LIF

Laboratoire d'Informatique Fondamentale
de Marseille

Unité Mixte de Recherche 6166
CNRS – Université de Provence – Université de la Méditerranée

An efficient algorithm for maximum disjoint matchings among intervals and related problems

Frédéric Gardi

Rapport/Report 07-2002

May 2002

An efficient algorithm for maximum disjoint matchings among intervals and related problems

Frédéric Gardi

LIF – Laboratoire d'Informatique Fondamentale de Marseille
UMR 6166
CNRS – Université de Provence – Université de la Méditerranée

Parc Scientifique et Technologique de Luminy,
163, avenue de Luminy - Case 901,
F-13288 Marseille Cedex 9, France.

gardi@lidil.univ-mrs.fr

Abstract/Résumé

In this paper we consider the problem of determining a maximum matching in a set of intervals (two intervals can be matched if they are disjoint). We propose an incremental algorithm to compute such a maximum disjoint matching. We show that this algorithm runs in $O(n)$ time if the list of sorted endpoints of the set of intervals is given in input. This result is applied to solve efficiently particular cases of two scheduling problems: working schedules planning and mutual exclusion scheduling for interval graphs. **Keywords:** matchings, interval graphs, linear-time algorithms, scheduling.

Dans ce papier, nous considérons le problème de la détermination du couplage maximum dans un ensemble d'intervalles (deux intervalles pouvent être couplés s'ils sont disjoints). Nous proposons un algorithme incrémental pour calculer un tel couplage. Nous montrons que cet algoritme s'exécute en $O(n)$ si la liste des extrémités triées de l'ensemble des intervalles est donnée en entrée. Ce résultat est appliqué à la résolution de cas particuliers de deux problèmes d'ordonnancement: la planification d'horaires de travail et un problème d'ordonnancement avec exclusion mutuelle pour les graphes d'intervalles. **Mots-clés:** couplages, graphes d'intervalles, algorithmes linéaires, ordonnancement.

Relecteurs/Reviewers: Michel Van Canegehem, Victor Chepoi.

1 Introduction

The problem of determining a maximum matching of a graph is a basic problem in algorithmic graph theory. It occurs in numerous problems of operations research and holds an important place in many practical applications. Let G be an undirected graph with n vertices and m edges. A *matching* $\mathcal{M}$ in G is a subset of pairwise nonincident edges [9]. Let $\nu(G)$ denote the size of a maximum cardinality matching in G. The first polynomial algorithm to determine a maximum matching in a graph was given by Edmonds [5]. The fastest algorithm is due to Micali and Vazirani [12]. Its complexity is $O(\sqrt{n}m)$, but it is complex and not considered practical.
On the other side, *intervals* are used to modelize many problems appearing in diverse areas like scheduling, genetics, psychology, sociology, archæology and others (see [9] and [13] for more details). Such models appear notably in graph-theoretical terms through interval graphs or interval orders. $G = (V, E)$ is an *interval graph* iff to each vertex $v \in V$ can be associated a closed (resp. open) interval I_v of the real line, such that a pair of distinct vertices u and v are adjacent in G if and only if $I_u \cap I_v \neq \emptyset$. An *interval representation* of G will be noted $\{I_v\}_{v \in V}$, with $l_v, r_v \in \mathbb{R}$ the left and right endpoints of I_v. The edges of the complement $\overline{G} = (V, F)$ (called *co-interval graph*) can be transitively oriented with $(u, v) \in \vec{F}$ iff $r_u < l_v$. This orientation $\vec{F}$ of the edges induces a partial order $P = (V, \vec{F})$ called *interval order*. We will write $I_u \prec I_v$ iff $r_u < l_v$.

In this paper, we consider the problem of determining a maximum matching in a set of intervals such that two intervals can be matched if they are *disjoint*. Here is the definition of a *maximum disjoint matching in a set of intervals*.

