Efficient algorithms for disjoint matchings among intervals and related problems [⋆]

Frédéric Gardi [⋆⋆]

Laboratoire d'Informatique Fondamentale,
Parc Scientifique et Technologique de Luminy,
Case 901 - 163, Avenue de Luminy,
13288 Marseille Cedex 9, France
Frederic.Gardi@lif.univ-mrs.fr

Abstract. In this note, the problem of determining disjoint matchings in a set of intervals is investigated (two intervals can be matched if they are disjoint). Such problems find applications in schedules planning. First, we propose a new incremental algorithm to compute maximum disjoint matchings among intervals. We show that this algorithm runs in $O(n)$ time if the intervals are given ordered in input. Additionally, a shorter algorithm is given for the case where the intervals are proper. Then, a $\mathcal{NP}$-complete extension of this problem is considered: the perfect disjoint multidimensional matching problem among intervals. A sufficient condition is established for the existence of such a matching. The proof of this result yields a linear-time algorithm to compute it in this case. Besides, a greedy heuristic is shown to solve the problem in linear time for proper intervals.

1 Introduction

A *matching* in an undirected graph $G = (V, E)$ is a subset $\mathcal{M} \subseteq E$ of edges such that no two edges are incident to a same vertex [1, 2]. The matching $\mathcal{M}$ is called *perfect* if every vertex $v \in V$ belongs to $\mathcal{M}$, *ie.* if the cardinality of the matching equals $n/2$. In this way, the *maximum matching problem* is to find the matching of maximum cardinality in a graph. The *perfect matching problem* is to determine the existence of a perfect matching in a graph (and compute it if necessary); this problem is clearly reducible to the maximum matching problem. These problems have been intensively studied in algorithmic graph-theory and combinatorics. They occur in numerous problems of operations research (for example personnel assignment [3], scheduling [4]) and also holds an important place in many practicle applications. The first polynomial algorithm to find a maximum matching in a graph was given by J. Edmonds [5]. The fastest algorithm is due to S. Micali and V.V. Vazirani [6]; its time complexity is $O(\sqrt{n}m)$ given a n-vertex, m-edge graph in input, but it is complex and not considered practical.

⋆ This paper is accepted for presentation and publication at the Fourth International Conference on Discrete Mathematics and Theoretical Computer Science DMTCS'03.

⋆⋆ The author works under contract with the firm Prologia–Groupe Air Liquide.

Definition of the problem. In this paper, a related problem is approached: the *maximum disjoint matching problem among intervals.* Given a set $\mathcal{I} = \{I_1, \ldots, I_n\}$ of n intervals of the real line, the problem is to find a maximum matching in $\mathcal{I}$ such that two intervals can be matched if they are *disjoint* (non-intersecting). An interval I_i is defined with its *left endpoint* $le(I_i)$ (shortly l_i) and its *right endpoint* $re(I_i)$ (shortly r_i). In graph-theoretic terms, such a problem is equivalent to the maximum matching problem in *complements of interval graphs.* An undirected graph G=(V,E) is an *interval graph* if to each vertex $v \in V$ can be associated an interval $I_v = [l_v, r_v]$ of the real line, such that any pair of distinct vertices u, v are connected by an edge of E if and only if $I_u \cap I_v \neq \emptyset$. The family $\{I_v\}_{v \in V}$ is an *interval representation* of G. The edges of the complement graph $\overline{G} = (V, F)$, called *co-interval graph*, are transitively orientable by setting $(u, v) \in \overrightarrow{F}$ if $r_u < l_v$. The orientation $\overrightarrow{F}$ of the edges induces a partial order called interval order (we shall write $I_u \prec I_v$ if $r_u < l_v$). An interval graph G is called *proper interval graph* if there is an interval representation of G such that no interval contains properly another. Interval graphs are used as models in many problems arising in diverse areas like scheduling, genetics, psychology, sociology, archæology and others. The interested reader can consult [7, 2] for surveys.

Previous works and results. At our acquaintance, two algorithms have been proposed for maximum disjoint matching among intervals. The first one appears in an unpublished manuscript of M.G. Andrews and D.T. Lee [8]. This algorithm, based on plane sweeping, runs in $O(n \log n)$ time even if a sorted interval representation is given in input. The second one is in a recent paper by M.G. Andrews *et al.* [9]. They give a parallel recursive algorithm which requires $O(\log^3 n)$ time using $O(n/\log^2 n)$ processors on the EREW PRAM (see [10] for an introduction to the world of parallel algorithms). The serialization of their algorithm provides an $O(n \log n)$ algorithm for computing maximum disjoint matchings among intervals. Moreover, the authors claim that this one runs in linear time if the input intervals are given sorted. However, this algorithm remains recursive and complicated. In Section 3, we propose a much simpler incremental algorithm running in $O(n)$ time and space given the intervals sorted in input. In addition, a shorter $O(n)$ algorithm is designed for the case where the intervals are proper; this one is quite different from the algorithm presented in [9]. According to these results, we establish that the maximum matching problem for a n-vertex, m-edge co-interval graph is solvable in $O(n + m)$ time.

