About Casting 2D-Bin Packing into Network Flow Theory

Alain Quilliot, Hélène Toussaint

About Casting 2D-Bin Packing into Network Flow Theory

Alain Quilliot[1] **Hélène Toussaint**[2]

Research Report LIMOS/RR-10-11

30 avril 2010

[1]quilliot@isima.fr
[2]toussain@isima.fr

About Casting 2D-Bin Packing into Network Flow Theory.

Keywords: Network Flow Theory, Multi-commodity Flow, Graphs, 2D-Bin Packing.

Abstract.

In this paper, we aim at making appear the way Flow and Multicommodity Flow Theory may be used in order to deal with combinatorial geometry problems like the 2D-Bin Packing problem. In order to do it, we introduce a notion of no circuit double flow, we state a Reformulation Theorem which ties some multicommodity flow model with a given bin-packing problem, and we provide an algorithm whose purpose is to study the way one may deal with the no circuit constraint which is at the core of our multicommodity flow.

I. Introduction.

Dealing with a 2D-Bin Packing problem (see [COF84, LOD02]) means locating a family V of rectangles (*2D-bins*) inside a given domain $\mathcal{A}$ of the 2-dimensional affine space R^2, in such a way that for any pair (v, v') of rectangles of V, the interior of the intersection $v \cap v'$ is empty. One may specialize this problem by requiring the area $\mathcal{A}$ to be a rectangle with given length and height, or by imposing the orientation of the rectangles of V. Also, one may turn it into an optimization problem by considering the domain $\mathcal{A}$ as part of the problem and by requiring it to be with minimal length or with minimal area. Additional constraints may be imposed, related to the way some elements of V are positioned in relation to others, and, of course, one may also deal with higher dimensions (see [FAR03, FEK04, FEK104]) or with general cutting plane patterns (see CHR77, BUR04]).

The 2D-Bin Packing problem is a difficult one: whatever the way it is formulated (see [GAR79, COF84]), it remains NP-Complete. Practically, getting exact results for the case when the domain $\mathcal{A}$ is a rectangle whose height is known in advance and whose length has to be minimized, and when every rectangle v in V must be oriented in such a way that its width side be parallel to the width side of $\mathcal{A}$, may prove itself to be difficult as soon as the cardinality of V exceeds 20. When its comes to the design of exact methods, 2D-Bin Packing is usually handled through Integer Linear Programming (see [CAP05]), through a combination of Branch and Bound tree search and constraint propagation techniques (see [BOS03, MAR98, DEL02, CLA08]), or through adaptation of 2D-Knapsack or RCPSP algorithms (see [CAP04, HAD95, ELH08]). But efficient heuristics may also be designed: one may for instance refer to [COF84] for greedy algorithms, to [FAR03, LOD99] for local search methods driven by metaheuristic scheme (Tabu, Simulated Annealing, memetic approaches).

Network Flow Theory (see [AHU95, AHU93, MIN89]) is essentially related to the modelling and to the algorithmic handling of problems which involves the circulation of goods, persons, money, energy or information. It has been essentially used in order to optimize the design of transportation and telecommunication networks (see [DAH94, BIE96]), or in order to help in managing the activity of gas or electricity distribution networks (see [AHU95, PAR98, MIR90]). It proved itself to be a very powerful tool for the management of such problems, not only as a modelling tool, but also as a special link between the linear programming machinery and purely combinatorial techniques. Part of current trends is about handling network flow problems while taking into account purely combinatorial constraints (see [BEN00, BAL98, PAR98, CHR81]).

So, the goal of this paper is to show how a 2D-Bin Packing problem may be reformulated as a particular network flow problem and to provide us with some tips about the design of new algorithms for this problem. We are first going to explain the way 2D-Bin Packing may be cast into the Network Flow framework. Next, we shall state a Reformulation Theorem, which links both formalisms, through the introduction of a 2-multicommodity flow submitted to a *no circuit* constraint. Finally, we shall propose an insertion greedy algorithm for the 2D-Bin Packing Problem which will derive in a straightforward way from our reformulation scheme. But our goal will not be here to deal with optimization, but only to study the kind of techniques one might use in order to deal with multicommodity flows submitted to a combinatorial constraint such that the no circuit constraint.

II. Networks and Flows related to a 2D-Bin Packing Instance.

II.1. Preliminary notations and definitions.

About numbers, geometry and topology: we shall denote by Q the set of the rational numbers and by R the set of the real numbers; if A is a given subset of the affine two-dimensional space R^2, we denote by A° the interior of A (in the topological sense) and by Conv(A) the convex hull of A; if M = (x, y) is a point in R^2, and if l and h are two positive numbers, then we denote by R = Rect(x, y, l, h) the closed rectangle which is the convex hull of the four points (x, y), (x, y + h), (x + l, y) and (x + h, y + l). The segment [(x, y), (x, y + h)] ([(x, y + h), (x +l, y + h)], [(x, y), (x + l, y)], [(x + l, y), (x + l, y + h)]) is called the *left(top, bottom, right) side* of R.
If R = Rect(x, y, l, h) and R' = Rect(x', y', l', h') are such rectangles, then we say that:

- R' *right-dominates* R iff $x + l \leq x'$;
- R' *top-dominates* R iff $y + h \leq y'$;
- R' *dominates* R iff R' right-dominates R and R' top-dominates R;

If R' right-dominates R and if the 1-dimensional intervals [y, y + h] and [y', y' + h'] are intersecting, then we call *horizontal link* from R to R' any segment [(x + l, t), (x', t)] such that $t \in [y, y + h] \cap [y', y' + h']$. If R' top-dominates R and if the 1-dimensional intervals [x, x + l] and [x', x' + l'] are intersecting, then we call *vertical link* from R to R' any segment [(t, y + h), (t, y')] such that $t \in [x, x + l] \cap [x', x' + l']$.

About algorithms and lists: we shall denote by <- the value allocation operation: "x <- α" will mean that the variable x takes the value α. So the symbol = will be used inside algorithm descriptions as a comparator or as a descriptor: "Set z = (y + x)" will mean that z is a new variable whose value will be permanently equal to (y + x); if << is some linear (or complete) order relation defined on some finite set X, we consider << as both a binary relation and as a list. If A is some subset of X, we shall denote by Min(A, <<) (Max(A, <<)) the smallest (largest) element of A according to <<, and if x is some element in A, we shall denote by Succ(x, A, <<) (Pred(x, A, <<)) the successor (predecessor) of x in A according to <<, which will be undefined in case x = Max(A, <<) (Min(A, <<)); We denote by $\oplus$ the concatenation operator which acts on lists (and also paths).

II.2.The Simple 2D-Bin Packing Problem.

An instance (V, L, H) of the simple *2D-Bin Packing* problem is defined by a set V of *non null area 2D-bins*, i.e a set of pairs of 2-uples v = (l, h), where the rational numbers l = l(v) > 0 and h = h(v) > 0 are respectively the *length* and the *height* of v, and by two rational numbers L > 0 and H > 0.
Solving such an instance (V, L, H) means associating, with any 2D-item v = (l, h) in V, a closed rectangle R(v) = Rect(x(v), y(v), l, h) in such way that:

- for any pair (v, v') in V, $v \neq v'$, the interior $(R(v) \cap R(v'))°$ of $(R(v) \cap R(v'))$ is empty;
- for any v in V, we have the inclusion $R(v) \subseteq$ Rect(0, 0, L, H).

