimum clique cover problem is related to the graph-coloring problem which consists of assigning the fewest possible colors to the vertices of a graph so that adjacent vertices receive different colors. Here, we will use the same notations as in [9]:
$\chi(G)$ denotes the *chromatic number* of G: the fewest number of colors needed to properly color

vertices of G, or equivalently, the fewest number of independent sets needed to cover the vertices of G.
$\alpha(G)$ is the *stability number* of G: the size of the largest stable set of G.
$k(G)$, denotes the *clique cover number*: the fewest number of complete subgraphs (cliques) needed to cover the vertices of G.
$w(G)$ is the *clique number* of G: the size of the largest clique of G.
For all graph (so for interval graphs), we have $k(G) = \chi(\bar{G})$ and $\alpha(G) = w(\bar{G})$, where $\bar{G}$ is the graph complement of G . There is a number of strong results related to on-line coloring algorithms for interval graphs; see e.g. [17,18,20,21]. For the on-line coloring of general graphs, the interested reader can see [12,13]. One can also see [14] for bounded coloring. Since the complements of interval graphs are not in general interval graphs, the relation $k(G) = \chi(\bar{G})$ does not help to solve the minimal clique cover problem via coloring.
In [11], Gyarfas and Lehel derives an absolute competitive ratio for the on-line minimal clique cover problem, LCC, on chordal graphs. They proved that if G is a chordal graph then, the value $k_{FF}(G)$ of the solution returned by the First Fit algorithm satisfies $k_{FF}(G) \leq 2\alpha(G) - 1$.
Let us now focus our attention on the on-line b-clique cover problem.

1. If $b = 2$, this problem is equivalent to the on-line maximum-cardinality matching problem which is solved by a 1/2-competitive algorithm.

Let us now consider the on-line version of the b-clique cover problem, with $b \in \{3, \cdots, n\}$.

2 The n-steps on-line minimal b-clique cover problem, (b-LCC)

In this section, we consider the on-line clique cover problem where for every interval graph $G = (V, E)$, its n vertices are revealed one-by-one (Such instance will be called an *n-steps instance*). The on-line algorithm has to irrevocably assign a clique to each vertex as soon as it is revealed. To our knowledge, it is the first time that one considers this on-line problem. In what follows, a clique C_i containing exactly b nodes (or intervals) is said to be *saturated*, i.e $|C_i| = b$; in the opposite case, $|C_i| < b$, and C_i is *unsaturated*. We first prove that b-LCC reduces to LCC. Indeed, if LA is an algorithm for LCC, then one can transform every clique C_i returned by LA into $\lfloor \frac{C_i}{b} \rfloor$ saturated cliques and probably one unsaturated clique. With such away every solution of LCC can be transformed into a solution of b-LCC. The question is how to exploit the competitive ratio guaranteed by an algorithm for LCC to derive results for (b-LCC)? Let us consider an algorithm LA solving LCC. We propose below an algorithm called b-LA which solves b-LCC.

***Algorithm* b-LA**: *use LA to solve the graph-instance I. Let $LA(I) = C_1, C_2, \cdots, C_\lambda$ be the solution returned by LA. Then, we form a solution for b-LCC by clustering each clique C_i into $\lfloor \frac{C_i}{b} \rfloor$ saturated cliques and possibly one unsaturate clique.*

The following theorem holds.

Theorem 1. *If an on-line algorithm LA guarantees the competitive ratio C_{LA} for the minimum clique cover problem, then its corresponding algorithm b-LA, for b-LCC, guarantees the competitive ratio $\frac{C_{LA}}{1+C_{LA}}$.*

Proof. Let us consider Algorithm b-LA solving b-LCC. Let N_s be the number of saturated cliques returned by b-LA and N_u the number of unsaturated cliques. Then, the total number $\lambda_b(G)$ of cliques returned by b-LA satisfies:

$$\lambda_b(G) = N_s + N_u \tag{1}$$

Let denote by $\overline{\chi}_b(G)$ (resp. $\lambda(G)$) the optimal value for the b-clique cover problem (resp. the number of cliques returned by LA for LLC). $\overline{\chi}_b(G)$ can also be defined as the b-bounded co-chromatic number. Then, we have $N_u \leq \lambda(G)$ and $N_s \leq \overline{\chi}_b(G)$. Moreover, relation (1) implies (recall that $k(G)$ denotes the minimal clique number of G):

