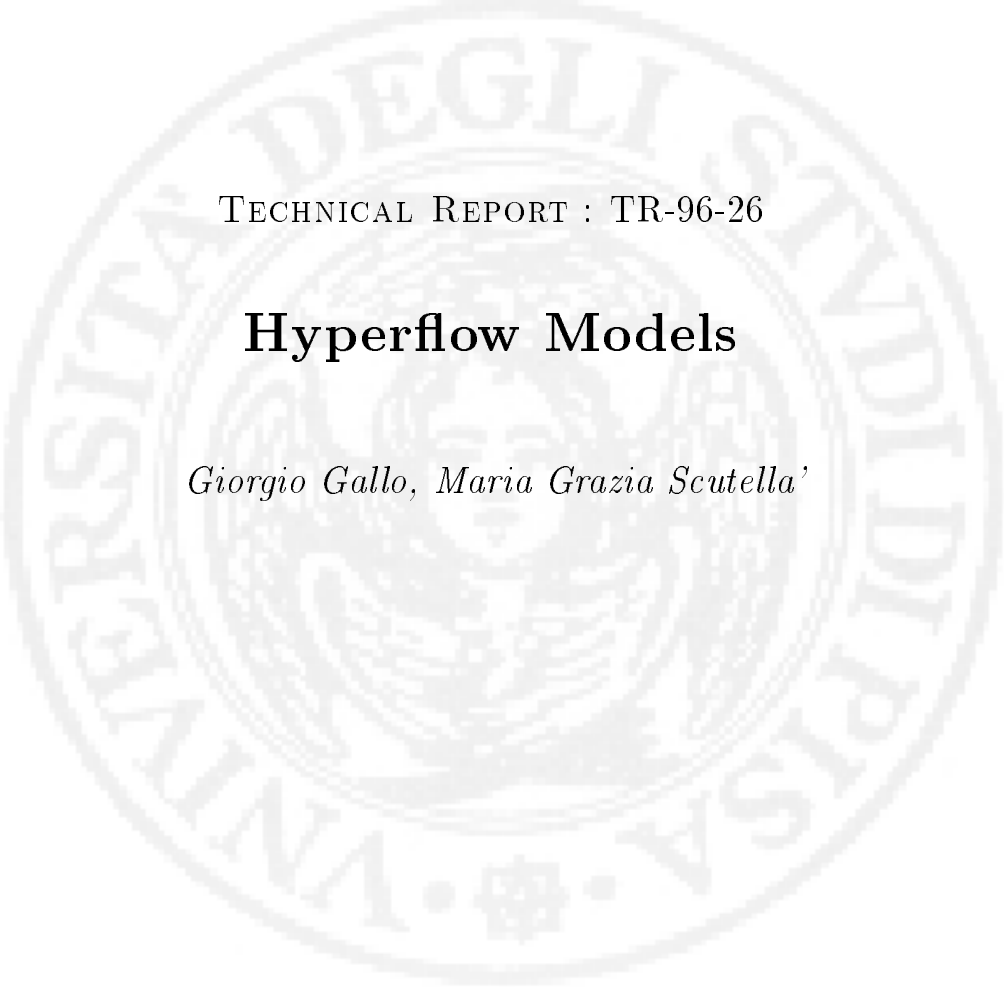
Università di Pisa

Dipartimento di Informatica



Technical Report : TR-96-26

Hyperflow Models

Giorgio Gallo, Maria Grazia Scutella'

July 01, 1996

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

1. Introduction

We consider the Minimum cost hyperflow problem, which generalizes the minimum cost flow problems on standard and on generalized graphs. This problem has a huge of very interesting applications, as will be described in this paper. Problems that can be modelled and solved in terms of hypergraph flows include both *Multicommodity flows* and applications in the area of flexible manufacturing systems, like the *One-Dimensional Cutting Stock Problem* and *Assembly Problems*. In particular, it is possible to model Assembly Problems with a given degree, say k, of parallelism, i.e. when k parallel identical machines are available for the execution of the operations. In addition, some *Economic applications* can be formulated as well in terms of minimum cost hyperflow problems.