Definition 1 *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of intervals with $I_i = [l_i, r_i]$, $l_i, r_i \in \mathbb{R}$ and $l_i < r_i$. A disjoint matching in $\mathcal{I}$ is a set $\mathcal{M}$ of pairs of intervals of $\mathcal{I}$ such that:*

- *for all $u, v \in \{1, \ldots, n\}$, $\{I_u, I_v\} \in \mathcal{M} \Longrightarrow I_u \cap I_v = \emptyset$,*
- *for all $u, v \in \{1, \ldots, n\}$, $\{I_u, I_v\} \in \mathcal{M} \Longrightarrow$ for all $w \in \{1, \ldots, n\}$, $\{I_u, I_w\} \notin \mathcal{M}$ or $\{I_v, I_w\} \notin \mathcal{M}$.*

$\mathcal{M}$ is a maximum disjoint matching if its cardinality is maximum.
$\nu(\mathcal{I})$ denotes the size of a maximum disjoint matching in $\mathcal{I}$.

In graph-theoretic terms, this is the problem of *finding a maximum matching in a co-interval graph*.

In a recent paper [1], M.G. Andrews et al. gave an $O(n \log n)$-time complex recursive algorithm to compute a maximum matching in a co-interval graph given by the list of the endpoints of the intervals. They claimed that this one can run in linear time if the endpoints of the intervals are sorted. At our acquaintance, only one $O(n \log n)$-time algorithm based on plane sweeping was given previously by M.G. Andrews and D.T. Lee for this problem [2]. Here we propose a much simpler incremental algorithm which can be implemented in $O(n)$ time if we have in input the list of *sorted* endpoints of the n intervals. This result will be applied to solve efficiently three problems: maximum matching in co-interval graphs, working schedules planning and mutual exclusion scheduling for interval graphs with special instances.

2 Preliminaries

Here are some definitions which we will use to describe clearly our matching algorithm. These ones are derived from the graph theory terminology [9].

Definition 2 *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of intervals with $I_i = [l_i, r_i]$.
A clique of $\mathcal{I}$ is a set $C \subseteq \mathcal{I}$ such that $I_u \cap I_v \neq \emptyset$ for all $I_u, I_v \in C$. The cardinality of the largest clique of $\mathcal{I}$ is denoted by $\omega(\mathcal{I})$. A stable of $\mathcal{I}$ is a set $S \subseteq \mathcal{I}$ such that $I_u \cap I_v = \emptyset$ for all $I_u, I_v \in S$. A q-coloring or a partition of size q of $\mathcal{I}$ into stables is a partition $\mathcal{S} = \{S_1, \ldots, S_q\}$ of $\mathcal{I}$ such that each S_i is a stable. The chromatic number $\chi(\mathcal{I})$ of $\mathcal{I}$ is the size of a partition of $\mathcal{I}$ into the least number of stables.*

Intervals own numerous properties which can be found in [9]. Here is proposition which our algorithm is based on.

Proposition 1 (Berge, 1960 [9]) *For a set of intervals $\mathcal{I}$, $\omega(\mathcal{I}) = \chi(\mathcal{I})$.*

Moreover, computing a maximum clique or a minimum coloring of $\mathcal{I}$ can be done in linear time [9, 10].

Next, we must define some orders on intervals that we use extensively in the next sections.

Definition 3 *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of intervals with $I_i = [l_i, r_i]$.
We denote by $<$ the order on $\mathcal{I}$ such that $I_u < I_v$ iff $l_u < l_v$ or ($l_u = l_v$ and $r_u \leq r_v$) and by $>$ the order on $\mathcal{I}$ such that $I_u > I_v$ iff $r_u > r_v$ or ($r_u = r_v$ and $l_u \geq l_v$).*

At last, another crucial notion which will appear in analysis of our algorithm is the convexity of bipartite graphs.

Definition 4 *A bipartite graph $G = (X, Y, E)$ is Y-convex iff there is an ordering $\lhd$ on Y such that if $(i, x) \in E$ and $(i, z) \in E$ with $i \in X$ and $x, z \in Y$, $x \lhd z$ implies that $(i, y) \in E$ for $x \lhd y \lhd z$.*

We can note that a convex bipartite graph G can be given by specifying the ordering $\lhd$ and for every $i \in X$, two values a_i and b_i, respectively the smallest and largest elements in the interval of the (ordered) vertices of Y connected to i. A maximum matching in a convex bipartite graph can be computed in linear time [8, 7, 14].