$$\lambda_b(G) \leq \overline{\chi}_b(G) + \lambda(G) \leq \overline{\chi}_b(G) + \frac{1}{C_{LA}} k(G), \tag{2}$$

where the last inequality of expression (2) holds since LA is C_{LA}-competitif. Let us also note that, for a given instance graph G, every solution of b-LLC is a feasible solution of LLC. Therefore, as they are minimization problems, we have $k(G) \leq \overline{\chi}_b(G)$. Relation (2) becomes:

$$\lambda_b(G) \leq \overline{\chi}_b(G) + \frac{1}{C_{LA}}\overline{\chi}_b(G)$$

i.e

$$\lambda_b(G) \leq (1 + \frac{1}{C_{LA}})\overline{\chi}_b(G)$$

Corollary 1. *b-LCC admits an asymptotic $\frac{1}{3}$-competitive algorithm for chordal graphs, so for interval graphs.*

Proof. In what follows a clique C_i is compatible with an interval I_j if all intervals in C_i intersect I_j. The classical First Fit (FF) algorithm used to solve the Bin-Packing problem [15,16] can be adapted to the minimum clique cover problem as follows:
FF : *Start with no clique and insert interval I_j ($j = 1, \cdots, n$) in the clique which has the lowest index i and is compatible with I_j. If no such a clique exists, then open a new clique to pack I_j.*
In [11], Gyarfas and Lehel proved that FF is 1/2-competitive for LCC on chordal graphs. Then applying Theorem (1), the result of corollary 1 immediately follows.

For the sequel, we need to recall the following definition used in [7].

Definition 2. *Let us denote by G_k the on-line graph induced by the first k elements $\{v_i, \cdots, v_k\}$ of V. Assume that C is a clique that is already in use after processing G_k. The range $r_k[C]$ of C at phase k is defined as follows: $r_k[C] = \bigcap\{v_j : 1 \leq j \leq k, A(v_j) = C\}$, i.e., $r_k[C]$ is the intersection of all the intervals (vertices) put in the clique C after revealing the first k vertices of G.*

3 Hardness result for the on-line minimum b-clique cover in interval graphs.

We consider an interval graph $G = (V, E)$ of which the n vertices will be revealed (one-by-one) as we progress with the resolution. One can think of an on-line problem as a game between two players. One player, the adversary, generates requests on-line according to some specified mechanism and is charged a cost for its selections according to some specified rules. For example, for a minimization problem, the adversary is charged the minimum (off-line) cost to perform any request sequence it generates. The other player is the algorithm, which responds to requests by making a decision, and incurs some total cost over the course of the game. In our model, the adversary has complete knowledge of the algorithm and it can first decide how many requests it wants to generate; then, it internally simulates the algorithm on each possible sequence of that length to find the sequence that maximizes the ratio between the on-line algorithm's cost and the adversary's cost. We emphasize that, as we deal with a minimization problem, the adversary's goal is to maximize the ratio of the algorithm's cost to the adversary's cost, while the on-line algorithm's goal is to minimize it. We have the following result.

Theorem 2. *No on-line algorithm for b-LCC can guarantee a competitive ratio $c_{LA} > 1/2$ on interval graphs.*

Proof. Let b-LA be an on-line algorithm for b-LCC. Let us consider an interval graph G of which vertices (or intervals) are revealed one-by-one. The adversary first reveals, one-by-one, a sequence of $N = 2k$ intervals (or nodes). Two main situations may occur:
Case 1: algorithm b-LA is greedy: at a step j of the on-line process, b-LA does never construct a new clique for a revealed interval I_j if I_j can be put in an existing clique C_i, with $i \leq j$; for example algorithm First Fit is greedy. In this case the adversary adopt the following strategy:
(i) For $i = 1, 2, \cdots, 2k-1$, interval I_i is linked to I_{i+1}, but for $i = 1, 2, \cdots, k-1$, intervals I_{2i-1} and I_{2i+1} are not connected. Moreover, for $i = 2, \cdots, k$, intervals I_{2i-2} and I_{2i} do not also intersect. Clearly, this first sequence of $2k$ intervals form a path. (ii) Let denote by $E_1, E_2, \cdots E_k$ the set of k edges ($k \geq 1$) selected by the greedy algorithm, $E_i = (I_{2i-1}, I_{2i})$, $i = 1, \cdots, k$. Then, the adversary