In order to solve the Minimum cost hyperflow problem, a specialization of the Primal Simplex Algorithm, named the *Hypergraph Simplex Algorithm*, has been recently proposed, which relies on the characterization of the basis matrices of the Linear Programming formulation in terms of *spanning hypertrees*. In order to assess the efficiency of this approach, a wide experimentation has been performed on random hypergraphs and on some kinds of structured hypergraphs, as reported in this work. In the experimentation, a C implementation of the Hypergraph Simplex Algorithm has been compared with the primal version of CPLEX code, by obtaining rather interesting results. We observed in fact that, when the hypergraph size increases, our code runs faster than CPLEX. In particular, its solution time is a rather slow increasing function of the hypergraph size. This seems to suggest that the proposed approach works well on dense instances of large size.

The plan of the paper is the following. In Section 2, the main properties of the hyperflows are briefly reviewed, and the Hypergraph Simplex Algorithm is outlined. In Section 3, the results of a wide computational experimentation on random hypergraphs are presented, which show the efficiency of the approach, mainly on dense hypergraphs of large size. Finally, in Section 4, some interesting applications of hyperflows are described.

2. Hyperflows and their properties

A *directed hypergraph* (or simply *hypergraph*) is a pair $H=(V,E)$, where $V=\{v_1, v_2, ..., v_n\}$ is the set of nodes, and $E=\{e_1, e_2, ..., e_m\}$ is the set of hyperarcs. A *hyperarc* e is a pair (T_e,h_e), where $T_e \subseteq V$ is the *tail* of e and $h_e \in V \setminus T_e$ is its *head*[1]. Particular hyperarcs are the headless and the tailless

---

[1] The directed hypergraphs considered here are a particular case of the more general directed hypergraphs introduced by Gallo, Longo, Nguyen and Pallottino (1992).

hyperarcs, which are denoted as $(T_e,\varnothing)$ and $(\varnothing,h_e)$, respectively. While the size of a standard graph is uniquely defined by n and m, the size of a directed hypergraph depends also on the cardinality of its hyperarcs; so, the size of H is defined as the sum of the cardinalities of its hyperarcs:

$$size(\mathrm{H}) = \sum_{e_i \in E} |e_i|.$$

Given a hypergraph H=(*V*,*E*), a positive real *multiplier* $\mu_v(e)$ associated with $v \in T_e$, $\forall e \in E$, and a real *demand* vector b associated with V, a *flow on* H is a function $f: E \rightarrow R$ which satisfies the following conservation constraints:

$$\sum_{v=h_e} f(e) - \sum_{v \in T_e} \mu_v(e) f(e) = b(v), \quad \forall v \in V \qquad \text{(conservation)}.$$

The flow is *feasible* if it satisfies the following capacity constraints, where u is an *upper capacity* vector associated with E:

$$0 \le f(e) \le u(e), \quad \forall e \in E \qquad \text{(capacity)}.$$

Let $c(e)$ be the cost associated with the hyperarc e, $\forall e \in E$. Then the *Minimum cost hyperflow problem* is to find a feasible flow on H which minimizes the function $\sum_{e \in E} c(e) f(e)$. If $A=[A(e_1), A(e_2), .., A(e_m)]$ is the nxm incidence matrix of H, where by $A(e)$ we denote the column whose elements are

$$A_v(e) = \begin{cases} -\mu_v(e) & \text{if } v \in T_e \\ +1 & \text{if } v = h_e \\ 0 & \text{otherwise} \end{cases},$$

then the problem can equivalently be stated in the following matrix formulation:

$$\min \sum_{e \in E} c(e) f(e) \qquad (*)$$
$$A f = b$$
$$0 \le f(e) \le u(e) \quad \forall e \in E.$$

Note that A has one positive component at most in each column, then it is a *pre-Leontief* matrix (Veinott, 1968). The fact that in A the positive component is 1 is not a loss of generality, since it can always be obtained by scaling the variables.