3 The matching algorithm

Before describing the algorithm in details, we outline the main ideas behind the algorithm and its correctness. The matching algorithm is notably based on the following result.

Theorem 1 *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of intervals with $I_i = [l_i, r_i]$. Let $\mathcal{S} = \{S_1, \ldots, S_{\chi(\mathcal{I})}\}$ be a partition of $\mathcal{I}$ into the minimum number of stables such that the number $s(\mathcal{I})$ of stables consisting of only one interval is as small as possible. Then the size of the maximum matching in $\mathcal{I}$ is $\nu(\mathcal{I}) = \lfloor \frac{n - s(\mathcal{I})}{2} \rfloor$.*

Proof. The proof of this assertion is essentially algorithmic. It is based on two lemmas.

Lemma 1 *If $|S_i| \geq 2$ for all $i = 1, \ldots, \chi(G)$ then $\mathcal{I}$ admits a perfect disjoint matching (i.e. a disjoint matching of size $\nu(\mathcal{I}) = \lfloor \frac{n}{2} \rfloor$).*

Proof of Lemma 1. First, we are going to show that from two stables S_i and S_j of odd size $2t + 1 \geq 3$, it is possible to extract one pair of intervals $\{I_u, I_v\}$ such that $I_u \in S_i$ and $I_v \in S_j$ and to redefine two new stables of even size. Let $I_a, I_b \in S_i$ and $I_c, I_d \in S_j$ be such that (i) $I_a \prec I_b$ and (ii) $I_c \prec I_d$. If $I_a \cap I_d = \emptyset$ then $\{I_a, I_d\}$ induces the wished pair of intervals. Otherwise, if (iii) $I_a \cap I_d \neq \emptyset$, then $I_b \cap I_c = \emptyset$ and $\{I_b, I_c\}$ is the desired pair of intervals (see Fig. 1). Indeed, (iii) implies that $l_d \leq r_a$. Now, by (i) and (ii), we have $r_a < l_b$ and $r_c < l_d$, and so $r_c < l_b$, thus establishing our assertion.

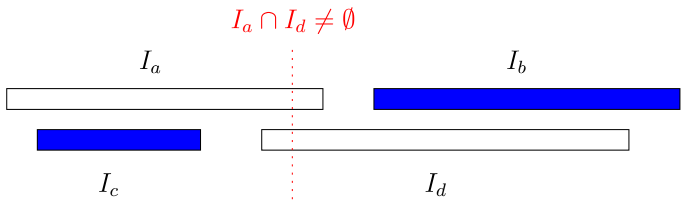

Fig. 1. The case where $I_a \cap I_d \neq \emptyset$ in the proof of Lemma 1.

At last, from stables of even size $2t \geq 2$, we can trivially extract t pairs of intervals, which allows us to conclude the proof of the lemma.

(Proof of Lemma 1) $\diamondsuit$

Lemma 2 *If $S_i = \{I_u\}$, i.e. $|S_i| = 1$, then I_u belongs to any maximum clique of $\mathcal{I}$.*

Proof of Lemma 2. Since $\omega(\mathcal{I}) = \chi(\mathcal{I})$ (Proposition 1), each stable S_i has an interval in each maximum clique of $\mathcal{I}$. Thus, if $S_i = \{I_u\}$, then I_u belongs necessarily to all maximum clique of $\mathcal{I}$.

(Proof of Lemma 2) $\diamondsuit$

By these two lemmas, we affirm that the problem of determining a maximum matching in $\mathcal{I}$ is equivalent to the problem of finding a partition of $\mathcal{I}$ into the minimum number of stables such that the number $s(\mathcal{I})$ of stables having only one interval is minimized. Indeed, if we have a minimum number $s(\mathcal{I})$ of stables of size one in $\mathcal{S}$, then Lemma 2 imposes that $s(\mathcal{I})$ intervals of $\mathcal{I}$ cannot belong to a maximum matching in $\mathcal{I}$ and Lemma 1 allows us to compute a perfect disjoint matching in the set of remaining $n - s(\mathcal{I})$ intervals.