reveals one-by-one a number of $M = (b-2)(k+1)+2$ intervals such that: $b-1$ intervals form a clique C_0 with I_1 and no interval of C_0 different from I_1 is connected to another interval which is out of C_0. For each $i \in \{1, 2 \cdots, k-1\}$, $b-2$ intervals form a clique C_i with both intervals I_{2i} and I_{2i+1}. Moreover these $b-2$ intervals do not intersect another interval out of the clique C_i. The last $b-1$ intervals form a clique C_{k+1} with I_{2k} and no interval of C_{k+1} different from I_{2k} is connected to an interval out of C_{k+1}.

Case 2. Algorithm b-LA does not adhere to the greedy principle. In this case:

1. The adversary first reveals one-by-one a sequence of $N = 2k$ intervals as follows (each interval $I_i, i = 1, \cdots, 2k$ will be denoted by $[a_i, b_i]$):
   (i) for $i = 1, 2, \cdots, k$, interval I_{2i-1} is linked to I_{2i}, but, for $i = 1, 2, \cdots, k-1$, I_{2i} is linked to I_{2i+1} if and only if algorithm b-LA has selected I_{2i-1} and I_{2i} in the same clique.
   (ii) If at some moment the algorithm puts I_{2i-1} and I_{2i} in a same clique, then the next two intervals, I_{2i+1} and I_{2i+2}, to be revealed are such that $a_{2i} < a_{2i+1} \leq b_{2i} < b_{2i+1}$ and $b_{2i} < a_{2i+2} \leq b_{2i+1} < b_{2i+2}$. In this case and only for this first sequence of N intervals, every interval $I_j, j > 2i$, to be revealed has to satisfy $b_{2i-1} < a_j$.
   (iii) For $i = 1, 2, \cdots, k$, if at some moment, Algorithm b-LA does not select both intervals I_{2i-1} and I_{2i} in a same clique, then the remaining intervals, $\{I_j, j = 2i+1, \cdots, 2k\}$, to be revealed in this first sequence are such that $b_j < a_{2i-1}$. We emphasize that if I_{2i-1} and I_{2i} are put in two different cliques, and if before I_{2i-1} and I_{2i} the algorithm has put intervals I_{2t-1} and $I_{2t}, (t < i)$ in the same clique, then every interval $I_j, j > 2t$, to be revealed in this first sequence of N intervals satisfies $b_{2t-1} < a_j$.

2. If after revealing this sequence of N intervals, no edge is selected by the algorithm, then the game stops. In the opposite case, let denote by $E_1, E_2, \cdots E_p$ the set of P edges selected by b-LA. Then, the adversary reveals one-by-one a number of $M = (b-2)(p+1)+2$ intervals (see bold intervals in Figure 1) such that: $b-1$ intervals form a clique C_0 with the initial endpoint $I_{i,1}$ of E_1 and these $b-1$ intervals have no link with the others intervals different from $I_{i,1}$ in the final graph-instance.
   For $j \in \{1, 2 \cdots, p-1\}$, $b-2$ intervals form a clique with the terminal endpoint, $I_{t,j}$, of E_j and the initial endpoint, $I_{i,j+1}$, of E_{j+1}. Moreover, except these two endpoints, the $b-2$ intervals have no connexion with the others intervals of the final graph G. The last $b-1$ intervals form a clique with the terminal endpoint of E_p and they are not linked to another interval of G.

The figure 1 illustrates a solution returned by a non-greedy algorithm b-LA. When no ambiguity occurs, an interval I_j will also be identified to its index j. In figure 1, intervals $\{1, 2, \cdots, 12\}$ constitute the first sequence of revealed intervals (they are revealed one-by-one). Let us consider that b-LA first forms cliques $\{1, 2\}$ and $\{3, 4\}$. So, the initial endpoints of intervals $\{I_j, j = 5, \cdots, 12\}$ are greater than the terminal endpoint of I_3. Now let us assume that b-LA puts intervals 5 and 6 in two different cliques. In this case, all intervals $\{I_j, j = 7, \cdots, 12\}$ to be revealed are such that their terminal endpoint are lower than the initial endpoint of interval 5. Algorithm b-LA also puts 7 and 8 in two different cliques. That justifies the position of intervals $\{9, \cdots, 12\}$ in figure 1. Finally, for this first sequence of 12 intervals, we consider that b-LA forms cliques $\{9, 10\}$ and $\{11, 12\}$.
After these 12 intervals, the adversary reveals one-by-one the second sequence of intervals, i.e., $\{13, 14, \cdots, 19\}$ (they are represented into bold font in figure 1).

