Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 493–500

DOI: 10.15439/2014F50
ACSIS, Vol. 2

Exact and Approximation Algorithms for Linear Arrangement Problems

Alain Quilliot
LIMOS CNRS UMR 6158
LABEX IMOBS3
Université Blaise Pascal
Bat ISIMA, BP 10125
Campus des Cézaux,
63173 Aubière, France
Email: alain.quilliot@isima.fr

Djamal Rebaine
UQAC
Département d'Informatique
Chicoutimi, Saguenay, Quebec
Canada
Email: Djamal.Rebaine@uqac.ca

***Abstract*—We present here new results and algorithms for the *Linear Arrangement* Problem (LAP). We first propose a new lower bound, which links LAP with the *Max Cut* Problem, and derive a LIP model as well as a branch/bound algorithm for the general case. Then we focus on the case of interval graphs: we first show that our lower bound is tight for unit interval graphs, and derive an efficient polynomial time approximation algorithm for general interval graphs.**

I. Introduction

Let G = (X, E) be a non oriented graph where X and E respectively denote the vertices and edges of G. The *Linear Arrangement* Problem (LAP) consists of finding a one-to-one mapping ϕ from X to {1, ... , |X|}that minimizes:

$$f(G,\varphi)=\sum_{(x,y)\in E}|\varphi(y)-\varphi(x)|$$

The LAP problem has applications (see [3, 11]) in Information Retrieval and Industrial Storage, and may also appear as a sub-problem of some Network Design models (see [4]). It is, even in practice, a very difficult combinatorial optimization problem. The corresponding decision LAP was first shown to be *NP*-complete for arbitrary graphs (see for example [8, 9]) and next, for interval graphs [5] and bipartite graphs [9]. However, polynomial time algorithms were designed for trees [4], unit interval graphs [6], paths, cycles, complete bipartite graphs, grid graphs [11] and restricted series-parallel graphs [1]. A survey is available in [3].

Since LAP is *NP*-hard, even for graphs which usually turn most difficult problems into time-polynomial ones, the heuristic approach is therefore justified to deal with it. So the goal of this theory oriented study is to provide tools for the design of exact and approximation LAP algorithms with some focus on interval graphs. The paper is organized as follows. Section 2 introduces an linear ordering based reformulation of LAP. In Section 3, we propose a general lower bound, which links LAP with the well-known *Max Cut* Problem, and next derive, in Section 4 and 5, an ILP model together with a branch/bound algorithm. Finally, in section 6 we restrict our study to the case of interval graphs and propose an approximation algorithm, whose efficiency is briefly tested in Section 7.

II. Notations, definitions, LAP reformulation

A simple (non oriented) graph with no loop is denoted by G = (X, E): X (E) is the node (edge) set of G. We denote by (x, y) = (y, x) an edge with end-nodes x and y in X. If A ⊆ X, then G_A is the sub-graph induced by A from G. If x ∈ X, then $\Gamma_G(x)$ = {y ∈ X such that (x, y) ∈ E} is the *neighbour* set of x. The complementary graph G^c = (X, E^c) of G is defined by: E^c = {(x, y) such that (x, y) ¿ E and x ≠ y}. A *triangle* of G is a clique with 3 nodes. An *anti-edge* is a pair e = (x, y) = (y, x), x ≠ y, such that (x, y) ∉ E. A *fork* with *root* x is any (non oriented) triple f = {x, y, z} = {x, z, y} such that (x, y), (x, z) ∈ E, and (y, z) ∉ E. An *anti-fork* with *root* z is any triple f = {x, y, z} = {y, x, z} such that (x, y) ∈ E and (x, z), (y, z) ∉ E.

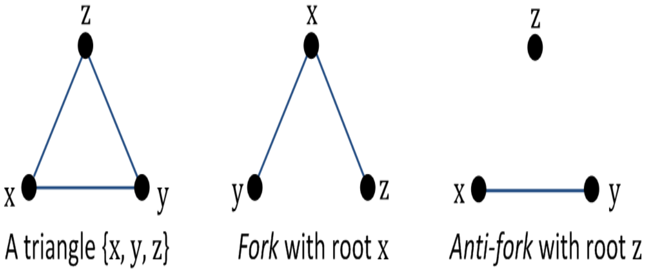


Figure 1: Triangles, Forks and Anti-Forks

LAP Reformulation: A *linear ordering* of a set X is a binary order relation σ such that, for any pair x, y in X, x ≠ y, we have either x σ y or y σ x. Given a graph G = (X, E) and a linear ordering σ of X. For any edge e = (x, y), we set BE(e, z, σ) = (*elementary break* of e by z according to σ) = 1 if x σ z σ y or y σ z σ x, and 0 otherwise. We set BG(G, σ) = (*Global Break* of G according to σ) = $\Sigma_{e,z}$ BE(e, z, σ). If ϕ(σ) is the one-to-one mapping from X into {1, ...,|X|} which derives from σ, then: f(G, ϕ(σ)) = $\Sigma_{(x,y)\in E}$ |ϕ(y) − ϕ(x)| = BG(G, σ) + |E| = $\Sigma_{e,z}$ BE(e, z, σ) + |E|.