$\diamondsuit$

We showed, by Theorem 1, that our problem could be reduced to a problem of minimization of the number of stables of size one in a partition $\mathcal{S}$ of $\mathcal{I}$ into the minimum number of stables. According to Lemma 2, we can solve this problem by computing a maximum matching between the intervals of a maximum clique C of $\mathcal{I}$ and the intervals of the set $\mathcal{I} \backslash C$. Indeed, having this maximum matching, denoted by $\mathcal{M}_b$, we will use the next procedure to complete a maximum number of stables of size one by one interval. Finally, after minimizing the number of stables having one interval, we will produce a maximum dijoint matching in $\mathcal{I}$ by using the algorithmic method described in the proof of Lemma 1.

Procedure Complete_Stables;
Input: $\mathcal{S} = \{S_1, \ldots, S_{\chi(\mathcal{I})}\}$ a minimum partition of $\mathcal{I}$ into stables;
$\mathcal{M}_b$ a maximum matching between the intervals of a maximum clique C of $\mathcal{I}$ and the intervals of the set $\mathcal{I} \setminus C$;
Output: $\mathcal{S}$ with a minimum number of stables of size 1;

begin;
  while there is $S_i = \{I_u\}$, i.e. $|S_i| = 1$, and $\mathcal{M}_b \neq \emptyset$ do
    if $\exists I_v \in S_j$ such that $\{I_u, I_v\} \in \mathcal{M}_b$ do
      $S_j \leftarrow S_j \setminus \{I_v\}$;
      $S_i \leftarrow S_i \cup \{I_v\}$;
      $\mathcal{M}_b \leftarrow \mathcal{M}_b \setminus \{I_u, I_v\}$;
end;

Lemma 3 *The procedure Complete_Stables minimizes the number of stables of size one in $\mathcal{S}$.*

Proof. Suppose that there are $I_u \in S_i$, $|S_i| = 1$, and $I_v \in S_j$, $|S_j| \geq 3$, such that $I_u \cap I_v = \emptyset$, this would imply that the matching $\mathcal{M}_b$ is not maximum, in contradiction with the hypothesis done in input of the procedure.

$\diamondsuit$

Now we can give a complete description of our matching algorithm.

Algorithm Matching_Disjoint_Intervals;
Input: $\mathcal{I} = \{I_1, \ldots, I_n\}$ with $I_i = [l_i, r_i]$;
Output: $\mathcal{M}$ a maximum disjoint matching in $\mathcal{I}$;

begin;
Stage 1:
  compute $\mathcal{S} = \{S_1, \ldots, S_{\chi(\mathcal{I})}\}$ a minimum partition of $\mathcal{I}$ into stables;
  if for all $i = 1, \ldots, \chi(G)$, $|S_i| \leq 2$ do goto Stage 3;
  if for all $i = 1, \ldots, \chi(G)$, $|S_i| \geq 2$ do goto Stage 3;
Stage 2:
  compute a maximum clique C of $\mathcal{I}$;
  let $G_b = (X, Y, E)$ be the bipartite graph such that
  $X = C$, $Y = \mathcal{I} \setminus C$, $E = \{e = I_iI_j, I_i \in C, I_j \in \mathcal{I} \setminus C \mid I_i \cap I_j = \emptyset\}$;
  compute a maximum matching $\mathcal{M}_b$ in G_b;
  Complete_Stable($\mathcal{S}$,$\mathcal{M}_b$);
Stage 3:

for each $S_i \in \mathcal{S}$ do
  if $|S_i| = 1$ do $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S_i\}$;
compute a perfect matching $\mathcal{M}$ in $\mathcal{S}$;
return $\mathcal{M}$;
end;

4 Complexity of the matching algorithm

We will analyse (according to the RAM model [4]) the time and space complexities of the matching algorithm. We do this analysis stage by stage considering that we have in input the list of endpoints of the n intervals and the orders $<$ and $>$ on $\mathcal{I}$ (Definition 3), i.e. a list of intervals sorted according the left (resp. the right) endpoints.

Stage 1 can be done in $O(n)$ time. Indeed, a minimum coloring of a set of intervals can be done in $O(n)$ time if the orders $<$ and $>$ are given [10]. The complexity of Stage 2 relies on the next lemma.