Extensions. A natural extension of the perfect matching problem is the *perfect multidimensional matching problem*: given a n-vertex graph G and a natural number k with n multiple of k, find a partition of G into n/k complete sets of size k if there exists one. The problem for fixed $k = 3$, also known as Exact Cover by Triangles, is $\mathcal{NP}$-complete for general graphs [11]. In Section 4, a related problem is considered for disjoint intervals: the *perfect disjoint k-dimensional matching problem among intervals*, shortly k-PDMI. In this way, the perfect disjoint matching problem is denoted 2-PDMI. In graph-theoretic terms, the k-PDMI problem is equivalent to the perfect k-dimensional matching problem for co-interval graphs. H.L. Bodlaender and K. Jansen [12] have shown that k-PDMI is $\mathcal{NP}$-complete even for fixed $k \geq 4$; the problem for $k = 3$ remains an open question at our knowledge. First, we establish a sufficient condition for the existence of a perfect disjoint k-dimensional matching among arbitrary intervals. As a byproduct of the proof, we obtain a linear-time algorithm to compute the matching in this case. Finally, a greedy heuristic is shown to solve the k-PDMI problem for any integer k when the input intervals are proper.

Applications. Our interest to disjoint matching problems among intervals comes from the following *working schedules planning* problem (WSP), which has actually inspired this research. Let $\{T_i\}_{i=1,\dots,n}$ be a set of tasks having each one a starting date l_i and an ending date r_i. The regulation imposes that an employee cannot execute more than k tasks (n is a multiple of k). Given that the tasks allocated to an employee must not overlap, build a planning requiring the minimum number of employees. Since the tasks are some intervals of the real line, the WSP problem is equivalent to the k-PDMI problem. Thus, the result of Section 3 provides an $O(n \log n)$ algorithm for WSP with $k = 2$. For $k \geq 3$, some easy (polynomial) cases are given in Section 4. Notably, the sufficient condition finds applications in WSP of municipal bus drivers or air terminal personnels (schedules planning problems solved by the firm Prologia–Groupe Air Liquide [13]). Indeed, the movements of buses or planes generates some packets of consecutive tasks which induce independent sets of size larger than k (for reasonnable values of k like $3, 4, 5$).

2 Preliminaries

Before giving the first results, some notations and definitions which shall be useful to the description and analysis of the matching algorithms are detailed. All the terms defined here are essentially derived from graph-theory and can be found in [1, 2].

Let $\mathcal{I} = \{I_1, \dots, I_n\}$ be a set of n intervals. A *complete set* or *clique* is a set of pairwise intersecting intervals. The *clique number* $\omega(\mathcal{I})$ is the cardinality of the largest clique in $\mathcal{I}$. On the opposite, an *independent set* or *stable* is a set of pairwise disjoint intervals. A *coloring* of $\mathcal{I}$ associates to each interval one color in such a way that two intersecting intervals have different colors. In fact, a coloring of $\mathcal{I}$ corresponds to a partition of $\mathcal{I}$ into stables. The *chromatic number* $\chi(\mathcal{I})$ is the cardinality of a partition of $\mathcal{I}$ into the least number of stables.

The structural properties of a set of intervals (or of its corresponding interval graph) are mentionned in [1, 2]. One of the most significant is that for a set of intervals $\mathcal{I}$, the equality $\omega(\mathcal{I}') = \chi(\mathcal{I}')$ holds for all $\mathcal{I}' \subseteq \mathcal{I}$ (C. Berge 1960, *cf.* [2]). Moreover, computing a maximum clique or a minimum coloring of $\mathcal{I}$ can be done in linear time (F. Gavril 1976, *cf.* [2]; see also [14, 15]). The linear orders induced by the endpoints are often used in the algorithmic of the sets of intervals. In further sections, we denote by $\lhd$ the order defined by the ascendant left endpoints ($I_u \lhd I_v$ if $l_u < l_v$ or $l_u = l_v$ and $r_u \leq r_v$). In the same way, we denote by $\rhd$ the order defined by the descendant right endpoints ($I_u \rhd I_v$ if $r_u > r_v$ or $r_u = r_v$ and $l_u \geq l_v$).

Another crucial notion appears in the analysis of one algorithm: the convexity in bipartite graphs. A bipartite graph $G = (X, Y, E)$ is Y-convex if there is an ordering $<$ on Y such that if $ix, iz \in E$ with $i \in X$ and $x, z \in Y$, then $x < z$ implies that $iy \in E$ for all $y \in Y$ with $x < y < z$. A convex bipartite graph G is specified by giving the ordering $<$ 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 nice result of F. Glover [16] establishes that the maximum matching problem for convex bipartite graphs is solvable in linear time. Successive improvements in the efficiency of Glover's algorithm can be found in [17–20].