978-83-60810-58-3/$25.00 © 2014, IEEE

So, solving LAP means seeking σ that minimizes the *Global Break* BG(G, σ): we denote by LAP(G) the related optimal value Inf_{σ} BE(G, σ).

III. Linking Max Cut and LAP: a general lower bound

Computing a good linear ordering σ of the vertices of a graph G = (X, E) means efficiently deciding, for any $x \in X$, which vertices of the *neighbour* set $\Gamma_G(x)$ are located before x according to σ. This local decision process may be linked to the well-known *Max Cut* Problem [2,7,10]:

> ***Max Cut* Problem**: Let H = (Z, F) be a simple graph. We denote by $Z = A \cup^{Ex} B$ any partition of Z into 2 disjoint subsets, and *Cut*(A, B) the number of edges of H with one extremity into A and the other into B. Solving the *Max Cut* Problem means seeking a *partition* $Z = A \cup^{Ex} B$ that maximizes *Cut*(A, B). We denote by *Max-Cut*(H) the related optimal value.

Let us consider now some graph G = (X, E). We denote by *Tr*(G) the number of triangles of G. For any vertex x in X, we set:

- H(x) = the complementary graph of the sub-graph of G which is induced by $\Gamma_G(x)$. Note that x is not a node of the graph H(x), since G has no loop;
- m(x) = the number of edges of H(x); V(x) = m(x) – *Max-Cut*(H(x)).

Theorem 1: *For any graph G = (X, E), we have: LAP(G) ≥ Tr(G) +* Σ_x *V(x).*

Proof: Let us consider a linear ordering σ of G, and set:

- Fk(G, σ) = number of *forks* f = {x, y, z}, f with root x, of G, such that $((x\ \sigma\ y) \wedge (x\ \sigma\ z)) \vee ((y\ \sigma\ x \wedge z\ \sigma\ x))$.
- AFk(G, σ) = number of *anti-forks* f = {x, y, z} of G, f with root z, such that $(x\ \sigma\ z\ \sigma\ y) \vee (y\ \sigma\ z\ \sigma\ x)$.

Let us first check that: BG(G, σ) = *Tr*(G) + Fk(G, σ) + AFk(G, σ). (E1)

In order to do so, we consider an edge e = (x, y), and a node z, different from x and y. While counting BE(e, z, σ), we consider three cases:

Case 1: x, y and z define a triangle. Then BE(e, z, σ) = 1 if either x σ z σ y or y σ z σ x. In such a case no quantity BE((x, z), y, σ), BE((z, y), x, σ) is equal to 1. So, if x, y, z define a triangle, there exists exactly one node t in{x, y, z} such that BE(e(t), t, σ) = 1, where e(t) is the edge which is defined by {x, y, z} – t. We get $\Sigma_{e = (x, y),\ z \text{ such that } (x, y, z) \text{ is a triangle}}$ BE(e, z, σ) = *Tr*(G).

Case 2: f = {x, y, z} is a *fork* with root x. Then BE((x, y), z, σ) = 1 if either x σ z σ y or y σ z σ x, and then BE((x, z), y, σ) = 0. Conversely, BE((x, z), y, σ) = 1 if either x σ y σ z or z σ y σ x, and then BE((x, y), z, σ) = 0. So, (x, y, z) yields an elementary break iff y and z are located on the same side of x according to σ. Then:

$$\Sigma_{e = (x, y),\ z \text{ adjacent to exactly 1 extremity of } e} \text{BE}(e, z, \sigma) =$$
$$\Sigma_x\ \Sigma_{y,z\ \in\ \Gamma G(x),\ (y, x) \notin E,\ y\ ,\ z \text{ located the same way with respect to } x,\ \sigma}\ 1$$
$$= \text{Fk}(G, \sigma).$$

Case 3: f = {x, y, z} is an *Anti-Fork* with root z. Therefore, we have:

$$\Sigma_{e = (x, y),\ z \text{ such that } (x, z)\ \notin E \text{ and } (y, z) \notin E} \text{BE}(e, z, \sigma) = \text{AFk}(G, \sigma).$$

We get (E1) from the relation: $\Sigma_{e,\ z}$ BE(e, z, σ) = $\Sigma_{e = (x, y),\ z \text{ such that } (x, y, z) \text{ is a triangle}}$ + $\Sigma_{e = (x, y),\ z \text{ adjacent to 1 extremity of } e}$ BE(e, z, σ) + $\Sigma_{e = (x, y),\ z \text{ adjacent to no extremity of } e}$ BE(e, z, σ).