(*) Whenever possible, we will not define explicitly a vector as a row or a column vector, and use it in both senses in the matrix multiplications.

If $\pi(v)$ denotes the *potential* which is associated with node v, and $\lambda(e)$ is the dual variable associated with hyperarc e, then the dual problem, named the *Potential problem on hypergraphs*, can be formulated as follows:

$$\max \sum_{v \in V} b(v)\pi(v) - \sum_{e \in E} \lambda(e)u(e)$$

$$\pi(h_e) - \sum_{v \in T_e} \mu_v(e)\pi(v) - \lambda(e) \le c(e), \ \forall e \in E$$

$$\lambda(e) \ge 0, \ \forall e \in E.$$

The Minimum cost hyperflow problem is a very general problem, which extends the uncapacitated minimum cost flow problem on hypergraphs studied by Jeroslow et al. (1992), and which includes all the bounded linear programming problems. Consider in fact a generic bounded linear programming problem and its constraint matrix A; it is sufficient to replace each column $A(j)$ with a column $A^+(j)$, having the positive coefficients of $A(j)$, with a column $A^-(j)$, having its negative coefficients, and with a constraint equating the two corresponding variables. The substitution for the variable corresponding to $A^+(j)$ by the slack in its upper bound constraint produces in fact a direct hypergraph format.

In order to solve the minimum cost hyperflow problem, a *Hypergraph Simplex Algorithm* has been proposed in [Cambini, Gallo & Scutellà, 1992; 1996]. This algorithm relies on the characterization of the basis matrices of the Linear Programming formulation in terms of *spanning hypertree*. In order to introduce this hypergraph structure, which generalizes the spanning tree for standard graph, let us informally provide the concepts of hyperpath and of hypertree.

A *directed path* P_{st} from s to t in $\mathtt{H}$ is a sequence $P_{st}=(v_1=\{s\}, e_1, v_2, e_2, ..., e_q, v_{q+1}=\{t\})$, where $s \in T_{e_1}$, $h_{e_q}=t$ and $v_i \in \{h_{e_{i-1}}\} \cap T_{e_i}$, $i=2, .., q$. If $s=t$, then P_{st} is a *directed cycle*. When no directed cycle exists, then $\mathtt{H}$ is said to be *cycle-free*.

A *directed hyperpath*, Π_{St}, from the source set S to the sink t in $\mathtt{H}$ is a minimal cycle-free sub-hypergraph containing both the nodes in S and node t, and such that each node, with the exception of the nodes in S, has exactly one entering hyperarc.

A hyperarc e' is said to be a *permutation* of a hyperarc e if $T_e \cup \{h_e\} = T_{e'} \cup \{h_{e'}\}$. A hypergraph $\mathtt{H}'$ is a *permutation* of a hypergraph $\mathtt{H}$ if its hyperarcs are permutations of the hyperarcs of $\mathtt{H}$.

A hypergraph Π'_{St} is said to be an *undirected hyperpath* from the source set S to the sink t if among its permutations one exists which is a directed hyperpath from S to t.

Examples of directed and of undirected hyperpaths are shown in Figure 1. The undirected hyperpath, on the right, is a permutation of the directed one, on the left. Both hyperpaths have $S=\{1, 2\}$ as the source set, and $t=6$ as the sink. Note that an undirected hyperpath may have directed cycles.