Lemma 4 ([1]) *The bipartite graph $G_b = (X, Y, E)$ is Y-convex.*

Proof. Let $C = \{c_1, \ldots, c_{\chi(\mathcal{I})}\}$ be a maximum clique of $\mathcal{I}$. Let $\mathcal{I} \setminus C = \mathcal{I}^+ \uplus \mathcal{I}^-$ such that $\mathcal{I}^+ = \{I \in \mathcal{I} \setminus C \mid \exists c \in C,\ c \prec I\}$ and $\mathcal{I}^- = \{I \in \mathcal{I} \setminus C \mid \exists c \in C,\ I \prec c\}$. By ordering $\mathcal{I}^+$ according the increasing l_i and $\mathcal{I}^-$ according the increasing r_i, we obtain an order on $\mathcal{I}$ such that $\forall c \in C$ with $c = [l_c, r_c]$, if a_c is the smallest index such that $c \cap I_{a_c} = \emptyset$ with $I_{a_c} \in \mathcal{I}^+$ and b_c is the largest index such that $c \cap I_{b_c} = \emptyset$ with $I_{b_c} \in \mathcal{I}^-$, then for all $j \in \{a_c, \ldots, b_c\}$, $c \cap I_j = \emptyset$, i.e. $cI_j \in E$ (see Fig. 2) and so G_b is Y-convex.

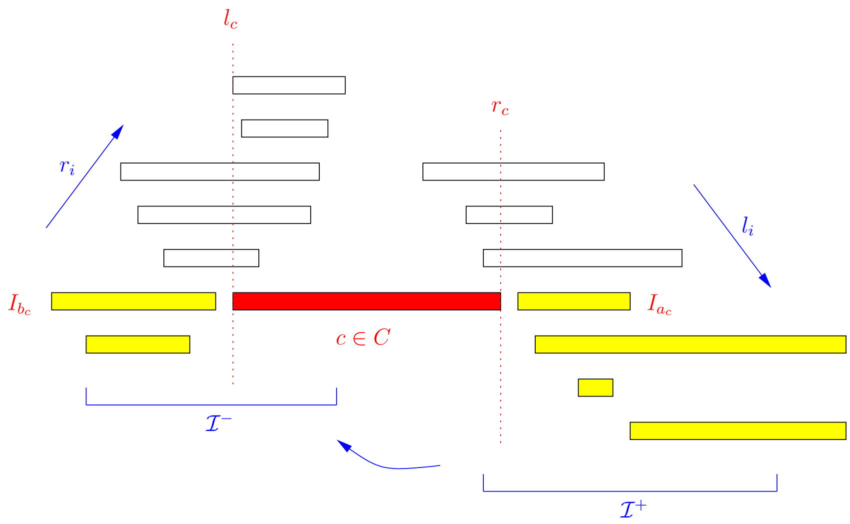

Fig. 2. An illustration of the proof of Lemma 4.

$\diamondsuit$

As we said in preliminaries, we do not need an explicit construction of G_b. We need only to compute the ordering on Y and for each $i \in X$, the two values a_i and b_i. This can be done as follows. First, compute a maximum clique $X = C$ in two different ways: the one, called C^+, is the maximum clique C ordered according to the order $<$, and the other, called C^-, is C ordered according to the order $>$. This can be done in $O(n)$ time [10]. Additionally, we compute $Y = \mathcal{I} \setminus C = \mathcal{I}^+ + \mathcal{I}^-$ ordered as described in Lemma 4. This can be done in $O(n)$ time thanks to the orders $<$ and $>$. Thus, we can determine a_i for all $i \in X$ by using the ordered representation C^- of X. Indeed, by a linear scanning of $\mathcal{I}^+$, we compute each a_i for all $i \in C^-$ (see Fig. 3). In the same way, we can determine b_i for all $i \in X$ by using C^+ and using $\mathcal{I}^-$. Now, a maximum matching in G_b can be computed in $O(n)$ time using the algorithm of G. Steiner and J.S. Yeomans [14].
The procedure Complete_Stables can be easily implemented in linear time (for example by using a data structure which for each $I \in \mathcal{I}$ indicates the stable in $\mathcal{S}$ which I belongs to).