For any $x \in X$, a feasible solution $A(x,\sigma) \cup^{Ex} B(x, \sigma)$ of *Max Cut* is defined on H(x), by setting: $A(x,\sigma) = \{y \in \Gamma_G(x)$, such that y σ x}; $B(x, \sigma) = \{y \in \Gamma_G(x)$, such that x σ y}. Its value, in *Max Cut* sense, is: m(x) - $\Sigma_{y,z\ \in\ \Gamma G(x),\ (y, x) \notin E,\ y,\ z \text{ located the same way with respect to } x\ ,\ \sigma}$ **1** ≤ *Max-Cut*(H(x)). It follows that, for any $x \in X$, $\Sigma_{y,z\ \in\ \Gamma G(x),\ (y, x) \notin E,\ y \text{ and } z, \text{ located the same way with respect to } x,\ \sigma}$ **1** ≥ V(x). Then we get: Fk(G, σ) = $\Sigma_x\ \Sigma_{y,z\ \in\ \Gamma G(x),\ (y, x) \notin E,\ y,\ z \text{ located the same way with respect to } x\ ,\sigma}$ **1** ≥ Σ_x V(x). We conclude. □

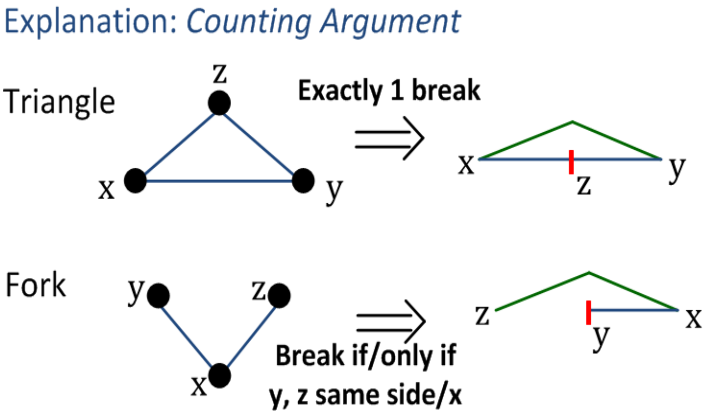


Figure 2: Theorem 1 Counting Argument

Remarks: Small experiments make appear that this bound is often tight, specifically in the case of *chordal* graphs. Still, the *Max Cut* Problem, which has been extensively studied, is *NP*-Hard [9]. So, one may ask about the practical use of the above lower bound. The answer is two-sided:

1. Even though *Max Cut* is NP-Hard, it may be considered as easier to handle than LAP: it admits a natural quadratic {0,1} formulation ([4, 11]) and instances related to the H(x), x ∈ X, are smaller than for original LAP.
2. We shall see in Section 6 that, in the case of interval graphs, our bounding scheme gives rise to an efficient polynomial time approximation scheme.

IV. Quadratic and linear models for LAP

The counting argument of Theorem 1 may also be used in order to derive a quadratic model for LAP. For every pair of nodes (x, y), we need to decide whether y is located right or left in relation to x according to the linear ordering σ. We express this information through a {0, 1} valued W = ($W_{x,y}$, x ≠ y) whose semantics is that: $W_{x,y}$ = 1 (0) ⇔ y located on the right (left) side of x according to linear ordering σ. Counting argument of Theorem 1 tells us that the number of elementary breaks induced by some linear ordering σ is a sum of:

- the number of triangles, which does not depend on linear ordering σ;
- the number of fork {x, y, z}, which are such that y and z are located on the same side with respect to root x according to σ ; (Q1)
- the number of anti-fork {x, y, z}, which are such that x and y are located on different side with respect to root z according to σ. (Q2)

In order to deal with quantity Q1, we introduce a {0, 1} valued vector U= ($U_{x,y}$, (x, y) edge ∈ E)), whose semantics are: $U_{x,y}$ = 1 (0) ⇔ y located on the right (left) side of x. In order to deal with quantity Q2, we introduce a {0, 1} valued vector V= ($V_{x,y}$, (x, y) anti-edge ∉ E)), whose semantics are: $V_{x,y}$ = 1 (0) ⇔ y located on the right (left) side of x. While it is easy to state the constraints which must be satisfied by U, V, W in order to make them define a consistent linear ordering σ, we see that Q1 becomes equal to:

(Number of *forks* with root x) –

$\Sigma_x \Sigma_{z,\, y \text{ such that } (y,\, z) \in E^c} U_{x,y}.(1 - U_{x,z})$.

Also Q2 becomes equal to:

$\Sigma_z \Sigma_{x,\, y \text{ such that } (x,\, y) \in E} V_{z,x}.(1 - V_{z,y})$.