II.3. Deriving a Network Flow from a Solution R of a Simple 2D-Bin Packing Instance (V, L, H).

Recall: *Network Flows*.
Given a network G = (Z, E), i.e an oriented graph with node (vertex) set V and arc set E, we denote by [x, y] any arc with origin node x and end node y, and we call *flow vector* any E-indexed vector f such that:

for any node z in Z, $\Sigma_{\text{z is the origin of e}} f_e = \Sigma_{\text{z is the extremity of e}} f_e$. (*Kirshoff Law*)

By extension, ϕ being a function defined from the node set Z to the set Q of the rational numbers, we say that a E-indexed vector f is a ϕ-flow vector iff:

for any node z in Z, $\Sigma_{\text{z is the origin of e}} f_e = \Sigma_{\text{z is the extremity of e}} f_e = \phi(z)$ (*Extended Kirshoff Law*)

As a matter of fact, a ϕ-flow vector f is nothing but a flow f^{ϕ} which is defined on the network G* obtained from G by splitting any node z into two copies z' and z", by setting an arc from z' to z" and by turning any arc [z, u] into an arc [z", u'], and which is such that: $f^{\phi}_{[z',z"]} = \phi(z)$.
We talk about *multicommodity flow* vectors when the flow values are vectors, instead of numbers.

Let us consider now an instance (V, L, H) of the simple 2D-Bin Packing Problem. We may define a network G(V) = (V*, E*) by introducing two auxiliary nodes s (source) and p (pit), and by setting:

- $V^* = V \cup \{s, p\}$;
- $E^* = \{[v, v'], v, v' \in V \cup \{[s, v], [v, p], v \in V\} \cup \{[p, v]\}$.

If we are provided with a solution R of the 2D-Bin Packing instance (V, L, H), then we may derive from R in a natural way a flow vector F-H on the network G(V), by setting:

- $F\text{-}H_{[p, s]} = H$;

- For any v in V, such that R(v) = Rect(x(v), y(v), l, h), $F\text{-}H_{[s,\ v]}$ = Measure of I(R, v) $\subset$ [0, H] which is defined by: I(R, v) = {t $\in$ [0, H], such that the horizontal link [(0, t), (x(v), t)] intersects the left side of R(v) and does not intersect any rectangle R(v'), v' $\neq$ v};
- For any v in V, such that R(v) = Rect(x(v), y(v), l, h), $F\text{-}H_{[v,\ p]}$ = Measure of J(R, v) $\subset$ [0, H] which is defined by: J(R, v) = {t $\in$ [0, H], such that the horizontal link [(x(v) + l, t), (L, t)] intersects the right side of R(v) and does not intersect any rectangle R(v'), v' $\neq$ v};
- For any v, v' in V, v $\neq$ v', such that R(v) = Rect(x(v), y(v), l, h), $F\text{-}H_{[v,\ v']}$ = Measure of K(R, v, v') $\subset$ [0, H] which is defined by: K(R, v, v') = {t $\in$ [0, H], such that there exists an horizontal link [(x(v) + l, t), (x(v'), t)] from R(v) to R(v') which does not intersect any rectangle R(v"), v" $\neq$ v, v'}.

We easily see that for any 2D-item v in V:

- $\Sigma_{\text{v is the extremity of e}}\ F\text{-}H_e = \Sigma_{\text{v is the extremity of e}}\ F\text{-}H_e = h(v)$;
- $\Sigma_{\text{s is the origin of e}}\ F\text{-}H_e = \Sigma_{\text{s is the extremity of e}}\ F\text{-}H_e = H$;
- $\Sigma_{\text{p is the origin of e}}\ F\text{-}H_e = \Sigma_{\text{p is the extremity of e}}\ F\text{-}H_e = H$;

This also means that if we define a function h* by setting, for any v in V* = V $\cup$ {s, p}:

if v $\in$ V then h*(v) = h(v)
else h*(v) = L.

then F-H is a h*-flow vector defined on G(V).

Interpretation: F-H transports the height geometrical resource H from the left side to the right side of $\mathcal{A}$ = Rect(0, 0, L, H) and provides with it the 2D-bins of V in such way it allows them to find room in $\mathcal{A}$.

By the same way, replacing H by L, "horizontal" by vertical", and setting, for any v in V*= V $\cup${s, p}:

if v $\in$ V then l*(v) = l(v) else l*(v) = L,

allows us to define a l*-flow vector F-L on the network G(V), which in turn expresses the transportation through the 2D-bins of V of the length geometrical resource L from the bottom side to the top side of the main rectangle $\mathcal{A}$ = Rect(0, 0, L, H).

Those two flows F-H and F-L define a *2-commodity flow vector*: we shall also talk about *double flow vector*.

II.4. A basic property of the flow vectors F-H and F-L..

Let us consider a solution R of a 2D-Bin Packing instance (V, L, H), together with 2 flow vectors F-H and F-L deriving from R as above. We derive from F-H and F-L two oriented graphs G-H = (V, E-H) and G-L = (V, E-L) by setting:

- the arc [v, v'] is in E-H (E-L) iff the flow vector value $F\text{-}H_{[v,\ v']}$ ($F\text{-}L_{[v,\ v']}$) is non null.

Clearly, G-H and G-L share no arc.
Then we may set:

- $E\text{-}H^-$ = {arcs [v, v'] of V.V such that the arc [v', v] is an arc of G-H};
- G-L+G-H = (V, E-H $\cup$ E-L);
- G-L – G-H = (V, E-L $\cup$ $E\text{-}H^-$).

Then we get:

Lemma 1: Non Circuit Lemma.
The oriented graphs G-L + G-H and G-L - G-H do not admit any circuit.

Proof-Lemma.
Because of symmetry, we only need to deal with the case of G1+G2. As a matter of fact, G1+ G2 is a partial subgraph of the oriented graph G-$\mathcal{Rect}$ which is defined on the set $\mathcal{Rect}$ of the non empty rectangles Rect(x, y, l, h), l $\neq$ 0, h $\neq$ 0, of the 2D affine plane R^2 by setting that there exists an arc [r, r'], r, r' $\in$ $\mathcal{Rect}$, from r = Rect(x, y, l, h) to r' = Rect(x, y', l', h') if there exists an horizontal link or a vertical link from r to r'.
So, we only need to prove that this graph G-$\mathcal{Rect}$ does not contain any circuit. Let us suppose that the converse is true, i.e that there exists some circuit $\Gamma = (r_0, r_1, \ldots, r_n, r_{n+1} = r_0)$ in G-$\mathcal{Rect}$, which may chosen with minimal length n +1. Of course, all the arcs of G cannot derive from horizontal (vertical) links.We may assume that the arc $[r_0, r_1]$ corresponds to an horizontal link, and that s is the smallest index in 1..n such that the arc $[r_s, r_{s+1}]$ corresponds to a vertical link. Let us set r_0 = Rect(x_0, y_0, l_0, h_0), r_s = Rect(x_s, y_s, l_s, h_s) and r_{s+1} = Rect(x_{s+1}, y_{s+1}, l_{s+1}, h_{s+1}). Because of the minimality of n, and since the projections on the vertical axis of R^2 of the left sides of $r_0, r_1, \ldots, r_n$ cover all the interval (the convex hull) defined by $y_0, y_1, \ldots, y_s$ and $y_0 + h$, it comes that y_{s+1} must

be at least equal to $y_0 + h$. Reasoning the same way with the projections on the horizontal axis of the projections of the bottom sides of r_0, r_1, …, r_s, we see that x_{s+1} must be at least equal to $x_0 + l$. It comes that r_{s+1} must dominate r_0. Keeping on the same way, we see that for any index $q \geq s+1$, we have that r_q dominates r_0. We conclude since the "dominate" relation is an order relation on the non null area rectangle set. *End-Lemma*.