3 Disjoint matchings among intervals

The matching algorithm. An incremental algorithm is presented to solve the maximum disjoint matching problem among intervals in linear time. Before describing the matching algorithm in details, we outline the main ideas behind its correctness. The following result establishes that the maximum disjoint matching problem in a set $\mathcal{I}$ of intervals can be reduced to the problem of minimizing the number of stables having only one interval in a minimum partition of $\mathcal{I}$ into stables.

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

The proof of this assertion is based on the two following lemmas.

Lemma 1. *Let $S_i = \{I_u\}$ be one of the $s(\mathcal{I})$ stables of $\mathcal{S}$ consisting of only one interval. Then I_u belongs to any maximum clique of $\mathcal{I}$.*

Proof. Since $\omega(\mathcal{I}) = \chi(\mathcal{I})$, every stable of $\mathcal{S}$ has an interval in any maximum clique of $\mathcal{I}$. Thus, if $S_i = \{I_u\}$, then I_u belongs necessarily to any maximum clique of $\mathcal{I}$. □

Lemma 2. *If every stable of $\mathcal{S}$ contains at least two intervals and n is an even integer, then $\mathcal{I}$ admits a perfect disjoint matching.*

Proof. The idea is to show that from two stables S_i and S_j of odd size (at least three), it is always possible to match two intervals, the one in S_i and the other in S_j, in order 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 $I_a \prec I_b$ and $I_c \prec I_d$. If I_a and I_d are disjoint, then they are the desired candidates to be matched. Otherwise, we claim that I_b and I_c make such a pair of intervals. Indeed, I_a intersecting I_d implies that $l_d \leq r_a$. Now, by using the inequalities $r_c < l_d$ and $r_a < l_b$, we have $r_c < l_b$ and also $I_b \cap I_c = \emptyset$. Finally, from the remaining stables of even size, we can trivially match intervals of each stable in pairs. By adding them to the intervals previously matched, we obtain a perfect disjoint matching in $\mathcal{I}$. □

Then, Proposition 1 is proved as follows.

Proof (of Proposition 1). The first lemma imposes that $s(\mathcal{I})$ intervals cannot be matched in $\mathcal{I}$ and also that the size of a maximum disjoint matching in $\mathcal{I}$ is at most $\lfloor (n - s(\mathcal{I}))/2 \rfloor$. Having removed these $s(\mathcal{I})$ intervals, Lemma 2 allows us to compute a perfect disjoint matching among the remaining $n - s(\mathcal{I})$ intervals (minus one if $n - s(\mathcal{I})$ is odd). □

Proposition 1 establishes that determining a maximum disjoint matching in $\mathcal{I}$ is reducible to find a minimum partition of $\mathcal{I}$ into stables such that the number $s(\mathcal{I})$ of stables of size one is minimized. According to Lemma 1, this new problem is solvable by computing a maximum disjoint matching between intervals of a maximum clique C and intervals of $\mathcal{I} \setminus C$. Indeed, having this maximum matching (denoted $\mathcal{M}^b$), Algorithm CompleteStables detailed below minimizes $s(\mathcal{I})$. Then, a maximum disjoint matching in $\mathcal{I}$ is obtained by using the constructive proofs of Proposition 1 and Lemma 2.

```
**Algorithm** CompleteStables;
**input:** $\mathcal{S}$ a minimum partition of $\mathcal{I}$ into stables,
    $\mathcal{M}^b$ a maximum matching between a maximum clique $C$ and $\mathcal{I} \setminus C$;
**output:** $\mathcal{S}$ with a minimum number of stables of size one;
**begin**;
    **while** there exists $S_i = \{I_u\}$ and $\{I_u, I_v\} \in \mathcal{M}^b$ with $I_v \in S_j$ **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**;
```

The validity of Algorithm CompleteStables relies on Lemma 1 and the maximality of the matching $\mathcal{M}^b$. Now we can provide a complete description of our matching algorithm.

```
**Algorithm** MatchDisjIntervals;
**input:** $\mathcal{I} = \{I_1, \ldots, I_n\}$ a set of intervals;
**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** $i = 1, \ldots, \chi(\mathcal{I})$, $|S_i| \leq 2$ **then goto** *stage 3*;
        **if for all** $i = 1, \ldots, \chi(\mathcal{I})$, $|S_i| \geq 2$ **then goto** *stage 3*;
    *stage 2:*
```

```
        compute a maximum clique $C = \{c_1, \ldots, c_{\chi(\mathcal{I})}\}$ in $\mathcal{I}$;
        construct the bipartite graph $G^b = (X, Y, E)$ such that:
            - $X = C$,
            - $Y = \mathcal{I} \setminus C$,
            - $E = \{(I_i, I_j) \mid I_i \in C, I_j \in \mathcal{I} \setminus C$ with $I_i \cap I_j = \emptyset\}$;
        compute a maximum matching $\mathcal{M}^b$ in $G^b$;
        CompleteStables($\mathcal{S}$,$\mathcal{M}^b$);
    stage 3:
        for each $S_i \in \mathcal{S}$ do
            if $|S_i| = 1$ then $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S_i\}$;
        compute a perfect disjoint matching $\mathcal{M}$ in $\mathcal{S}$;
        return $\mathcal{M}$;
end;
```