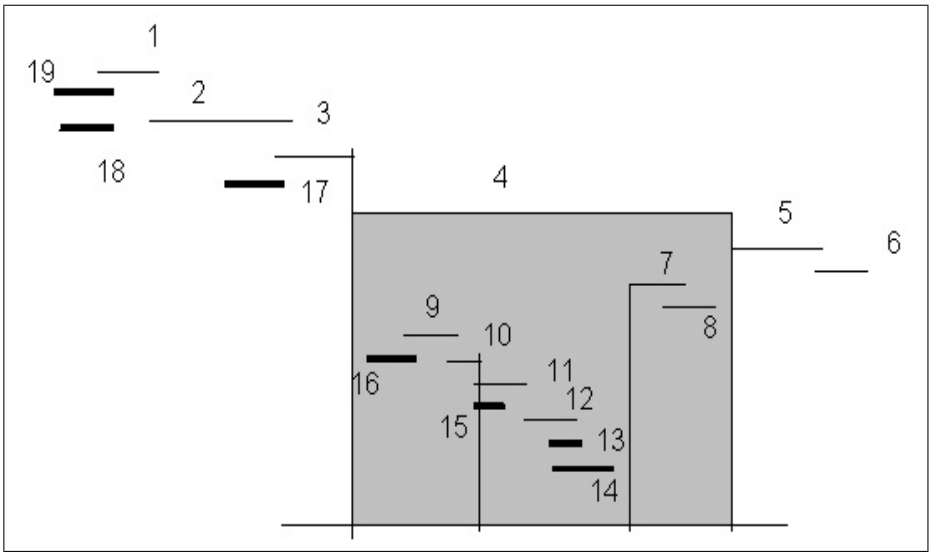

Fig. 1. An interval-representation of the final instance, with $b = 3$

Let S be the number of single clique returned by b-LA in the first sequence of revealed intervals (for the instance here revealed by the adversary, S=0 if algorithm b-LA is a greedy one). Let also denote by T the number of the remaining revealed intervals, $T = M + 2P = b(p + 1)$. Then, the total number of revealed intervals is $n = S + T$. In [7], Finke et al. proved that the following First Fit algorithm optimally solves the off-line b-clique cover problem on intervals graphs.
Consider the tasks in nondecreasing order $I_1, ..., I_n$ of their terminal endpoints b_i, breaking ties arbitrarily.
Start with no clique and insert interval $I_j (j = 1, ..., n)$ in the unsaturated clique B_i which has lowest index i and is compatible with Ij; if there is no such (unsaturated and compatible) batch then a new batch is created and interval I_j is put into it.

Using this algorithm for the instance here revealed, it forms $S/2$ edges (i.e $S/2$ 2-cliques) and T/b b-cliques. So, The value of the optimal solution verifies:

$$\overline{\chi}_b(G) = \frac{S}{2} + \frac{T}{b} = \frac{bS + 2T}{2b} \tag{3}$$

Let us determine the number of cliques returned by the algorithm b-LA. It fisrt constructs: S single cliques and p edges. Then, with the second sequence of intervals, it forms in best case, two $(b-1)$-cliques and $p-1$ $(b-2)$-cliques. Therefore, in best case,

$$\begin{aligned} \lambda &= S + 2p + 1 \\ &= S + 2(\frac{T}{b} - 1) + 1 \text{ since } p = \frac{T}{b} - 1 \\ &= \frac{bS + 2T - b}{b} \end{aligned} \tag{4}$$

Then, revisiting relation (3), the expresion (4) becomes: $\lambda = 2\overline{\chi}_b(G) - 1$ Then,

$$C_{LA} = \frac{\overline{\chi}_b(G)}{2\overline{\chi}_b(G) - 1}$$

Finally, for $\overline{\chi}_b(G)$ large enough the asymptotic competitive ratio of 1/2 is tight.