III. A converse result: the resolution of the simple 2D-Bin Packing problem is equivalent to the resolution of a specific double-flow problem.

The No Circuit Lemma of section II.4 leads us to set a new definition: let us consider an instance (V, L, H) of the simple 2D-Bin Packing problem, together with the network G(V) = (V*, E*) which was defined in II.2 and with the function h* and l* which were defined on the node set V* in II.3; then we say that a pair (F-H, F-L) is a *double-flow vector associated with the instance* (V, L, H) iff F-H is a h*-flow vector and F-L is a l*-flow vector; we say that this double-flow is *no circuit* iff the oriented graphs G-L + G-H and G-L – G-H which derive from G-H and G-L as in II.4 do not admit any circuit.

III.1. A Structural Result.

The above definitions allows us to state the following result, which turns the 2D-Bin Packing instance related to (V, L, H) into the search for a double-flow vector (2 multi-commodity flow vector) submitted to a specific combinatorial constraint (the no circuit constraint):

Theorem 1 (Reformulation Theorem).
The instance (V, L, H) of the simple 2D-Bin Packing Problem admits a solution if and only if there exists a no circuit double-flow vector associated with (V, L, H).

Proof-Theorem.
The part (only if) of the above equivalency derives in a straightforward way from the Non Circuit Lemma.
Conversely, let us consider a no circuit double-flow vector (F-H, F-L) associated with (V, L, H). As in section II.4, we denote by G-H = (V, E-H) and G-L = (V, E-L) the oriented graphs which are defined on the node set V by those of the arcs of the graph G(V) which are respectively support for F-H and F-L. We are first going to prove the following lemma:

Lemma 2.
Let A-H and A-L be two subsets of the arc set V.V, which are such that neither A-L ∪ A-H nor A-L ∪ A-H⁻ admits any circuit, and let v, v' in V, v ≠ v', such that neither the arc [v, v'] nor the arc [v', v] is in A-H ∪ A-L. Then it is possible to choose some subset A among A-H and A-L and some arc e among [v, v'] and [v', v] and to insert e into A in such a way that A-L ∪ A-H and A-L ∪ A-H⁻ remain no circuit.

Proof-Lemma.
Since A-L ∪ A-H and A-L ∪ A-H⁻ do not contain any circuit, they both define a partial order relation on the set V, and both admit a linear extension. Let us recall that a linear ordering of a set Z is an order relation << which is such that, for any two elements x, y in Z, we have either x << y or y << z. So let σ and τ be two linear extensions of respectively A-L ∪ A-H and A-L ∪ A-H⁻. We may suppose that v is before v' according to σ (we may think into σ and τ as into two lists). Then we need to consider two cases:

- first case: v is before v' according to τ.
  In such a case, we add the arc [v, v'] to A-L. If this insertion were creating a circuit, that would mean either the existence of a A-L ∪ A-H path from v' to v, which would contradict the fact that v is before v' according to σ, or the existence of some A-L ∪ A-H⁻ path from v' to v, which would contradict the fact that v is before v' according to τ;
- second case: v' is before v according to τ.
  In such a way, we add the arc [v, v'] to A-H, and we proceed as above in order to check that this operation is not going to create any circuit.

End-Lemma.

Lemma 2 allows us to assert that it is possible to extend E-L and E-H into two arc sets E-L* and E-H* in such a way that:

- both E-L* ∪ E-H* and E-L ∪ E-H*⁻ are no circuit;

- for any node pair v, v' in V, either [v, v'] or [v', v] is in E-L* $\cup$ E-H*. (P1)

We define an oriented graph structure H-H = (V $\cup$ {s, p}, E-H* $\cup$ {[s, v], [v, p], v $\in$ V}) on the node set (V $\cup$ {s, p}), together with a length function D-H, which provides us with the length of any arc of H-H according to the following formulas:

- D-H(s, v) = 0;
- D-H(v, v') = D-H(v, p) = l(v);

By the same way, we define an oriented graph structure H-L = (V $\cup$ {s, p}, E-L* $\cup$ {[s, v]}) on the node set (V $\cup$ {s, p}), together with a length function D-L, which provides us with the length of any arc of H-H according to the following formulas:

- D-L(s, v) = 0;
- D-L(v, v') = D-L(v, p) = h(v);

Then for any v in V, we may compute:

- Π-H(v) = Length of a largest (in the sense of D-H) path Γ-H from s to v in the oriented graph H-H;
- Π-L(v) = Length of a largest (in the sense of D-L) path Γ-L from s to v in the oriented graph H-L;

This allows us to associate, with any 2D-item v = (l, h) in V, the rectangle R(v) = Rect(Π-H(v), Π-L(v), l, h). Clearly, a consequence of the above property (P1) is that the interiors of two rectangles R(v) and R(v'), v $\neq$ v', cannot be intersecting. What remains to be checked is that all those rectangles R(v), v $\in$ V, are included into the main area $\mathcal{A}$ = Rect(0, 0, L, H). In order to deal with this last point, we first state the following lemma:

Lemma 3.
Let W some subset of V such that, whatever be two nodes v, v' in W, there is no E-L path from v to v'. Then the following inequality is true: $\Sigma_{v \in W}\, l(v) \leq L$.

Proof-Lemma.
We may proceed by induction on the cardinality of the arc set E-L of the support graph G-L. Let us pick up some node v_0 in W, together with some path Γ of G-L which starts from s and ends into p, and which includes v_0. Then let us set $\delta = \text{Inf}_{v, v' \in V \cup \{s, p\} \text{ consecutive in } \Gamma} F\text{-}L_{[v, v']}$. The number δ is strictly positive, and the path Γ does contain any node of W, but v_0. So we conclude by first replacing the global length L by L- δ, and, for any v in $\Gamma \cap$ V, the length l(v) by l(v) – δ, by next removing the δ quantity from any flow value F-L(e), where e is an arc of Γ, and by finally applying the induction hypothesis to the resulting network flow F-L. *End-Lemma*.

Clearly, we may state a similar result related to the vector flow F-H: if W is some subset of V such that, whatever be two nodes v, v' in W, there is no E-H path from v to v', then the following inequality is true: $\Sigma_{v \in W}$ h(v) $\leq$ H. We notice that all the rectangles R(v) = Rect(Π-H(v), Π-L(v), l, h), v $\in$ V, are included into the rectangle Rect(0, 0, Π-H(p), Π-L(p)). So what we need to prove is that Π-H(p) $\leq$ L and that Π-L(p)) $\leq$ H. We may choose to focus on the first inequality Π-H(p) $\leq$ L and consider some largest path (for the D-H length function) Γ-H from s to p. Since both E-L* $\cup$ E-H* and E-L $\cup$ E-H*$^{-}$ are no circuit, we are sure that whatever be two nodes v, v' in Γ-H, v, v' $\neq$ s, p, there is no E-L path from v to v' in the support graph G-L, which means that v and v' do not exchange any F-L flow. Moreover, we may write: Π-H(p) = $\Sigma_{v \in \text{G-H}, v \neq s, p}$ l(v). It comes that we only need to apply Lemma 3 with W = V $\cap$ Γ-H in order to conclude. *End-Theorem*.