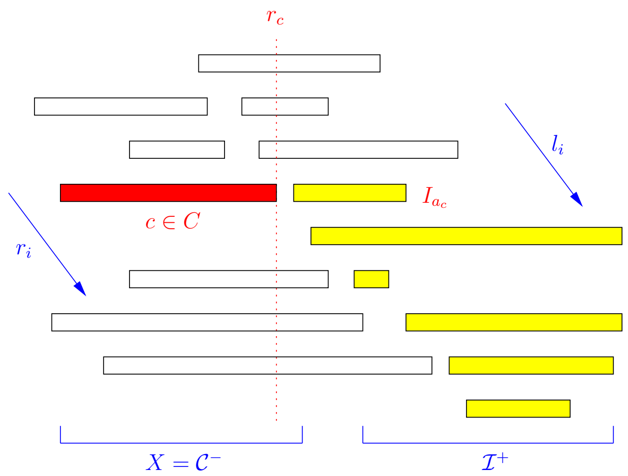

Fig. 3. The determination of a_i for all $i \in X$.

Finally, the Stage 3 can be done in linear time too. Indeed, the proof of Lemma 1 yields a linear-time algorithm to compute a perfect matching $\mathcal{M}$ in $\mathcal{I}$. The overall space used in different stages of the matching algorithm always remains in $O(n)$, therefore we can state the following result.

Theorem 2 *The algorithm Matching_Disjoint_Intervals finds a maximum disjoint matching in a set of intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ in $O(n)$ time and space if we have in input the list of endpoints of the n intervals and the orders $<$ and $>$ on $\mathcal{I}$.*

5 Applications and perspectives

Our interest in developing an efficient algorithm for maximum disjoint matching among intervals comes actually from the study of two scheduling problems. First, we have the practical problem of *working schedules planning* (WSP).

WSP problem:

Let $T_1, \ldots, T_n$ be n tasks such that $T_i = (l_i, r_i)$ with $l_i, r_i \in \mathbb{N}$, the starting and ending dates of T_i. Let $m \in \mathbb{N}$ be the number of employees available and qualified to execute these tasks. Given that the tasks allocated to an employee musnt not overlap and the reglementation imposes no more than $k \in \mathbb{N}$ tasks by employees, are there enough employees to execute all the tasks ?

Then, the following problem arises in scheduling theory: n unit-time jobs must be complete on k processors with the constraint that some jobs cannot be executed at the same time because they share a same ressource. By creating an undirected graph $G = (V, E)$ with a vertex for each of the n jobs and an edge between each pair of conflicting jobs, we can see that a minimum length of schedule corresponds to a partition of V into a minimum number of independent sets of size at most k. In this way, B.S. Baker and E.G. Coffman defined the *mutual exclusion scheduling* (MES) problem as follows: given an undirected graph G and $k \in \mathbb{N}$, find a minimum coloring of G, such that each colour is used at most k times. Clearly, the WSP problem is equivalent to the MES problem for interval graphs. This last one, and so the WSP problem, were proved $\mathcal{NP}$-complete even if k is a constant such that $k \geq 4$ [3] (the complexity for a constant $k = 3$ remains an open question at our knowledge). On the other hand, they can be solved for $k = 2$ by matching technics. Indeed for $k = 2$, these two problems correspond to the problem of finding a maximum disjoint matching in a set of intervals, i.e. the problem of finding a maximum matching (MM) in a co-interval graph.
The difference between WSP, MES and MM relies on the structures used to represent the datas in entry of the problems. Indeed, the input for the WSP problem is the list of endpoints of the n intervals: we can consider that in this case the size of the datas is in $O(n)$. The MES problem has in input a representation of the interval graph $G = (V, E)$ by adjacency lists, which size is in $O(|V| + |E|)$. In the same way, for the MM problem in a co-interval graph $\overline{G} = (V, \overline{E})$, we have some adjacency lists of size $O(|V| + |\overline{E}|)$ in input.

From Theorem 2, we can state the next corollary.

Corollary 1 *The algorithm Matching_Disjoint_Intervals solves:*

(1) the WSP problem with $k = 2$ in $O(n \log n)$ time where n is the number of tasks,

(2) the MES problem for interval graphs with $k = 2$ in linear time,

(3) the problem of maximum matching in co-interval graphs in linear time.