Complexity of the matching algorithm. In this section, we analyse time and space complexities of the matching algorithm according to the classical RAM computational model [21]. We suppose to have in input the set $\mathcal{I} = \{I_1, \ldots, I_n\}$ of intervals and in addition, the two orders $\lhd$ and $\rhd$ defined on $\mathcal{I}$. Concretely, the data structures used to represent the abstract objects manipulated by the algorithm are defined as follows. $\mathcal{I}$ is an array of size n; the interval I_i, specified with its endpoints l_i, r_i, is the i^{th} element of $\mathcal{I}$. $\lhd$ and $\rhd$ are two arrays of size n containing (the indices of) the intervals of $\mathcal{I}$ in the specified order. $\mathcal{S}$ is an array of size $\chi(\mathcal{I})$; the j^{th} element S_j of $\mathcal{S}$ is an array of size $S_j.size$. For every interval $I_i \in \mathcal{I}$, $I_i.stable$ determines the index of the stable containing it. C is an array of size $\chi(\mathcal{I})$; the j^{th} element c_j of C represents the interval which belongs to the stable S_j in the clique C. $\mathcal{M}^b$ is an array of size $\chi(\mathcal{I})$; for $j = 1, \ldots, \chi(\mathcal{I})$, $\mathcal{M}^b_j$ contains the interval matched to $c_j \in C$ (or $\emptyset$ if it is not matched). $\mathcal{M}$ contains the output pairs of matched intervals; this can be indifferently an array or a list. Then, the time requires to access (in *read* or *write* mode) to one element of these data structures is considered to be $O(1)$.

Having $\lhd$ and $\rhd$, a minimum coloring of n intervals is done in $O(n)$ time [14]; hence, *stage 1* requires $O(n)$ time. The complexity of *stage 2* relies on the following lemma.

Lemma 3. *The bipartite graph $G_b = (X, Y, E)$ is Y-convex.*

Proof (Sketch). We recall that $X = C$, a maximum clique of $\mathcal{I}$ and $Y = \mathcal{I} \setminus C$. The set $\mathcal{I} \setminus C$ is divided into two distinct parts: the set $\mathcal{I}^r$ of intervals which are to the right of the clique C and the set $\mathcal{I}^l$ of intervals which are to its left. By ordering $\mathcal{I}^r$ according to the increasing left endpoints and $\mathcal{I}^l$ according to the increasing right endpoints, we obtain a linear ordering of the set Y. For each $c_j \in C$, set $a_j = \min\{i \mid c_j \prec I_i \ (I_i \in \mathcal{I}^r)\}$ and $b_j = \max\{i \mid I_i \prec c_j \ (I_i \in \mathcal{I}^l)\}$. One can verify that for every $i \in \{a_j, \ldots, b_j\}$, we have $c_j \cap I_i = \emptyset$ (see Fig. 1 in Appendix). Consequently, the bipartite graph G^b is Y-convex. □

Since G_b is convex bipartite, a maximum matching $\mathcal{M}^b$ can be computed in $O(n)$ time by using the algorithm of G. Steiner and J.S. Yeomans [20]. Their algorithm must have in input a construction of G^b such that it is described in the preliminaries: the ordering on Y and for each $c_j \in X$, the two values a_j and b_j. Having $\lhd$ and $\rhd$, order Y such that it was done in the proof of Lemma 3 requires $O(n)$ time. Then, we can determine each a_j by sweeping the set $\mathcal{I}^r$ if the intervals of C are sorted according to the right endpoints; see Procedure Determine_a_i below.

```
Procedure Determine_$a_j$;
input: $C = \{c_1, \dots, c_{\chi(\mathcal{I})}\}$ sorted according to the right endpoints,
       $\mathcal{I}^r = \{I_1^r, \dots, I_{n^r}^r\}$ sorted according to the left endpoints;
output: $a_j$ for each $c_j \in C$;
begin;
    $i \leftarrow 1$;
    for all $j = 1, \dots, \chi(\mathcal{I})$ do
        while $i \leq n^r$ and $le(I_i^r) \leq re(c_j)$ do
            $i \leftarrow i + 1$;
        $a_j \leftarrow i$;
end;
```