III.2. A Reconstruction Algorithm.

Let us consider a no circuit double-flow vector (F-H, F-L), associated with an instance (V, L, H) of the Simple 2D-Bin Packing problem. One easily deduces from the proof of the Reformulation Theorem that the following *Reconstruction Algorithm* derives a solution R of (V, L,H) from a pair (F-H, F-L).

Reconstruction Algorithm.
Input: *the no circuit double flow vector (F-H, F-L), associated with (V, L, H).*
Output: *two functions Π-H and Π-L, which, to any v in V, make correspond Π-H(v) and Π-L(v) in such a way that the rectangles R(v) = Rect(Π-H(v), Π-L(v), l(v), h(v)), v $\in$ V, define a solution R of (V, L, H).*
Let E-H and E-L be the support arc sets associated with F-H and F-L;
Compute two linear extensions σ and τ of respectively the arc sets E-L $\cup$ E-H and E-L $\cup$ E-H^{-};
E-H* <- E-H; E-L* <- E-L;
For x, y in σ, y located after x in σ, do
    If x is located before y in τ then Insert the arc [x, y] into E-L*
    Else Insert the arc [x, y] into E-H*;

For any v in V compute:

- Π-H(v) = Length of a largest path Γ-H from s to v in the oriented graph H-H = (V $\cup$ {s, p}, E-H* $\cup$ {[s, v],[v, p], v $\in$ V}), considered as provided with the length function D-H, which is defined, for any v, v' in V, by: D-H(s, v) = 0; D-H(v, v') = D-H(v, p) = l(v);
- Π-L(v) = Length of a largest path Γ-L from s to v in the oriented graph H-L = (V $\cup$ {s, p}, E-L* $\cup$ {[s, v], [v, p], v $\in$ V}) considered as provided with the length function D-L, which is defined, for any v, v' in V, by: D-L(s, v) = 0; D-L(v, v') = D-L(v, p) = l(v).

IV. A Simple Insertion Algorithm for the Construction of a No Circuit Double Flow.

We are going to propose here a simple insertion algorithm, which deals with the optimization version of the 2D-Bin Packing problem:

The 2D-Bin Packing Problem with Length Minimization:
{*Let V a non null area 2D-bin set, given together with a positive height number H. Find the smallest length number L such that the instance of the Simple 2D-Bin Packing Problem defined by V, L and H admits a solution*}

IV.1 The Bipartite Double-Flow procedure and the Opt-Insertion Procedure.

In order to describe it in an accurate way, we introduce a notion of *Bipartite Double Flow Vector*, together with a *Bipartite Double Flow* Problem:

Bipartite Double Flow.
Let (X, <<) be some partially ordered set, and X = A $\cup$ B, a partition of X into two disjoint sets. We suppose that s and p are respectively minimum and maximum elements in X, in such a way that s $\in$ A, p $\in$ B, and that Out-H and Out-L (In-H and In-L) are Q-valued functions with respective domains A and B, in such a way:

Out-H $\geq$ 0; Out-L $\geq$ 0; In-H $\geq$ 0; In-L $\geq$ 0;
Out-H(s) = In-H(p) = 0;
$\Sigma_{x \in A}$ Out-H(x) = $\Sigma_{y \in B}$ In-H(y);
$\Sigma_{x \in A}$ Out-L(x) = $\Sigma_{y \in B}$ In-L(y);

Then we say that two vectors G-H = (G-$H_{x,y} \geq 0$, x $\in$ A, y $\in$ B) $\geq$ 0, and G-L = (G-$L_{x,y} \geq 0$, x $\in$ A, y $\in$ B) $\geq$ 0, define a *bipartite double flow vector* related to the vectors In-H, In-L, Out-H and Out-L iff:

- for any x in A, Out-H(x) = $\Sigma_{y \in B}$ G-$H_{x,y}$ and Out-L(x) = $\Sigma_{y \in B}$ G-$L_{x,y}$;
- for any y in B, In-H(y) = $\Sigma_{x \in A}$ G-$H_{x,y}$ and In-L(y) = $\Sigma_{x \in A}$ G-$L_{x,y}$.
- We say that this bipartite double flow is *no circuit* iff there does not exist any circuit in the oriented graph $\mathcal{N}$ whose vertex set is X and whose arc set is E = { [x, x'] such that x, x' $\in$ A and x << x'} $\cup$ {[y, y'], y, y' $\in$ B such that y << y'} $\cup$ {[x, y], x $\in$ A, y $\in$ B, such that G-$L_{x,y} \neq 0$} $\cup$ {[y, x], x $\in$ A, y $\in$ B, such that G-$H_{x,y} \neq 0$}.

This definition leads us to introduce the following *Bipartite Double Flow Problem*:

The Bipartite Double Flow Problem: *{We consider (X, <<), A, B, In-H, In-L, Out-H, Out-L as above while assuming that Out-L(s) and In-L(p) are undetermined. Find 2 numbers Out-L(s) and In-L(p) $\geq$ 0, together with a no circuit bipartite double flow vector (G-H, G-L) related to In-H, In-L, Out-H and Out-L in such a way that Out-L(s) be the smallest possible.*}

In order to deal with this problem, we design the following *Bipartite Double Flow* Procedure:

Bipartite-Double-Flow Procedure.
Input: (X, <<), A, B, In-H, In-L, Out-H, Out-L as above.
Output: a value Out-L(s), together with a no circuit bipartite double flow vector G-H, G-L.

Main Loop.
Extend << into a linear ordering;
Compute G-H through the *Match-Flow* procedure below, while considering that (Out, In) = (Out-H, In-H) and that G = G-H;

Compute $\alpha = Sup_{y \in B} [(\Sigma_{z << y \text{ or } z = y} In\text{-}L(z)) - (\Sigma_{x \setminus z << y \text{ for any } z \text{ such that } F\text{-}H(x, z) \neq 0} Out\text{-}L(x))]$;
Compute $\beta = \alpha + (\Sigma_{x \in A, x \neq s} Out\text{-}L(x)) - (\Sigma_{y \in B} In\text{-}L(y))$;
$In\text{-}L(p) <- \beta$; $Out\text{-}L(s) <- \alpha$;
Compute G-L through the *Match-Flow* Procedure below, while considering that (Out, In) = (Out-L, In-L) and that G = G-L;
Bipartite-Double-Flow <- Out-L(s).

Match-Flow Procedure:
Extend << into a linear ordering;
x <- Min(A, <<); y <- Min(B, <<);
While x and y are both defined do
r <- Inf(Out(x), In(y)); $G_{x,y}$ <- r;
In(y) <- In(y) – r; Out(x) <- Out(x) – r;
If Out(x) ≠ 0 then y <- Succ(y, B, <<)
Else x <- Succ(x, A, <<);

Theorem 2.
The above Bipartite Double Flow procedure computes a feasible solution of the Bipartite Double Flow problem. In case the partial order relation << is linear, this solution is optimal.

Proof-Theorem.
We first easily check that the Match-flow Procedure yields a vector G such that: (P2)

- for any x in A, $Out(x) = \Sigma_{y \in B} G_{x,y}$;
- for any y in B, $In(y) = \Sigma_{x \in A} G_{x,y}$;
- there does not exist x, x' in A, y, y' in B such that: $x << x', y' << y, G_{x,y} \neq 0$ and $G_{x',y'} \neq 0$.