Proof. In the case of the WSP problem, the orders $<$ and $>$ on $\mathcal{I}$ can be computed in $O(n \log n)$ time by some simple sortings. Then, from an interval graph G with n vertices and m edges (resp. a co-interval graph $\overline{G}$ with n vertices and $\overline{m}$ edges), we can compute an interval representation $\mathcal{I}$ of G (resp. $\overline{G}$), i.e. a list of endpoints of the n intervals, with the orders $<$ and $>$ on $\mathcal{I}$ in $O(n+m)$ (resp. $O(n+\overline{m})$) time [11]. Consequently, from Theorem 2 we deduce the three assertions of the corollary.

$\diamondsuit$

Remark. The linear-time algorithm for maximum matchings in convex bipartite graphs of G. Steiner and J.S. Yeomans is complex and not the most practicle. It is possible to keep a time complexity in $O(n \log n)$ in the case (1) by using the much simpler $O(n \log n)$-time algorithm of G. Gallo [7] to compute a maximum matching in G_b. In the same way, we can prefer the F. Glover's algorithm [8], running in $O(\overline{m})$, to compute the Stage 2 in the case (3).

To conclude, we can note that the Theorem 1 is still valid for a class of graphs which englobes the class of co-interval graphs: the co-triangulated graphs, i.e. the graphs whose the complementaries are some triangulated graphs. The triangulated graphs are the graphs which have no cycle of length four without chord as an induced subgraph [9]. They also have many good properties as to be colored in linear time. Consequently, a maximum matching in these graphs can already be computed in $O(\sqrt{n}m)$ thanks to our matching algorithm and by using the practical algorithm for maximum matchings in bipartite graphs of [6] to compute Stage 2. However, it seems to be interesting to study the properties of the arbitrary bipartite graph G_b in this case, in order to design a most efficient algorithm for finding a maximum matching in co-triangulated graphs.

Acknowledgements

We would like to thank Professor Michel Van Caneghem and Professor Victor Chepoi for their precious advice and the firm PROLOGIA - Groupe Air Liquide for its grant.

References

[1] M.G. Andrews, M.J. Atallah, D.Z. Chen, and D.T. Lee. Parallel algorithms for maximum matching in complements of interval graphs and related problems. *Algorithmica*, pages 263–289, 2000.

[2] M.G. Andrews and D.T. Lee. An optimal algorithm for matching in interval graphs. manuscript, 1992.

[3] H.L. Bodlaender and K. Jansen. Restrictions of graph partition problems. Part I. *Theoretical Computer Science*, 148:93–109, 1995.

[4] S.A. Cook and R.A. Reckhow. Time bounded random access machines. *Journal Comput. System Sci.*, 7:354–375, 1973.

[5] J. Edmonds. Maximum matching and a polyedron with 0,1 vertices. *J. Research N.B.S. 69 B*, pages 125–130, 1965.

[6] S. Even and R.E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4:507–518, 1975.

[7] G. Gallo. An $O(n \log n)$ algorithm for the convex bipartite matching problem. *Operation Research Letters*, 3 (1):31–34, 1984.

[8] F. Glover. Maximum matchings in a convex bipartite graph. *Naval Research Logistics Quartely*, 4 (3):313–316, 1967.

[9] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Computer Science and Applied Mathematics. Academic Press, New-York, 1980.

[10] U.I. Gupta, D.T. Lee, and J.Y.T. Leung. Efficient algorithms for interval and circular-arc graphs. *Networks*, 12:459–467, 1982.

[11] M. Habib, R. McConnel, C. Paul, and L. Viennot. Lex-BSF and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234:59–84, 2000.

[12] S. Micali and V.V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *Proc. 21st Annual Symposium on Foundations of Computer Science*, pages 17–27, 1980.

[13] F.S. Roberts. *Graph Theory and its Applications to Problems of Society.* SIAM, Philadelphia, 1978.

[14] G. Steiner and J.S. Yeomans. A linear time algorithm for maximum matchings in convex, bipartite graphs. *Computers Math. Applic.*, 31 (12):91–96, 1996.

10