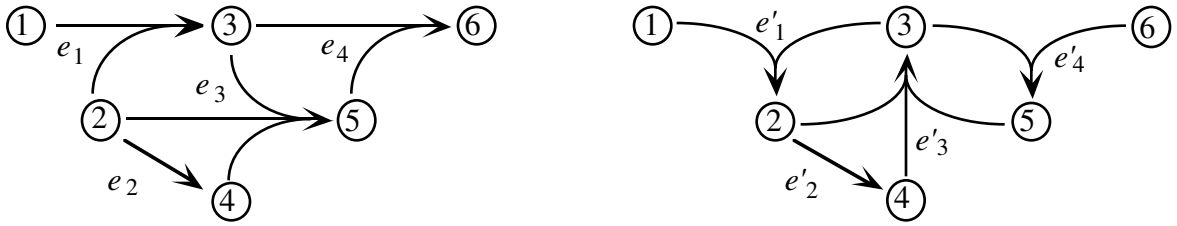


Fig. 1

The definition of hyperpath can be extended as follows. We say that $\mathtt{T}_R=(R \cup N, E)$ is a *directed hypertree* (or simply *hypertree*) rooted at R if:

(i) it is cycle-free,

(ii) $R \cap N=\varnothing$,

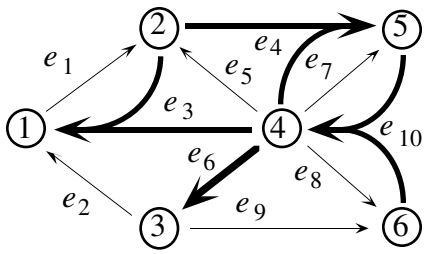
(iii) each node $v \in N$ has exactly one entering hyperarc and no hyperarc has a node of R as its head.

The set R is called the *root set* of $\mathtt{T}_R$, while N is the set of *non-root* nodes. Any non-root node v not contained in the tail of any hyperarc of $\mathtt{T}_R$ is a *leaf* of the hypertree. From the definition it follows that, for any $v \in N$, there is a unique directed hyperpath Π_{Rv} in $\mathtt{T}_R$.

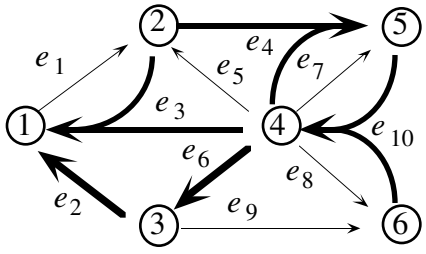
An *undirected hypertree* rooted at R is any permutation of a directed hypertree rooted at R. In the case of undirected hypertrees, a leaf is a non-root node which belongs to exactly one hyperarc.

Now, we have the ingredients to extend the concept of spanning tree to hypergraphs. A *spanning hypertree of* $\mathtt{H}=(V,E)$ is an undirected hypertree $\mathtt{T}_R=(V,E_T)$ such that $E_T \subseteq E$ and $(T_e \cup \{h_e\}) \not\subseteq R$, $\forall e \in E \setminus E_T$. In this definition, the property $(T_e \cup \{h_e\}) \not\subseteq R \quad \forall e \in E \setminus E_T$ is to impose a kind of "minimality" of the root set R; in fact, if a hyperarc e exists such that $(T_e \cup \{h_e\}) \subseteq R$, we can use it to span one of the nodes of R, and decrease by one the cardinality of the root set. The hyperarcs in E_T are called *tree* hyperarcs, whereas the ones in $E_X=E \setminus E_T$ are said *external* hyperarcs.

Examples of spanning hypertrees are shown in Figure 2, where, with a little abuse of notation, a hypertree is represented by means of one of its top-down traversals, i.e. a traversal from the roots toward the leaves. Note that, for a given hypergraph with n nodes, it is $|R| + |E_T| = n$ for all spanning hypertrees, whereas the cardinalities of R and E_T may vary from hypertree to hypertree.