We deduce the following quadratic LAP model:

A Quadratic Linear Formulation of LAP.

- **Variables**
  - $U_{x,y}$, x, y such that (x, y) edge ∈ E: $U_{x,y}$ = 1 (0) ⇔ y located to the right (left) of x
  - $V_{x,y}$, x, y such that (x, y) anti-edge ∈ E^c: $V_{x,y}$ = 1 (0) ⇔ y located to the right (left) of x
  - $W_{x,y}$, x ≠ y: $W_{x,y}$ = 1 (0) ⇔ y located to the right (left) of x
- **Constraints** *(Consistency)*
  - For any x, y, $W_{x,y} + W_{y,x} = 1$
  - For any edge (x, y) ∈ E, $U_{x,y} = W_{x,y}$
  - For any anti-edge (x, y) ∈ E^c, $V_{x,y} = W_{x,y}$
  - For any x, y, z, all distincts, $W_{x,y} + W_{y,z} \geq W_{x,z}$
  - For any x, y, z, all distincts, $(1 - W_{x,y}) + (1 - W_{y,z}) \geq (1 - W_{x,z})$
- **Minimize**

  $\Sigma_z \Sigma_{(x,\, y) \in E} V_{z,y}.(1 - V_{z,x})$ –

  $\Sigma_x \Sigma_{(y,\, z) \in E^c} U_{x,z}.(1 - U_{x,y})$

If we denote by *Fork*(G) the number of forks of the graph G, we easily get:

Theorem 2 : *The optimal value of this quadratic {0,1} program is equal to LAP(G) – Tr(G) – Fork(G).*

This quadratic {0, 1} model may be easily turned into a linear one by introducing additional vectors S and T as follows:

- S = (S_f, f = (x, y, z), fork with root x) subject to: $S_f \leq (1 - U_{x,z})$ and $S_f \leq U_{x,y}$; we consider here that forks are oriented, that means that (x, y, z) and (x, z, y) define 2 distinct forks with root x;
- T = (T_f, g = (x, y, z), anti-fork with root z) T_g subject to: $T_g \geq (1 - V_{z,x})$ and $T_g \geq V_{z,y}$; we consider here that anti-forks are oriented, that means that (x, y, z) and (y, x, z) define 2 distinct anti-forks with root z.

Then minimizing the quadratic quantity $\Sigma_z \Sigma_{(x,\, y) \in E} V_{z,x}.(1 - V_{z,x}) - \Sigma_x \Sigma_{(y,\, z) \in E^c} U_{x,y}.(1 - U_{x,z})$ means minimizing the linear quantity $\Sigma_g T_g - \Sigma_f S_f$.

V. A BRANCH/BOUND ALGORITHM FOR LAP

We may derive from previous section an exact Branch/Bound method for LAP:

- *Branching* is performed by picking up some pair (x, y) of nodes and considering the two alternatives x σ y and y σ x according to linear ordering σ; Any sequence of such decisions may be extended through transitivity into a partial ordering of the node set X;

- *Bounding* is performed through integer linear programming, while extending Theorem 1 in a natural way: if σ is a partial ordering of the node set X obtained as above, we may set, for any node x:

  - *Max-Cut*$_\sigma$(H(x)) = Optimal value of the *Max-Cut* instance which is defined on the graph H(x), augmented with the following constraints:
    - If y such that (x, y) ∈ E is also such that y σ x , then y must be on the subset A of the partition $\Gamma_G(x) = A \cup^{Ex} B$;
    - If z such that (x, z) ∈ E is also such that x σ z , then z must be on the subset B of the partition $\Gamma_G(x) = A \cup^{Ex} B$;

  - $V_\sigma(x)$ = m(x) - *Max-Cut*$_\sigma$(H(x));

  - $W_\sigma(x) = \mathrm{Inf}_{\text{partitions } A \cup Ex\ B \text{ of } X - \{x\} - \Gamma G(x)}$ Card({(x, y) ∈ E, with x, y such that x ∈ A, y ∈ B})

  Then we see that the quantity *Tr*(*G*) + $\Sigma_x V_\sigma(x) + \Sigma_x W_\sigma(x)$ provides us with a lower bound for the best (in LAP sense) linear extension of σ.

- Branching strategy comes in a natural way: we give priority to pairs (x, y) which define edges of the graph G, and choose them in such a way the difference between the best alternative and the worst one is the largest possible.

VI. THE CASE OF INTERVAL GRAPHS: A RESTRICTED VERSION OF LAP

We first introduce additional definitions related to interval graphs:

- A simple graph with no loop G = (X, E) is an *interval graph* if it is the intersection graph of a set [o(x), d(x)], x ∈ X, of closed intervals of the real line. Those intervals may be chosen such that points o(x), d(x), x ∈ X, are distinct. We assume this hypothesis to be always satisfied. We set:
  - x ⊂ y if o(x) < o(y) and d(y) < d(x);
  - x << y if d(x) < o(y);
  - x *Ov* y if o(x) < o(y) < d(x) < d(x).