4 Conclusion

This work points out two results related to the on-line b-cliques cover problem on interval graphs: we first proved that the on-line clique cover problem, LCC, reduces to its bounded version called b-LCC, where each clique is of size at most b. This reduction leads us to a 1/3-competitive algorithm

for *b-LCC* on chordal graphs.
Then, we derive a hardness result for *b-LCC* on intervals graphs: we show that if the vertices of an interval graph are revealed one-by-one, then the *b-LCC* problem cannot be solved strictly better than 1/2-competitively.
It seems appropriate to analyze the competitive behavior of an on-line algorithm which adheres to the First Fit principle. Such an algorithm puts, at each step j, the revealed interval in the unsaturated clique which has the lowest index i and is compatible with I_j . If no such a clique exists, then a new clique is opened to pack I_j. The model of *b-LCC* where intervals are revealed by cluster seems also to be an interest research area that one can explore.

References

1. *S. Booth and , S. and S. Lueker*: Testing for consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. J. Comput. syst. Sci., 13:335-379, 1976.
2. *M. Boudhar and G. Finke*: Scheduling on batch processing machines with constraints of compatibility between jobs. In Proc. Second Conference on Management and Control of Production and Logistics, (MCPL'2000), volume 2, pages 703-708, 2000.
3. *J. Edmonds*: Paths, trees, and flowers. Canadian J. Math, 17:449-467, (Al.l; AI.2), 1965.
4. *M.R. Garey and D.S. Johnson*: Computers and intractability. a guide to the theory of *NP*-completeness. Discr. Appl. Math, 1979.
5. *F. Gavril*: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. , SIAM J. Comput, 1:180-187. (Al.l; AI.2; A4.1), 1972.
6. *F. Gavril*: Algorithms on circular-arc graphs. Networks, 4:357-369, (Al.l; A 1.2), 1974.
7. *M. G. Finke, V. Jost, M. Queyrianne and A. Sebő*: Batch processing with interval graph compatibilities between tasks. Les cahiers du laboratoire Leibniz, 108, Août 2004. http://www-leibniz.imag.fr/LesCahiers/.
8. *M. C. Golumbic*: The complexity of comparability graph recognition and coloring. Computing, 18:199-208, (Al.l; AI.2), 1977.
9. *M. C. Golumbic*: Algorithmic Graph Theory and Perfect Graphs. Academic Press, 1980.
10. *M. Grotschel, L. Lovász and A. Schrijver*: Polynomial algorithms for perfect graphs. Annals of Discrete Math., 21:325-256, 1984.
11. *A. Gyarfas, J. Lehel*: A First Fit on-line coloring. Journal of Graph Theory, 12:217-227, 1988.
12. *M. Halldórsson*: Online coloring known graphs. In Electronic J. of Combinatorics, Feb 2000.
13. *M. M. Halldórsson and Mario Szegedy*: Lower bounds for on-line graph coloring. Theoretical Computer Science, 130:163-174, August 1994.
14. *P. Hansen, A. Hertz and J. Kuplinsky*: Bounded vertex colorings of graphs. Discrete Mathematics, 111:305-312, 1993.
15. *D. S. Johnson*: Fast algorithms for bin packing. J. Comput. System Sci., 8:272-314, 1974.
16. *D.S Johnson*: Near-optimal bin packing algorithms. PhD thesis, MIT, Cambridge, Massachusetts, 1973.
17. *H.A. Kierstead*: Recursive and on-line graph coloring, in Handbook of recursive Mathematics, volume 2, Recursive Algebra, Analysis and combinatorics.
18. *H.A. Kierstead and J. Qin*: Coloring interval graphs with First Fit. Descrete Math, 144:47-57., 1995.
19. *S. Micali and V. Vaziraní*: An $O(\sqrt{|V|}|E|)$ Algorithm for finding maximum matchings in general graphs. In Proc. 21st. Symp. Foundations of Computing, pages 17-27, 1980.
20. *M. Slasurek*: A coloring algorithm for interval graphs. Mathematical Foundations of Computer Science, 379:471-480, 1989.
21. *M. Slasurek*: Optimal on-line coloring of circular arc graphs. RAIRO Inform. Theor. Apli., 29:423-429, 1995.