The correctness of Procedure Determine_a_j is based on the fact that for each $c_j \in C$, we have $re(c_{j-1}) \leq re(c_j)$ and also $a_{j-1} \leq a_j$. Having the order $\rhd$, sorting C according to the right endpoints is done in $O(n)$ time and in the worst case, the loop runs in $O(|C| + |\mathcal{I}^r|)$ time. Consequently, the execution of the procedure requires $O(n)$ time. Clearly, the b_j's can be determined by a symmetric Procedure Determine_b_j among the set $\mathcal{I}^l$ if the c_j's are ordered according to the left endpoints. Thus, the time complexity to compute each b_j's is still $O(n)$. Finally, the construction and the maximum matching of G^b are computed in $O(n)$ time. To complete the analysis of *stage 2*, we propose an implementation of Algorithm CompleteStables to run in linear time.

```
Procedure CompleteStables;
input: $\mathcal{S}$ the set of stables, $\mathcal{M}_b$ the maximum matching in $G^b$;
output: $\mathcal{S}$ with a minimum number of stables of size one;
begin;
    $\mathcal{S}^{one} \leftarrow \emptyset$;
    for each $S_j \in \mathcal{S}$ do
        if $S_j.size = 1$ then
            $\mathcal{S}^{one} \leftarrow \mathcal{S}^{one} \cup \{S_j\}$;
    while $\mathcal{S}^{one} \neq \emptyset$ do
        $\mathcal{S}^{one} \leftarrow \mathcal{S}^{one} \setminus \{S_j\}$;
        let $I_i$ be the interval contained in $\mathcal{M}_j^b$;
        if $I_i \neq \emptyset$ then
            $j' \leftarrow I_i.stable$;
            $S_{j'} \leftarrow S_{j'} \setminus \{I_i\}$, $S_{j'}.size \leftarrow S_{j'}.size - 1$;
            $S_j \leftarrow S_j \cup \{I_i\}$, $S_j.size \leftarrow S_j.size + 1$;
            if $S_{j'}.size = 1$ then
                $\mathcal{S}^{one} \leftarrow \mathcal{S}^{one} \cup \{S_{j'}\}$;
    end;
```

To conclude, *stage 3* is done in $O(n)$ time too: having removed stables of size one, the proof of Proposition 1 yields a simple linear-time algorithm to compute a perfect matching in $\mathcal{I}$. The space used all along the three stages never exceeding $O(n)$, the following result is established.

Theorem 1. *Algorithm MatchDisjIntervals finds a maximum disjoint matching among n intervals in $O(n)$ time and space given in input the set of intervals and the orders $\lhd$ and $\rhd$.*

Corollary 1. *The maximum matching problem for a n-vertex, m-edge co-interval graph is solvable in $O(n+m)$ time.*

Proof. Computing an ordered interval representation from a co-interval graph is done in $O(n+m)$ time [22]. Then, Theorem 1 allows to conclude. □

Note. The time complexity remains in $O(n+m)$ if the more direct $O(m)$ Glover's algorithm [16] is used to compute a maximum matching in G^b at *stage 2*.

A short algorithm for proper intervals. In [9] a simpler algorithm is presented for the maximum matching problem among proper intervals. This algorithm makes use of a red-blue matching algorithm [23] to compute the size of a maximum matching. Here we propose another short algorithm based on a new characterization of the size of a maximum matching.

Lemma 4. *Let $\mathcal{I}$ be a set of n proper intervals. Then the size $\vartheta(\mathcal{I})$ of a maximum disjoint matching in $\mathcal{I}$ is* $\min(n - \omega(\mathcal{I}), \lfloor n/2 \rfloor)$.

Proof. Let $\mathcal{S}$ be a minimum partition of $\mathcal{I}$ into stables. We recall that the cardinality of $\mathcal{S}$ equals $\omega(\mathcal{I})$. Then, three cases are possible.

Case 1: every stable of $\mathcal{S}$ has a size at least two. This implies that $\lfloor n/2 \rfloor \leq n - \omega(\mathcal{I})$ (n even $\Rightarrow n \geq 2\omega(\mathcal{I})$, n odd $\Rightarrow n > 2\omega(\mathcal{I})$). Then, according to Lemma 2, we have $\vartheta(\mathcal{I}) = \lfloor n/2 \rfloor$.

Case 2: every stable of $\mathcal{S}$ has a size at most two. Clearly, this implies that $n \leq 2\omega(\mathcal{I})$ and also $\lfloor n/2 \rfloor \geq n - \omega(\mathcal{I})$. In this case, $\vartheta(\mathcal{I})$ equals the number of stables of size two, which is $n - \omega(\mathcal{I})$.

Case 3: $\mathcal{S}$ contains stables of size one and stables of size at least three. We claim that in this case we can bring us back to one of the two previous situations by completing small stables with intervals from large stables. To prove this claim, let us consider two stables S_i and S_j respectively of size one and three. Since the intervals are proper, the interval of S_i cannot intersect the three intervals of S_j. Therefore, there exists at least one interval which can be removed in S_j to be added to S_i. By repeating this exchange process while there are in $\mathcal{S}$ a stable of size one and another of size at least three, the claim is proved. Thereby, Cases 1 and 2 allows us to conclude. □

Now the next lemma characterizes a maximum matching in a set of proper intervals.