- In case X is an interval family with distinct endpoints, we say that a linear ordering σ of X is (*Ov*, <<)-*consistent* if it is consistent with both orderings *Ov* and <<. We denote by σ-*can* the *canonical linear ordering,* which is defined as follows: x σ-*can* y if, and only if, o(x) < o(y).

- Then, we say that a *fork* f = {x, y, z} with root x of such an interval graph G = (X, E) is a *strong fork* if there exists t ∈ {y, z} such that t ⊂ x, and that a triangle (x, y, z) is a *strong triangle* if at least some node is contained into another one (for instance z ⊂ x).

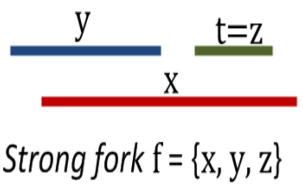


Figure 3: ***Strong fork* f = {x, y, z}**

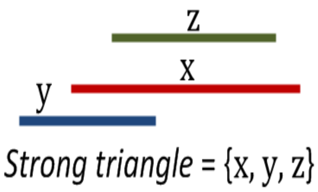


Figure 4: ***Strong triangle* = {x, y, z}**

- We say that G is a *Unit Interval graph* if intervals [o(x), d(x)], x ∈ X may be chosen in such way that no pair x, y exists such that x ⊂ y.

- We finally say that a subset Y of X is a *Left-(Ov, <<)-Section* (*Right-(Ov, <<)-Section*) if, for any x, y ∈ X such that x ∈ Y and (y Ov x) ∨ (y << x), then we also have y ∈ Y (x ∈ Y).

A. A Direct Application of Theorem 1 to Unit Interval Graphs

In the case of unit interval graphs, Theorem 1 allows us to state:

Theorem 2: *If G = (X, E) is a unit interval graph, then σ-can is an optimal solution of LAP.*

Proof: Let us suppose that an *elementary break* (e = (x,y), z, σ-*can*) exists, and that x *Ov* y, which implies that x σ-*can* y. If x << z then y σ-*can* z and z does not break e. Similarly, if z << x then z << x and z does not break e. It comes that x ∩ z is not empty. By the same way, y ∩ z is not empty and{x, y, z} is a *triangle*. So, there is a one-to-one correspondence between *triangles* and *elementary breaks*. So, BG(G, σ-*can*) = *Tr*(G), and we conclude. □

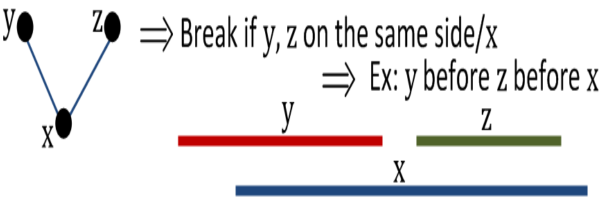


Figure 5: Theorem 2 Argument

B. An Approximation Result.

In the case of general interval graphs, σ-*can* may not be optimal. As a matter of fact, optimal solution may even not be (*Ov*, <<)-*consistent*:

⇒ LAP(G) = 11, optimal σ-*opt* such that y σ-*opt* z σ-*opt* x, while BG(G, σ-*can*) = 14

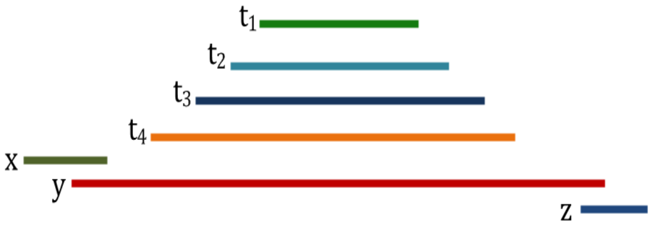


Figure 6: non consistency of (*Ov*, <<)

Still, what can be easily checked is that σ-*can* produces a 2-approximation if we refer to the standard definition of LAP:

Theorem 3: *Given an interval graph G = (X, E) with m edges. Then the following inequality holds:*

$$BG(G, \sigma\text{-}can) \leq 2LAP(G)+m.$$

Proof: A *Global Break* oriented proof comes by induction on the cardinality of X. Let x_0 be the first (smallest) element of X according to σ-*can*, and σ-*opt* some optimal solution of LAP. Induction tells us that: (E2)

$$\Sigma_{e,\, z \neq x0}\ BE(e, z, \sigma\text{-}can) \leq$$
$$m - |\Gamma_G(x_0)| + 2.\ \Sigma_{e,\, z \neq x0}\ BE(e, z, \sigma\text{-}opt).$$