So the first part of this result comes in straightforward way from the way the quantity α is computed. For any y in B, the inequality:

$\Sigma_{z << y \text{ or } z = y} In\text{-}L(z) \geq (\Sigma_{x \setminus z << y \text{ for any } z \text{ such that } F\text{-}H(x, z) \neq 0} Out\text{-}L(x)) + \alpha$,

combined with the fact that G-H satisfies the (P2) property (is an ordered bipartite flow), implies that the G-L flow value which enters into y comes from nodes x such that:

for any z in B such that $H\text{-}H_{x, z} \neq 0$, we have $z << y$. (P3)

Let us denote by $\mathcal{N}$ = (X, E) the oriented graph whose vertex set is X and whose arc set is E = {[x, x'] such that x, x' ∈ A and x << x'}∪ {[y, y'], y, y' ∈ B such that y << y'} ∪ {[x, y], x ∈ A, y ∈ B, such that $G\text{-}L_{x,y} \neq 0$} ∪ {[y, x], x ∈ A, y ∈ B, such that $G\text{-}H_{x,y} \neq 0$}. We deduce from (P3) that, for any path {x, y, y', x'} in $\mathcal{N}$, such that x, x' ∈ A, y, y'∈ B, we must have x << x' (similar statement for the case when we exchange the roles played by A and B). Then it comes that the bipartite double flow (G-H, G-L) is no circuit.

In order to get the second part of this result, we consider some no circuit feasible double flow vector (H-H, H-L) which is such that the (P2) property misses to be satisfied for at least one of both flow vectors H-H or H-L. Let us suppose for instance that H-H does satisfies (P2) and that x, x', y, y' are such that: $x << x', y' << y, H\text{-}H_{x, y} \neq 0$ and $H\text{-}L_{x', y'} \neq 0$. The we may set $\delta = Inf(H\text{-}H_{x, y}, H\text{-}H_{x', y'}$ and redirect the flow F-H by setting:

$H\text{-}H_{x, y} <- H\text{-}H_{x, y} - \delta$; $H\text{-}H_{x', y'} <- H\text{-}H_{x', y'} - \delta$;
$H\text{-}H_{x, y'} <- H\text{-}H_{x, y'} + \delta$; $H\text{-}H_{x', y} <- H\text{-}H_{x', y} + \delta$.

By doing this, we keep the no circuit property for the double flow vector (H-H, H-L). Let us for instance suppose that some circuit Γ exists in the oriented graph $\mathcal{N}$ which may be expressed as a concatenation of some path Γ_1 and of the arc [y, x']: Γ = {y, x'}⊕ Γ_1. Of course, Γ_1 starts in x' and ends in y. Then we see that the concatenation {y, x, x'}⊕ Γ_1 makes appear a circuit in the oriented graph $\mathcal{N}$ defined as it was before the application of the above redirection process. Also, this flow redirection process makes H-H gets closer to the (P2) property in the sense of the metrics induced by the lexicographic order $<<_H$ defined on the bipartite flow vectors H-H as follows:

- $H\text{-}H <<_H H\text{-}H'$ iff there exists x ∈ A, y ∈ B such that:
  - for any x'∈ A such that x' << x, and for any y ∈ B, we have: $H\text{-}H_{x', y} = H\text{-}H'_{x', y}$;
  - for any y' ∈ B such that y' << y, we have: $H\text{-}H_{x, y'} = H\text{-}H'_{x, y'}$;
  - $H\text{-}H_{x, y} \geq H\text{-}H'_{x, y}$.

It comes that we may make in such a way that (H-H, H-L) both satisfy the (P2) property. Since the above Match-Flow Lemma tells us that H-H and H-L are unique once the value Out-L(s) is fixed, we only need to check that

there cannot exist any no circuit bipartite double flow vector (H-H, H-L) which is such that both H-H and H-L satisfy (P2) and that Out-L(s) < α. This comes easily from the fact that if Out-L(s) < α, then there must exist $y \in B$ such that $\Sigma_{z << y \text{ or } z = y}$ In-L(z) < ($\Sigma_{x \setminus z << y \text{ for any z such that F-H(x, z)} \neq 0}$ Out-H(x)) + Out-L(s). This inequality means the existence of z << , $z \in B$, which is going to receive some H-L flow value from some x which will also send H-H flow to some y' such that y' >> y or y' = y. Such a situation induces the existence of a circuit {x, z, y', x} in the oriented graph $\mathcal{N}$, and consequently a contradiction with the no circuit property. *End-Theorem*.

We may now introduce a notion of *Insertion Flow*: let (X, <<) be some partially ordered set, and X= A $\cup$ B, a partition of X into two disjoint sets, and let x_0 be some 2D-item, which is not in X and which is provided with some length $l(x_0)$ and some height $h(x_0)$. We suppose that s and p are respectively minimum and maximum elements in X, in such a way that $s \in A$, $p \in B$, and that Out-H and Out-L (In-H and In-L) are Q-valued functions respectively defined on A and B in such a way:

Out-H ≥ 0; Out-L ≥ 0; In-H ≥ 0; In-L ≥ 0; Out-H(s) = In-H(p) = 0;

$\Sigma_{x \in A}$ Out-H(x) = $\Sigma_{y \in B}$ In-H(y); $\Sigma_{x \in A}$ Out-L(x) = $\Sigma_{y \in B}$ In-L(y);

Then we say that two vectors G-H = ($G\text{-}H_{x,y} \geq 0$, $x \in A$, $y \in B \cup \{x_0\}$) ≥ 0, and G-L = ($G\text{-}L_{x,y} \geq 0$, $x \in A \cup \{x_0\}$, $y \in B$) ≥ 0, define an *insertion flow vector* related to the vectors In-H, In-L, Out-H and Out-L iff:

- for any x in A, Out-H(x) = $\Sigma_{y \in B \cup \{x0\}}$ $G\text{-}H_{x,y}$ and Out-L(x) = $\Sigma_{y \in B \cup \{x0\}}$ $G\text{-}L_{x,y}$;
- for any y in B, In-H(y) = $\Sigma_{x \in A \cup \{x0\}}$ $G\text{-}H_{x,y}$ and In-L(y) = $\Sigma_{x \in A \cup \{x0\}}$ $G\text{-}L_{x,y}$;
- $h(x_0) = \Sigma_{y \in B}$ $G\text{-}H_{x0,y} = \Sigma_{x \in A}$ $G\text{-}H_{x,x0}$; $l(x_0) = \Sigma_{y \in B}$ $G\text{-}L_{x0,y} = \Sigma_{x \in A}$ $G\text{-}L_{x,x0}$;

We say that this insertion flow is *no circuit* iff there does not exist any circuit in the oriented graph $\mathcal{M}$ with node set $X \cup \{x_0\}$ and arc set E defined by: E = {[x, x'], x, x' $\in$ A \ x << x'} $\cup$ {[y, y'], y, y' $\in$ B \ y << y'} $\cup$ {[x, y], x, y $\in X \cup \{x_0\}$ \ $G\text{-}L_{x,y}$ is defined and non null} $\cup$ {[y, x], x, y $\in X \cup \{x_0\}$ \ $G\text{-}H_{x,y}$ is defined and non null}.