Lemma 5. *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of proper intervals ordered according to the left endpoints and $\vartheta(\mathcal{I})$ the size of a maximum disjoint matching in $\mathcal{I}$. Then $\mathcal{M} = \{(I_i, I_{n-\vartheta(\mathcal{I})+i}) \mid i = 1, \ldots, \vartheta(\mathcal{I})\}$ is a maximum disjoint matching in $\mathcal{I}$.*

Proof. Suppose that two matched intervals $(I_i, I_{n-\vartheta(\mathcal{I})+i})$ of $\mathcal{M}$ are intersecting (*ie.* $l_{n-\vartheta(\mathcal{I})+i} < r_i$). When the intervals are proper, the right endpoints have the same order as the left endpoints. Consequently, the intervals $I_i, I_{i+1}, \ldots, I_{n-\vartheta(\mathcal{I})+i}$ overlap the portion $[l_{n-\vartheta(\mathcal{I})+i}, r_i]$ of the real line and also induce a clique of cardinality $n - \vartheta(\mathcal{I}) + 1$. Hence, the size of a maximum disjoint matching cannot be larger than $\vartheta(\mathcal{I}) - 1$, which is a contradiction. □

According to Lemmas 4 and 5, one can design the following algorithm for maximum disjoint matching among proper intervals.

```
Algorithm MatchDisjProperIntervals;
input: $\mathcal{I} = \{I_1, \ldots, I_n\}$ a set of ordered proper intervals;
output: $\mathcal{M}$ a maximum disjoint matching in $\mathcal{I}$;
begin;
    compute $\omega(\mathcal{I})$;
    $\vartheta(\mathcal{I}) \leftarrow \min(n - \omega(\mathcal{I}), \lfloor n/2 \rfloor)$;
    $\mathcal{M} \leftarrow \emptyset$;
    for all $i = 1, \ldots, \vartheta(\mathcal{I})$ do
        $\mathcal{M} \leftarrow \mathcal{M} \cup (I_i, I_{n-\vartheta(\mathcal{I})+i})$;
    return $\mathcal{M}$;
end;
```

Since the calculation of $\omega(\mathcal{I})$ is done in $O(n)$ time [14], the algorithm runs in $O(n)$ time too.

Theorem 2. *Algorithm MatchDisjProperIntervals determines a maximum disjoint matching among n proper intervals in $O(n)$ time and space given the set of intervals ordered in input.*

4 Perfect disjoint multidimensional matchings

A sufficient condition for arbitrary intervals. The following proposition gives us a sufficient condition to obtain a perfect disjoint k-dimensional matching among arbitrary intervals. The proposition generalizes Lemma 2.

Proposition 2. *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of intervals and k an integer with n multiple of k. If there exists a coloring of $\mathcal{I}$ such that each color is used at least k times, then $\mathcal{I}$ admits a perfect disjoint k-dimensional matching. Moreover, this matching is computed in $O(n)$ time and space given the set of ordered intervals and the coloring in input.*

The proof relies on the next lemma.

Lemma 6. *Let $S_1, \ldots, S_t$ be t stables of $\mathcal{I}$ satisfying the following conditions:*

. *$t \in \{1, \ldots, k\}$,*
. *for $i = 1, \ldots, t$, $|S_i| = k + r_i$ with $r_i \in \{1, \ldots, k-1\}$,*
. *the sum of the r_i's for $i = 1, \ldots, t$ equals k.*

Then there exists a stable S^* of size k such that for all $i = 1, \ldots, t$, r_i intervals of S^* belong to S_i. In other words, $S_1, \ldots, S_t$ admits a perfect disjoint k-dimensional matching of cardinality $t + 1$.

Proof. An algorithm having the stables $S_1, \ldots, S_t$ in input is proposed for the construction of S^*. The intervals of each stable are supposed to be ordered according to the relation $\prec$. The *rank* of an interval in a stable is its number in this order. In this way, $I_{i,j}$ denote the interval of rank j in the stable i. At each step, the algorithm selects one interval among the t stables, removes it from its stable and includes it in S^*. After k steps, the stable S^* is returned in output. The selection of the interval I_j^* of rank j in S^* is done as follows: choose the interval having the smallest right endpoint among the intervals of rank j, which belong to stables of size still larger than k. The complete algorithm is detailed below.

```
Algorithm $k$-MatchDisjIntervals;
input: $k$ an integer,
        $S_1, \ldots, S_t$ a set of stables satisfying the conditions of Lemma 6;
output: the stable $S^* = \{I_1^*, \ldots, I_k^*\}$;
begin;
    $S^* \leftarrow \emptyset$;
    for all $j = 1, \ldots, k$ do
        $F \leftarrow \emptyset$;
        for all $i = 1, \ldots, t$ do
            if $|S_i| > k$ then
                $F \leftarrow F \cup \{I_{i,j}\}$;
        let $I_j^*$ be the interval having the smallest right endpoint in $F$;
        remove $I_j^*$ from its stable and add it to $S^*$;
    return $S^*$;
end;
```