Since all vertices of $\Gamma_G(x_0) \cup \{x_0\}$ are consecutive according to σ-*can*, we get that:

$$\Sigma_{x \in \Gamma G(x0),\, z \in X}\ BE((x_0, x), z, \sigma\text{-}can) =$$
$$\Sigma_{x,\, z \in \Gamma G(x0)}\ BE((x_0, x), z, \sigma\text{-}can)$$
$$= |\Gamma_G(x_0)| \times (|\Gamma_G(x_0)| - 1)/2.$$

On the other hand, if we refer to σ-*opt*, we get: (E3)

$$\Sigma_{x \in \Gamma G(x0),\, z \in X}\ BE((x_0,x), z, \sigma\text{-}opt) \geq$$
$$\Sigma_{x,\, z \in \Gamma G(x0)}\ BE((x_0,x), z, \sigma\text{-}opt) \geq$$
$$\lfloor|\Gamma_G(x_0)|/2\rfloor(\lfloor\Gamma_G(x_0)\rfloor-1)/2+\lceil|\Gamma_G(x_0)|/2\rceil(\lceil|\Gamma_G(x_0)|/2\rceil-1)/2.$$

We derive the result by combining (E2) and (E3). □

C. A Restricted Version CLAP of LAP.

However, experiments will show that best linear orderings are most often (*Ov*, <<)-*consistent*. So we are going to study the following restriction CLAP of LAP:

***(Ov*, <<)-*Consistent Linear Arrangement* Problem (*CLAP*)**: {Compute a (*Ov*, <<)-consistent linear ordering σ which minimizes BG(G, σ)}.

The following lemma bridges CLAP with Theorem 1.

Lemma 1: *In the case the linear ordering σ is (Ov, <<)-consistent, we have that: BG(G, σ) = Tr(G) + SFk(G, σ), where SFk(G, σ) is the number of strong forks f = (x, y, z), x = Root(f) such that $((x\ \sigma y) \wedge (x\ \sigma z)) \vee (y\ \sigma x) \wedge (z\ \sigma x))$.*

Proof: left to the reader (same proof as for Theorem 1). □

Extending Theorem 1 to CLAP leads us to introduce a specific version of *Max-Cut*:

(Ov, <<)-Consistent Unit Cost Max-Cut Problem (C-Max-Cut): Given a graph H = (Z, F), which is the complementary graph of an interval graph $H^c = (Z, F^c)$, and two disjoint subsets A_0 and B_0 of Z, such that:

- A_0 (B_0) is a *Left-(Ov, <<)-Section* (*Right-(Ov, <<)-Section*) of H^c;
- Both A_0 and B_0 define complete sub-graphs of $H^c = (Z, F^c)$.

Compute a partition $Z = A \cup^{Ex} B$, such that:

1. A contains A_0 and is a *Left-(Ov, <<)-Section* of H^c;
2. B contains B_0 and is a *Right-(Ov, <<)-Section* of H^c;
3. the number of edges of H which connect A and B = $|\{(x, y) \in E, x \in A, y \in B\}|$ is the largest possible;
4. A is maximal for the set inclusion order, provided 1, 2, 3 are satisfied.

We denote by *C-Max-Cut*(H, A_0, B_0) the related optimal value. Then we set, for the interval graph G = (X, E) and for any vertex x in X:

- $\Gamma^{Ov,\subset}{}_G(x)$ = (*Ov*, ⊂)-*neighbour* set of x = $\{y \in \Gamma_G(x), y \neq x$, such that $(y \subset x)$ or (y *Ov* x) or (x Ov y)};
- H(x) = complementary graph of the sub-graph induced by $\Gamma^{Ov,\subset}{}_G(x)$;
- $A_0(x)$ = {y ∈ Z such that y *Ov* x}; $B_0(x)$ = {y ∈ Z such that x *Ov* y};

- m(x) = number of edges of H(x); CV(x) = m(x) - *C-Max-Cut*(H(x), A_0(x), B_0(x)).

Then we get:

Lemma 2: *CLAP(G) ≥ Tr(G) +* Σ_x *CV(x).*

Proof: For every x ∈ X, we set E*(x) = {non oriented pairs (y, z) such that:

- y ∈ $\Gamma^{Ov,\subset}{}_G$(x), z ∈ $\Gamma^{Ov,\subset}{}_G$(x); (y, z) ∉ E ;
- at least one of both relations y ⊂ x or z ⊂ x holds;
- relation ((x σ y) ∧ (x σ z)) ∨ ((y σ x) ∧ (z σ x)) holds}

SFk(G, σ) may be written as $\Sigma_{x \in X}$ |E*(x)|. Since σ is (*Ov*, <<)-consistent, we may relax the "*at least … y ⊂ x or z ⊂ x holds*" constraint which characterizes E*(x).