This definition leads us to introduce the following *Insertion Flow Problem*:

The Insertion Flow Problem: *{We consider (X, <<), A, B, In-H, In-L, Out-H, Out-L and x_0 as above while assuming that Out-L(s) and In-L(p) are undetermined. Find 2 values Out-L(s) and In-L(p) $\geq$ 0, together with a no circuit insertion flow vector (G-H, G-L) related to In-H, In-L, Out-H and Out-L in such a way that Out-L(s) be the smallest possible.*}

In order to deal with this Insertion Flow Problem, we first need to compute the values $G\text{-}H_{x, x0}$ and $G\text{-}L_{x, x0}$, $x \in A$. We will call this process the *attachment process*. We clearly must do it in such a way that there will not exist any pair x, x' in A such that: x << x' or x = x', $G\text{-}H_{x, x0} \neq 0$ and $G\text{-}L_{x', x0} \neq 0$. (P4)

As a matter of fact we do it through an ATTACH Procedure, which takes a node u in A as an input, and which computes values $G\text{-}H_{x, x0}$ and $G\text{-}L_{x, x0}$, $x \in A$ which satisfy (P4) and which are also such that: (P5)

- if x and x'are such that: x' << x << u << x" and $G\text{-}L_{x', x0} \neq 0$, then we also have: $G\text{-}L_{x, x0}$ = Out-L(x);
- if x and x' are such that: u (<< or equal) x << x'and $G\text{-}H_{x', x0} \neq 0$, then we also have: $G\text{-}H_{x, x0}$ = Out-H(x);

This procedure ATTACH, which is completely determined by (P4) and (P5) if the ordering << is linear, consequently modifies the values Out-H(x) and Out-L(x), $x \in A$. It provides as an output the value $G\text{-}L_{s, x0}$. The node u is called the *Attachment Node*.

Then we handle the Insertion Flow problem through the following *Insertion* Procedure:

Insertion Procedure(u: u is a node of A)

R <- ATTACH(u);

Apply the *Bipartite Double Flow* procedure to $X \cup \{x_0\}$, $A \cup \{x_0\}$, while considering that:

- << is extended in such a way that Pred(u, A, <<) << x_0 << u;
- Out-H(x_0) = $h(x_0)$; Out-L(x_0) = $l(x_0)$;

Insertion <- R + Out-L(s);

Comment: the value provided by Insertion(u) is the additional value Out-L(s) value which is required in order to perform the insertion of the 2D-item x_0 "between" A and B, with *attachment* node u.

Opt-Insertion Procedure:

Choose $u_0 \in A$ in such a way that the value provided by an application of the procedure Insertion(u_0) be the smallest possible;

S_0 <- Insertion(u_0); (* this instruction requires an effective application of the procedure Insertion(u_0)*)
Opt-Insertion <- (u_0, S_0);

Comment: the value provided by Opt-Insertion(u) is a pair (attach-node, value) which provides us with an adequate attachment node u_0 and with an additional Out-L(s) value which makes possible performing the insertion of the 2D-item x_0 between A and B, with attachment node u_0.

We may state the following result:

Theorem 3.
The Opt-Insertion procedure yields a feasible solution of the Insertion Flow Problem. In case, the partial order relation << is linear, this solution is optimal.

Proof-Theorem.
The first part of this statement comes in an immediate way from property (P4) and from Theorem 2.
In order to get the second part of this statement, we suppose that << is linear and we see that we only need to check that if we consider some feasible solution (G-H, G-L) of the Insertion Flow Problem, and if we set:

u = Inf x, x such that G-H(x, x_0) $\neq$ 0,

then it is possible to modify (G-H, G-L) in such a way that we end getting the (P5) property, without making us lose the feasibility of (G-H, G-L) and without making Out-L(s) increase. In order to do it, we consider for instance x, x' in A, such that u << x' and such that:

- u << x or u = x;
- G-$H_{x', x0}$ $\neq$ 0 and G-$H_{x, x0}$ $\neq$ Out-H(x).

In such a case, we may find y $\in$ B, such that G-$H_{x, y}$ $\neq$ 0, and we may apply to G-H the following flow redirection process:

- R <- Inf G-$H_{x',x0}$, G-$H_{x, y}$;
- G-$H_{x',x0}$ <- G-$H_{x',x0}$ - R ; G-$H_{x,,y}$ <- G-$H_{x,,y}$ – R ;
- G-$H_{x',,y}$ <- G-$H_{x',,y}$ + R ; G-$H_{x,,x0}$ <- G-$H_{x,,x0}$ + R ;

We may of course define a similar redirection process while dealing with G-L. Then it becomes simple matter check that applying such a redirection process makes (G-H, G-L) get closer to the (P5) property while not making us lose neither the feasibility of (G-H, G-L) nor making Out-L(s) increase. Thus, if (G-H, G-L) is an optimal solution of the Insertion Flow Problem, we may turn it into an optimal solution which satisfies (P5). Then Theorem 2 tells us that we may next deduce, through application of the Insertion Procedure, an other feasible solution (G-H*, G-L*), with a value which is better or identical to the value of (G-H, G-L). So we conclude. *End-Theorem.*

IV.2. The Double-Flow Algorithm.

We are now able to describe the **double-flow** algorithm, which deals with the *2D-Bin Packing Problem with Length Minimization*. This algorithm works as follows: any time we enter the main loop of this algorithm, some subset W of V has been "inserted" in some rectangle Rect(0, 0, L, H), which means that some no circuit double flow vector (F-H, F-L) has been computed, related to the triple (W, L, H). Also, a linear extension σ of the no circuit arc set E-H $\cup$ E-L together with a linear extension τ of the no circuit arc set E-L $\cup$ E-H$^-$ have been computed, E-H and E-L being respectively the support arc set associated with W, F-H and F-L. So we pick up v_0 in V – W, and we try to "insert" v_0 into W, which means that we try to compute F-H and F-L in such a way that they define a no circuit double flow vector related to the triple (W $\cup$ {v_0}, L, H).

In order to do it, In order to do it, we call *Cut* of W, related to σ and τ, any subset U of W such that:

- for any v in U, and any v' in W such that v' σ v, we have v' $\in$ U. (P6)

For any such a Cut, we set: (P7)

- X = W $\cup${s, s-aux, p, p-aux}; A = *Cut*(v) $\cup${s, s-aux}; B = X – A; << is the relation τ, extended in such a way that s (s-aux) is maximal (minimal) in A and that p (p-aux) is minimal (maximal) in B;
- for any x in *Cut*(v), Out-H(x) = $\Sigma_{y \in B}$ F-$H_{x,y}$ and Out-L(x) = $\Sigma_{y \in B}$ F-$L_{x,y}$;
- for any y in W – *Cut*(v), In-H(y) = $\Sigma_{x \in A}$ F-$H_{x,y}$ and In-H(y) = $\Sigma_{x \in A}$ F-$H_{x,y}$;
- Out-H(s) = $\Sigma_{y \in B}$ F-$H_{s,,y}$; Out-L(s) = 0; In-H(p) = $\Sigma_{x \in A}$ F-$H_{x,p}$; In-L(p) = 0;
- Out-H(s-aux) = In-H(s-aux) = 0; Out-L(s-aux) and In-L(s-aux) are to be minimized.