To conclude, the correctness of the algorithm is established. At each step of the algorithm, an interval is selected (every input stable has more than k

intervals). Therefore, S^* contains exactly k intervals in output. Now, we claim that for all $j = 1, \dots, k-1$, we have $I^*_j \prec I^*_{j+1}$, *ie.* $re(I^*_j) < le(I^*_{j+1})$. Indeed, assume that $I^*_j \equiv I_{u,j}$ and $I^*_{j+1} \equiv I_{v,j+1}$ with $u, v \in \{1, \dots, t\}$. If $u = v$, the claim is proved. Otherwise, suppose that I^*_j and I^*_{j+1} are intersecting. We have $le(I_{v,j+1}) \leq re(I_{u,j})$ and also $re(I_{v,j}) < re(I_{u,j})$. Now, $I_{v,j+1} \in F$ at step $j+1$ implies necessarily $I_{v,j} \in F$ at step j. Then, $I_{u,j} \equiv I^*_j$ is not the interval having the smallest right endpoint in F at step j, which is a contradiction. □

Then, the proposition is proved as follows.

Proof (of Proposition 2). Let $\mathcal{S} = \{S_1, \dots, S_q\}$ be a partition of $\mathcal{I}$ into stables such that for all $i = 1, \dots, q$, we have $|S_i| \geq k$. Define $|S_i| = \alpha_i k + \beta_i$ to be the size of the stable S_i with α_i a non-zero integer and $\beta_i \in \{0, \dots, k-1\}$. First, from each stable S_i are extracted $\alpha_i - 1$ stables of size k, plus one if $\beta_i = 0$. After this preprocessing, at most $2k-1$ intervals remain in each stable. Then, Lemma 6 is applied with $t = k$ to extract stables of size k while at least k stables of size strictly greater than k exist in the partition $\mathcal{S}$. When it remains less than k such stables in $\mathcal{S}$, a last application of Lemma 6 allows to conclude (n is a multiple of k). The execution of Algorithm k-MatchDisjIntervals requiring $k\ O(t)$ time, the method described here runs in $k\ O(n/k) = O(n)$ time (given the intervals ordered according to $\prec$ in each input stable). □

Corollary 2. *Let $\mathcal{I}$ be a set of n intervals and k an integer with n multiple of k. If $\mathcal{I}$ admits a perfect disjoint k-dimensional matching, then $\mathcal{I}$ admits a perfect disjoint k'-dimensional matching for any integer $k' < k$ with n multiple of k'.*

A linear-time algorithm for proper intervals. In this last part, the disjoint multidimensional matching problem is proved to be linear-time solvable for proper intervals. At the same time, a strong sufficient condition is established for the existence of disjoint matchings among proper intervals. The result extends Theorem 2.

```
Algorithm $k$-MatchDisjProperIntervals;
input: $\mathcal{I} = \{I_1, \dots, I_n\}$ a set of ordered proper intervals,
       $k$ an integer with $n$ multiple of $k$;
output: $\mathcal{M}$ a perfect $k$-dimensional disjoint matching in $\mathcal{I}$;
begin;
    compute $\omega(\mathcal{I})$;
    $\mathcal{M} \leftarrow \emptyset$;
    if $n/k \geq \omega(\mathcal{I})$ then
        for all $i = 1, \dots, n/k$ do
            $\mathcal{M} \leftarrow \mathcal{M} \cup (I_{i+j(n/k)} \mid j = 0, \dots, k-1)$;
    return $\mathcal{M}$;
end;
```

Theorem 3. *Algorithm k-MatchDisjProperIntervals solves the disjoint k-dimensional matching problem among n proper intervals in $O(n)$ time and space given the set of intervals ordered in input.*

Proof. The result could be established by extending the proof of Lemma 4 and using Proposition 2, but here we give a more direct one. Immediately, if $n/k < \omega(\mathcal{I})$, no perfect disjoint k-dimensional matching exists in $\mathcal{I}$. Otherwise, we claim that the algorithm finds such a matching. Indeed, suppose that two intervals $I_{i+j(n/k)}$ and $I_{i+(j+1)(n/k)}$ (of the k-dimensional matching i) are intersecting for any $j \in \{0, \ldots, k-2\}$. Since the intervals are proper, the intervals $I_{i+j(n/k)}, I_{i+j(n/k)+1}, \ldots, I_{i+(j+1)(n/k)}$ overlap the portion $[l_{i+(j+1)(n/k)}, r_{i+j(n/k)}]$ of the real line and also induce a clique of size $n/k+1$. Such a clique imposing that $\omega(\mathcal{I}) > n/k$, we obtain a contradiction. □

Corollary 3. *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of proper intervals and k an integer with n multiple of k. Then $\mathcal{I}$ admits a perfect k-dimensional disjoint matching if and only if $n/k \geq \omega(\mathcal{I})$.*

Acknowledgements. We thank Professors Michel Van Caneghem and Victor Chepoi for their advice and the firm Prologia–Groupe Air Liquide for its grant. We are also grateful to the two anonymous referees for their appreciations.