So, for any x ∈ X:

|E*(x)| ≥

$\Sigma_{\text{y,z} \in \Gamma\text{G(x), (y, x)} \notin \text{E, y,z are located the same way with respect to x, } \sigma}$ **1.**

For any x ∈ X, we get a feasible solution A(x,σ) $\cup^{Ex}$ B(x, σ) of the *C-Max-Cut* instance defined by H(x), A_0(x), B_0(x), by setting: A(x,σ) = {y ∈ $\Gamma^{Ov,\subset}{}_G$(x), such that y σ x}; B(x, σ) = {y ∈ $\Gamma^{Ov,\subset}{}_G$(x), such that x σ y}. Its value is:

m(x) –

$\Sigma_{\text{x,y} \in \Gamma Ov,\subset\text{G(x), (y, x)} \notin \text{E, y, z located the same way with respect to x, } \sigma}$ **1**

≤ *C-Max-Cut*(H(x), A_0(x), B_0(x)).

It follows that, for any x ∈ X:

|E*(x)| ≥

$\Sigma_{\text{x,y} \in \Gamma Ov,\subset\text{G(x), (y, x)} \notin \text{E, y, z located the same way with respect to x, } \sigma}$ **1**

≥ CV(x).

Then, we get that SFk(G, σ) = Σ_x |E*(x)| ≥ Σ_x CV(x). We conclude. □

D. Solving C-Max-Cut and Evaluating CV(x)

The complexity of the *Max Cut* problem in the case of the complementary graph of an interval graph is still an open issue. However, things are easier with *C-Max-Cut*:

Theorem 4: *Given = (Z, F),* A_0 *and* B_0 *as in the definition of C-Max-Cut. Let us set, for every vertex z ∈ Z – (*$A_0 \cup B_0$*):*

- $d_H^-(A_0, z)$ = |*{t ∈ Z –* A_0 *such that t << z }*| + |*{t ∈* A_0 *such that t << z}*|;
- $d_H^+(B_0, z)$ = |*{t ∈ Z –* B_0 *such that z << t }*| + |*{t ∈* B_0 *such that z << t }*|.

Then we solve C-Max-Cut by setting:

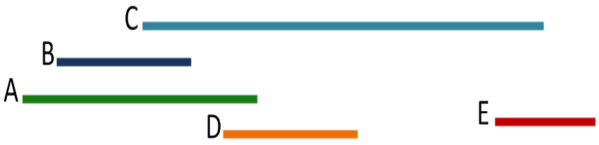
- *A = {z ∈ Z – (*$A_0 \cup B_0$*) such that* $d_H^-(A_0, z) \geq d_H^+(B_0, z)$*}* $\cup A_0$*;*
- *B = {z ∈ Z – (*$A_0 \cup B_0$*) such that* $d_H^-(A_0, z) < d_H^+(B_0, z)$*}* $\cup B_0$*.*

Proof: Left to the reader. □

E. An Exact Solution σ-bal for CLAP.

We construct this solution σ-*bal*, by setting, for any pair x, y in X, x σ-*bal* y if, and only if, one among the following options holds:

- (x << y) or (x *Ov* y);
- (x ⊂ y) and $d_{H(y)}^-(A_0(y), x) \leq d_{H(y)}^+(B_0(y), x)$; (E5)
- (y ⊂ x) and $d_{H(x)}^-(B_0(x), y) < d_{H(x)}^+(A_0(x), y)$. (E6)



CV(A) = 0; CV(B) = 0; CV(C) = 1;CV(D) = 0; CV(E) = 1; *Tr*(G) = 2;

σ-*bal*: B σ-*bal* A σ-*bal* D σ-*bal* C σ-*bal* E; BG(G, σ-*bal*) = 3

Figure 7: A σ-*bal* construction

Lemma 3: *The σ-bal relation is transitive.*

Proof: left to the reader. □

We are now ready to state the optimality of σ-*bal*.

Theorem 5: *The relation σ-bal is an optimal solution of CLAP, which satisfies:*

1. *BG(G, σ-bal) ≤ BG(G, σ-can).*
2. *Tr(G) ≤ BG(G, σ-bal) = Tr(G) +* Σ_x *CV(x) ≤ Tr(G) + Strong-Fork/2, where Strong-Fork is the number of strong forks of the interval graph G.*

Sketch of the Proof: From Lemma 3, we have that σ-*bal* is a (<<, *Ov*)-consistent linear ordering. Then the optimality of σ-*bal* (and so, the fact that BG(G, σ-*bal*) ≤ BG(G, σ-*can*)) derives, through a simple computation, from the fact that since it locally achieves, for any node x, the lower bound CV(x), then it also globally achieves the lower bound of Lemma 2. □

We easily deduce that this result has an algorithmic interpretation:

Corollary 3 (left to the reader): *Computing σ-bal may be done in O(Arc-⊂) time, where Arc-⊂ is the number of arcs of the digraph induced on X by the ⊂ordering.*

VII. numerical Experiments

We implemented both the Branch/Bound algorithm of Section 5 and the Approximation algorithm σ-*bal*. We did it on a LINUX server CentOS 5.4, Processor Intel Xeon 3.6 GHZ, while using the CPLEX 12 library when dealing with integer linear programs. For interval graphs, our Branch/Bound scheme could solve, in no more than few minutes, instances with up to 40 nodes. This allowed us to perform a comparative analysis of the precision of the Lower Bound $LB(G) = Tr(G) + \Sigma_x V(x)$ provided by Theorem 1, and of both approximation algorithms σ-*can* and σ-*bal*.