This construction enables us to apply the Opt-Insertion Procedure we just described above. Let us suppose that we just did it: we obtained an insertion flow (G-H, G-L), and we may derive F-H and F-L values by setting: (P8)

- for any v in *Cut*(v) $\cup$ {s, v_0}, w in (W –*Cut*(v)) $\cup$ {p, v_0}: F-$H_{v, w}$ = G-$H_{v, w}$;
- for any v in *Cut*(v) $\cup$ {v_0}, w in (W –*Cut*(v)) $\cup$ {v_0}: F-$L_{v, w}$ = G-$L_{v, w}$;
- for any w in (W –*Cut*(v)) $\cup$ {v_0}: F-$L_{s, w}$ = G-$L_{s\text{-}aux, w}$;
- for any v in *Cut*(v) $\cup$ {v_0}: F-$L_{v, p}$ = G-$L_{v, p\text{-}aux}$;
- F-$L_{s, p}$ = G-$L_{s\text{-}aux, p\text{-}aux}$;

and by keeping on with former values F-$H_{v, w}$ and F-$L_{v, w}$ for any pair (v, w) in A.A or in B.B. Clearly, the No Circuit Insertion Flow property on (G-H, G-L) keeps the oriented graph defined on W $\cup$ {v_0} by E-L $\cup$ E-H^- (E-L and E-H are the support arcs defined by F-H and F-G on W $\cup$ {v_0} according to the section II.2) from admitting any circuit. But the very definition of a Cut (property (P6)) also provides the arc set E-L $\cup$ E-H from defining any circuit. So it comes, that for any cut U of W, the double-flow (F-H, F-L) which we may compute this way on the network G(W $\cup${v_0}) through the (P8) equation and through application of the above process, is no circuit.

If we consider now some node element v of W, we may associate in a natural way a cut *Cut*(v) with v, by setting: *Cut*(v) = {v' $\in$ W such that v' σ v}. While searching for a best cut U in the general sense seems to be a difficult problem, we can easily scan the list σ and choose v_0 in such a way that an application to U_0 = *Cut*(v_0) of the Opt-Insertion Procedure in the sense of (P7) and (P8) yields the best possible result. So the Double-Flow algorithm is going to work this way, by applying the Opt-Insertion to a well-chosen cut Cut(v_0) as it has just been told, and by extending σ and τ to W $\cup${s, p, v_0} in such a way that they both remain linear extensions of respectively E-L $\cup$ E-H and E-L $\cup$ E-H^-.

The whole process may be summarized as follows:

Double-Flow Algorithm.
Input : the 2D-Item set V and the height number H.
Output: F-H, F-L, and L.
Initialization
W <- Nil; L <- 0; F-$L_{s, p}$ <- 0 ; F-$H_{s, p}$ <- H ; τ <- {s, p} ; σ <- {s, p}.
Main Loop.
While V – W $\neq$ Nil do
Randomly Pick up v_0 in V – W ; (I1)
Compute v_1 in W such that the application of the Opt-Insertion procedure to: (I2)

- X = W $\cup${s, s-aux, p, p-aux}; A = A(v_1) = *Cut*(v) $\cup${s, s-aux}; B = B(v_1) = X – A; << is the relation τ, extended in such a way that s (s-aux) is maximal (minimal) in A and that p (p-aux) is minimal (maximal) in B;
- for any x in *Cut*(v), Out-H(x) = $\Sigma_{y \in B}$ F-$H_{x,y}$ and Out-L(x) = $\Sigma_{y \in B}$ F-$L_{x,y}$;
- for any y in W – *Cut*(v), In-H(y) = $\Sigma_{x \in A}$ F-$H_{x,y}$ and In-H(y) = $\Sigma_{x \in A}$ F-$H_{x,y}$;
- Out-H(s) = $\Sigma_{y \in B}$ F-$H_{s,,y}$; Out-L(s) = 0; In-H(p) = $\Sigma_{x \in A}$ F-$H_{x,p}$; In-L(p) = 0;
- Out-H(s-aux) = In-H(s-aux) = 0; Out-L(s-aux) and In-L(s-aux) are to be minimized.

yields the best possible value S = Out-L(s-aux);
Let (u_1, S_1) be the (attach-node, value) pair provided by the related application of Opt-Insertion to *Cut*(v1);
Apply the Insertion(u_1) procedure on the input related to v_1 according to the instruction (I2);
Set L <- $\Sigma_{x \in Cut(v1)}$ G-$L_{s,x}$ + S_0;
Insert v_0 between v_1 and its successor Succ(v_1, σ, X) into σ;
Let w_1 be the largest element in A(v_1) in the sense of τ which provides F-L flow to v_0;
Let u'_1 (w'_1) be the largest (smallest) element in B(v_1) in the sense of τ which receives F-H (F-L) flow from v_1; Insert v_0 into τ in such a way that it becomes larger than w_1 and u'_1 and smaller than u_1 and w'_1 according to τ;
W <- W $\cup$ {x_0}.

We may state:

Theorem 4: *The above Double Flow procedure computes a feasible solution of the 2D-Bin Packing Problem with Length Minimization*.

Proof-Theorem. It is completely contained in the way we just described this procedure. *End-Proof.*

Remark: this algorithm implements a randomized greedy insertion scheme (Instruction (I1)), which enables us to integrate it into a Monte Carlo algorithmic scheme. On an other side a characteristic of this rather simple algorithm is that it works in a purely combinatorial way, without involving any numerical computation about paths and geometrical coordinates.. As a matter of fact, our goal here is not to really deal with optimization, but only to study the kind of techniques one may use in order to deal with multi-commodity flows submitted to a combinatorial constraint such that the no circuit constraint.

IV.3. Some Numerical Tests.

Though this procedure essentially intends to show the way one may handle the no circuit procedure, rather than efficiently tackling the 2D-Bin Packing problem with length optimization, we tried it on instances which were randomly generated, through the following Covering procedure, in such a way they were "perfect" for the 2D-Bin packing problem with length optimization, which means that their optimal value was equal to the quotient of the sum of the areas of the 2D-bins and H.

Covering Procedure:
Randomly Generate (uniform distribution) two integers H and L in [0,H-Max].[0, L-Max];
Not Full; x-cour <- 0; y-cour <-0; Bin-Number <- 0; Free-Area <- H.L;
While Not Full do
    Compute L-Rest, H-Rest which are respectively the length and the height of the largest rectangular free area with left bottom corner in (x-cour, y-cour);
Randomly generate h, and l in [0,H-Rest].[0, L-Rest];
Locate the 2D-item (h, l) into Rect(x-cour, y-cour, l, h);
Bin-Number <- Bin-Number + 1; Update Free-Area;
If Free-Area = 0 then Full Else Compute (x-cour, y-cour) in such a way that: x-cour (y-cour) is the smallest value such that Rect(0, 0, x-cour, H) (Rect(0, 0, x-cour + 1, y-cour)) is not completely covered by the currently inserted 2D-bins.

We tried several instance packages, every package made with 10 instances, all of them involving the same values H and L, and, for every instance package, we kept memory of the following quantities:

- N-item = the mean number of 2D-bins;
- H = the height of the area; L is the (predetermined) optimal length of the area;
- K = the number of times the Double-Flow procedure is launched on a same instance (Monte-Carlo process);
- T = the mean running/instance time of those K iterations of the Double-Flow procedure (in seconds);
- V = the mean gap between the optimal value of the instance and the best value obtained those K iterations of the Double-Flow procedure.
- I = the identifier of the instance package.