References

1. C. Berge (1985). *Graphs.* Elsevier Science Publishers B.V., Amsterdam, 2nd edition.
2. M.C. Golumbic (1980). *Algorithmic Graph Theory and Perfect Graphs.* Computer Science and Applied Mathematics. Academic Press, New-York.
3. P. Ramanan, J. Deogun and C. Liu (1984). A personnel assignment problem. *Journal of Algorithms* 5, 132-144.
4. G. Steiner and J.S. Yeomans (1993). Level schedules for mixed-model, just-in-time processes. *Management Science* 39(6), 728–735.
5. J. Edmonds (1965). Maximum matching and a polyedron with 0,1 vertices. *Journal of Research of N.B.S. B* 69, 125–130.
6. S. Micali and V.V. Vazirani (1980). 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.
7. F.S. Roberts (1978). *Graph Theory and its Applications to Problems of Society.* SIAM, Philadelphia, PA.
8. M.G. Andrews and D.T. Lee (1992). An optimal algorithm for matching in interval graphs. manuscript.
9. M.G. Andrews, M.J. Atallah, D.Z. Chen and D.T. Lee (2000). Parallel algorithms for maximum matching in complements of interval graphs and related problems. *Algorithmica* 26, 263–289.
10. J. Jàjà (1992). *An Introduction to Parallel Algorithms.* Addison-Wesley, Reading, MA.
11. M.R. Garey and J.S. Johnson (1979). *Computer and Intractability: A Guide to $\mathcal{NP}$-Completeness.* W.H. Freeman.
12. H.L. Bodlaender and K. Jansen (1995). Restrictions of graph partition problems. Part I. *Theoretical Computer Science* 148, 93–109.
13. Bamboo–Planification by Prologia–Groupe Air Liquide. http://prologianet.univ-mrs.fr/bamboo/bamboo_planification.html

14. U.I. Gupta, D.T. Lee and J.Y.-T. Leung (1982). Efficient algorithms for interval and circular-arc graphs. *Networks* 12, 459–467.
15. S. Olariu (1991). An optimal greedy heuristic to color interval graphs. *Information Processing Letters* 37, 21–25.
16. F. Glover (1967). Maximum matchings in a convex bipartite graph. *Naval Research Logistics Quartely* 4(3), 313–316.
17. W. Lipski, Jr. and F.P. Preparata (1981). Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. *Acta Informatica* 15, 329–346.
18. G. Gallo (1984). An $O(n \log n)$ algorithm for the convex bipartite matching problem. *Operation Research Letters* 3(1), 31–34.
19. H.N. Gabow and R.E. Tarjan (1985). An linear-time algorithm for the special set union. *Journal of Computer and System Sciences* 30, 209–221.
20. G. Steiner and J.S. Yeomans (1996). A linear time algorithm for maximum matchings in convex, bipartite graphs. *Computers and Mathematics with Applications* 31(12), 91–96.
21. S.A. Cook and R.A. Reckhow (1973). Time bounded random access machines. *Journal of Computer and System Sciences* 7, 354–375.
22. M. Habib, R. McConnel, C. Paul and L. Viennot (2000). Lex-BSF and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science* 234, 59–84.
23. S.K. Kim (1989). Optimal parallel algorithms on sorted intervals. In *Proc. 27th Annual Allerton Conference on Communication, Control and Computing*, pages 766–775. Monticello, IL.

Appendix

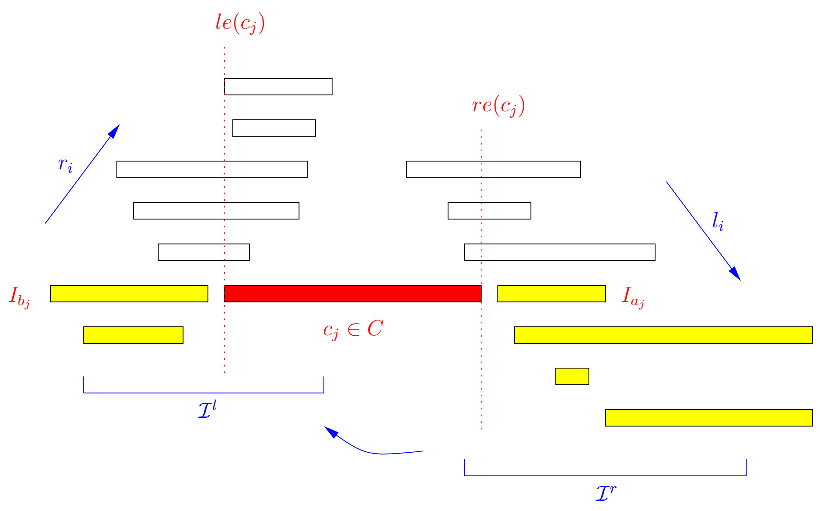

Fig. 1. The proof of Lemma 3