We use 10 instance groups related to Card(X) = 10, 20, 30 and 40, and for every instance group, compute:

- the mean gap LB-GAP =
  (LAP(G) –LB(G))/LB(G)
  between the optimal LAP value and the lower bound LB;
- the mean gap CLAP-GAP =
  (BG(G, σ-*bal*) -LAP(G))/LAP(G)
  between the optimal value of CLAP, computed by σ-*bal*, and the optimal value;
- the mean gap CAN-GAP =
  (BG(G, σ-*can*) -LAP(G))/LAP(G)
  between the value defined by the canonical ordering σ-*can*, and the optimal value.

We get results which are described in the following Table 1.

N	**10**	**20**	**30**	**40**
LB-GAP	9.8%	12.5%	10.7%	14.5%
CLAP-GAP	3.2%	4.5%	4.1%	5.0%
CAN-GAP	9.6%	12.3%	11.5%	10.9%

Table 1: Comparative precision of lower bound LB and approximation solutions σ-*can* and σ-*bal*.

Table 2 provides now the specific results related to n = 10, which by the way, gives an estimation of the LAP(G) values which may derive from interval graph of this size:

Instance number	**LB(G)**	**LAP(G)**	**CLAP(G)**	**CAN(G)**
1	10	12	12	13
2	8	8	8	8
3	13	16	17	19
4	12	13	14	15
5	12	13	13	14
6	10	11	11	11
7	15	15	17	18
8	13	15	15	17
9	11	11	11	12
10	9	10	10	10

Table 2: Values LAP(G), LB(G), CLAP(G) and CAN(G) related with a 10 instances group with Card(X) = 10.

VIII.Conclusion

This paper, with theoretical focus, proposes approximation results for the *Linear Arrangement* problem, in the case of interval graphs. Further research should be about the extension of our approaches to chordal graphs and circular graphs, as well as about the design of efficient exact algorithms.

References

1. Achouri S., Bossart T., Munier-Kordon A. (2009): A polynomial algorithm for MINDSC on a subclass of series parallel graphs, *RAIRO Operations Research*, pp. 145-156, DOI: 10.1051/ro/2009009
2. Barahona F., Mahjoub A.R (1986): On the cut polytope, *Math. Prog*. 36, pp. 157-173, DOI: 10.1007/BF02592023
3. Charon I., Hudry O. (2010): An updated survey on the linear ordering problem for weighted or unweighted tournaments, *Annals of Operations Research,* 175, pp. 107-158, DOI: 10.1007/010479-009-0648-7
4. Chung FRK. (1984): On optimal linear arrangement of trees. *Comp. & Maths/Appl*., 11, pp. 43-60, DOI: 10.1145/73833.738333.73866

5. Cohen J., Fomin F., Heggernes P., Kratsch D., Kucherov G. (2006): Optimal linear arrangement of interval graphs, *Proc. MFCS'06*, pp 267-279, Springer-Verlag, DOI: 10.1007/1182069_24
6. Corneil DG., Kim H., Natarajan S., Olarin S., Sprague AP. (1995): A simple linear time algorithm of unit interval graphs, *Information Processing Letters* 55, pp. 99-104, DOI: 10.1016/0020-0190(95)00046-F
7. Chvatal V., Ebenegger C. (1990): A note on line digraphs and the directed Max-Cut problem, *Discrete Applied Maths* 29, pp 165-170, DOI: 10.1016/0166-218X(90)90141-X
8. Even S., Shiloach Y. (1975): NP-Completeness of Several Arrangement Problems, *Technical Report #43*, *Computer Science Department*, *The Technion*, Haifa, Israel, DOI: 10.1007/11821069_24
9. Garey MR., Johnson DS. (1979): *Computers and intractability: a guide to the theory of NP-completeness*, Computer Press, ISBN-13: 978-0716710455.
10. Grotschel, M. (ed.) (2004): *The Sharpest Cut*, MPS-SIAM Series on Optimization, ISBN-13: 978-0898715521
11. Horton SB. (1997): *The optimal linear arrangement problem: algorithms and approximation*, Phd thesis, Georgia Institute of Technology.