We provide here a result table related to 15 instance packages, of different sizes generated as described above: (and processed on PC IntelXeonwith 1.86 GHz, 3.25 Go Ram, while using a Visual Studio C++ compiler).

I	N-item	K	T (in sec)	V (in %)	H	L
1	15.2	10	0.010	2.7	10	10
2	26.7	10	0.010	4.8	10	20
3	29.3	10	0.021	5.0	10	20
4	68.5	10	0.235	8.9	20	40
5	58.4	10	0.143	7.4	20	40
6	15.5	100	0.047	1.7	10	10
7	26.0	100	0.098	2.2	10	20
8	29.9	100	0.216	4.6	10	20
9	68.4	100	2.342	7.5	20	40
10	58.2	100	1.432	6.2	20	40
11	15.6	1000	0.981	0.4	10	10
12	26.3	1000	2.165	0.7	10	20
13	29.2	1000	2.159	3.1	10	20

14	68.5	1000	23.420	3.5	20	40
15	58.1	1000	14.322	5.0	20	40

Comment: these results are not really good by themselves. It is not surprising, since our goal here was not to deal with optimization, but to study the kind of techniques one might use in order to deal with multicommodity flows submitted to a combinatorial constraint such that the no circuit constraint. As a matter of fact, we did not try here to turn the Double-Flow algorithm into a true optimization algorithm by introducing shortest paths or 2D-item corner coordinates. In that sense, it provides us with rather satisfactory results.

V. Conclusion.

What we just did here was establish a link between Flow Theory and a specific packing problem. It would be interesting to try to go further, and study the way extensions of the simple 2D-bin problem, involving larger dimensions or specific positioning constraints, may be cast into the flow formalism. Moreover, it would be interesting to study whether it is possible to take advantage form the flow Theory algorithmic machinery in order to derive efficient algorithms for those kinds of problems. That would essentially mean finding ways to deal with flow and multi-commodity flow vectors while taking into account a purely combinatorial constraint like the no circuit constraint.

VI. Bibliography.

[AHU95]. R.K.AHUJA, T.L.MAGNANTI, J.B.ORLIN, M.R.REDDY: "*Applications of network optimization*"; Chapter 1 of **Network Models, Handbook of Operation Research and Management Science** 7, p 1-83, (1995).

[AHU93]. R.V.AHUJA, T.L.MAGNANTI, J.B.ORLIN : « **Network Flows : Theory, Algorithms and Applications** », Prentice hall, Englewood Cliffs, N.J, (1993).

[BAL98]. A.BALAKRISHNAN, T.MAGNANTI, P.MIRCHANDANI: "*Designing hierarchical survivable networks*"; **Operat Research** 46, 1, p 116-130, (1998).

[BEN00]. W.BEN AMEUR : « *Constrained length connectivity and survivable networks* » ; **Networks** 36, 1, (2000).

[BIE96]. D.BIENSTOCK, O.UNLUK: "*Capacited network design: polyedral structure and computation*"; **INFORMS Journ of Computing** 8, p 243-259, (1996).

[BOS03]. M.BOSCHETTI, A.MINGOZZI: *The two dimensional finite bin packing problem, part 1: new bounds for the oriented case*; **4OR** 1, p 27-42, (2003).

[BUR04]. E.BURKE, G.KENDALL, G.WHITWELL: *A news placement heuristic for the orthogonal stock cutting problem*; **Operat. Research** 52, 4, p 655-671, (2004).

[CAP04]. A.CAPRARA, M.MONACCI: *On the two dimensional knapsack problem*; **Operations Research Letters** 32, p 5-14, (2004).

[CAP05]. A.CAPRARA, M.LOCATELLI, M.MONACCI: *Bilinear packing by bilinear programming* ; **IPCO 05 Conf. Proc. LNCS 3509, BERLIN, SPRINGER**, p 377-391, (2005).

[CHR77]. N.CHRISTOPHIDES, C.WHITLOCK: *An algorithm for two-dimensional cutting problems*; **Operations Research**, 25, p 30-44, (1977).

[CHR81] . N.CHRISTOPHIDES, C.A.WHITLOCK : « *Network synthesis with connectivity constraint : a survey* » ; **Operat Research**, p 705-723, (1981).

[CLA08]. F.CLAUTIAUX, A.JOUGLET, J.CARLIER, A.MOUKRIM: *A new constraint programming approach for the orthogonal packing problem*; *Computers and Operations Research* 35, p 944-959, (2008).

[COF84]. E.COFFMAN, M.GAREY, D.JOHNSON: *Approximation algorithms for bin packing: an updated survey*; In G.AUSIELLO, M.LUCERTINI, P.SERAFINI Ed, ***Algorithm design for computer system design***, New York, Springer, p 49-106, (1984).

[DAH94]. G.DAHL, M.STOER : « *A polyedral approach to multicommodity survivable network design* » ; **Numerisch Mathematik** 68, p 149-167, (1994).

[DEL02]. Dell'AMICO, S.MARTELLO, D.VIGO: *A lower bound for the non oriented two dimensional bin packing problem*; **Disc. Applied Math**. 118, p 13-24, (2002).

[ELH08] J.EL HAYEK, A.MOUKRIM, S.NEGRE: *New resolution algorithms and pretreaments or the two-dimensional bin-packing problem*; **Computers and Operations Research** 35, p 3184-3201, (2008).

[FAR03]. O.FAROE, D.PISINGER, M.ZACHARIENSEN: *Guided local search for the three-dimensional bin-packing problem*; **INFORMS Journal of Computing** 15, p 267-283, (2003).

[FEK04].S.FEKETE, J.SCHEPERS: *A combinatorial characterization of higher dimensional orthogonal packing*; **Mathematics of Operations Research** 29, p 353-368, (2004).
[FEK104]. S.FEKETE, J.SCHEPERS: *A general framework for bounds for higher dimensional orthogonal packing problems*; **Mathematics of Operations Research** 60, p 311-329, (2004).
[GAR79]. M.GAREY, D.JOHNSSON: **Computers and Intractability, a guide to the theory of NP-completeness*; New York*, *Freeman Ed***, (1979).
[HAD95]. E.HADJICONSTANTINOU, N.CHRITOFIDES: *An exact algorithm for general, orthogonal, two dimensional knapsack problem*; **Europ. Journ. of Operations Research**, 83, p 39-56, (1995).
[LOD99]. A.LODI, S.MARTELLO, D.VIGO: *Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems*; **INFORMS Journ. Computing** 11, p 347-357, (1999).
[LOD02]. A.LODI, S.MARTELLO, D.VIGO: *Recent advances on two dimensional bin packing problems*; **Disc. Applied Maths** 123, p 379-396, (2002).
[MAR98]. S.MARTELLO, D.VIGO: *Exact solution of the two-dimensional finite bin packing problem*; **Management Sciences** 44, p 388-399, (1998).
[MIN89]. M. MINOUX : *" Network synthesis and optimum network design problems : models, solution methods and application "*, **Networks** 19, p 313-360, (1989).
[MIR90]. P.B.MIRCHANDANI, L.R.FRANCIS: "***Discrete Location Theory***"; *.John WILEY* Eds, N.Y, (1990).
[PAR98]. P.M.PARDALOS, D.Z.DU: "*Network design: connectivity and facility location*"; **DIMACS Series 40**, *N.Y, American Math Society*, (1998).