$\mathtt{T}_R$ =({5,6}, e_{10}, 4, e_4, 2, e_3, 1, e_6, 3)



$\mathtt{T}_{R'}$ =({1}, e_2, 3, e_6, 4, e_3, 2, e_4, 5, e_{10}, 6)

Fig. 2

In the Minimum Cost Hyperflow Problem, each basis M corresponds to a pair $(\mathtt{T}_R, E_X)$ characterized as follows. $\mathtt{T}_R$ is a *spanning hypertree* of the sub-hypergraph $\mathtt{H}*$ corresponding to M ($\mathtt{H}*$ contains the hyperarcs whose indices are in M). E_X is the set of the *external hyperarcs*, i.e. the basic hyperarcs out of the spanning hypertree; its cardinality is the same as the cardinality of the root node set of $\mathtt{T}_R$.

The correspondence between bases and pairs $(\mathtt{T}_R, E_X)$ is not a one-to-one correspondence. In fact, the above structure corresponds to a basis if and only if a particular matrix, called the *root matrix*, and denoted by M_R, is proved to be non-singular. The root matrix has the same size as the root node set and the external hyperarc set. Moreover, both M_R and its inverse, say M_R^{-1}, can be completely characterized in terms of hypergraph concepts; in fact, a characterization of their rows and of their columns in terms of flows and of dual variables has been provided. The hypergraph basis characterization generalizes the one for "standard" and for "generalized" minimum cost flows, where bases correspond to spanning trees with empty or with singleton root node set.

By exploiting the basis characterization in terms of spanning hypertrees, the Hypergraph Simplex Algorithm solves linear systems of type $Mf=\overline{b}$ and $\pi M=\overline{c}$, where M is an nxn basis and $\overline{b}$ and $\overline{c}$ are n-vectors, directly on the hypergraph. This is obtained by performing labelling procedures on the basis spanning hypertree, and by properly using the root inverse matrix and its hypergraph characterization. Also the determination of the leaving hyperarc and the updating of the basis hypergraph structure and of the inverse of the root matrix are performed directly on the hypergraph at each step. In particular, the updating of the root inverse matrix exploits the characterization of the rows and of the columns of M_R^{-1} in terms of flows and of potentials, respectively. Like for the network simplex algorithms for the standard and for the generalized minimum cost flow problems, most of the computations performed at each pivot operation have thus direct hypergraph interpretations. For

formal proofs concerning the correctness of the basis characterization and of the entire Hypergraph Simplex Algorithm, we refer to the above mentioned works.

3. Experimental results

The Hypergraph Simplex algorithm has been implemented in ANSI-C language, and tested on a workstation Hewlett Packard HP 9000/710 with 64 Mb RAM under the HP-UX V.9.8 Operating System. The implementation makes use of "ad-hoc" data structures for representing the input hypergraph and the current basic feasible solution. In addition, crucial aspects concern the choice of the initial feasible basis, the strategy used for determining the entering hyperarc at each pivot step (*pricing*), the dual solution updating and a periodical updating of the current basic spanning hypertree (*basis representation updating*). A detailed description of the implementation choices can be found in [Gallo, Licheri & Scutellà, 1995]. Here, we present the results of a wide experimentation performed on random hypergraphs. Experimental results concerning some kinds of structured hypergraphs will be described in the next section.

In order to assess the efficiency of the code, named *HySimplex*, we have used random hypergraphs with (0,±1) incidence matrices, and we have compared *Hysimplex* with one of the best state-of-the art LP codes, CPLEX (Version 3.0). This is a commercial package for solving LP problems, which provides both Simplex implementations of type "primal", "dual" and "network", and the possibility of combining these kinds of approaches. For more details, we refer to the original CPLEX Libraries.

By fixing the number of nodes (n=400), random hypergraphs with (0,±1) incidence matrix have been generated for increasing values of the number of hyperarcs, m, and of the maximum tail set cardinality, say k. For each problem typology (n, m, k), different instances have been generated by varying the hyperarc costs and the upper capacities in the range [30,100] and [1,100], respectively. Each instance has been solved both by *HySimplex* and by the primal version of CPLEX code. About 360 hypergraphs have been obtained by generating 15 different m-values within the range [800,12000], and 8 different k-values within the range [4,200]. The computational results are summarized in Figures from 3 to 10.

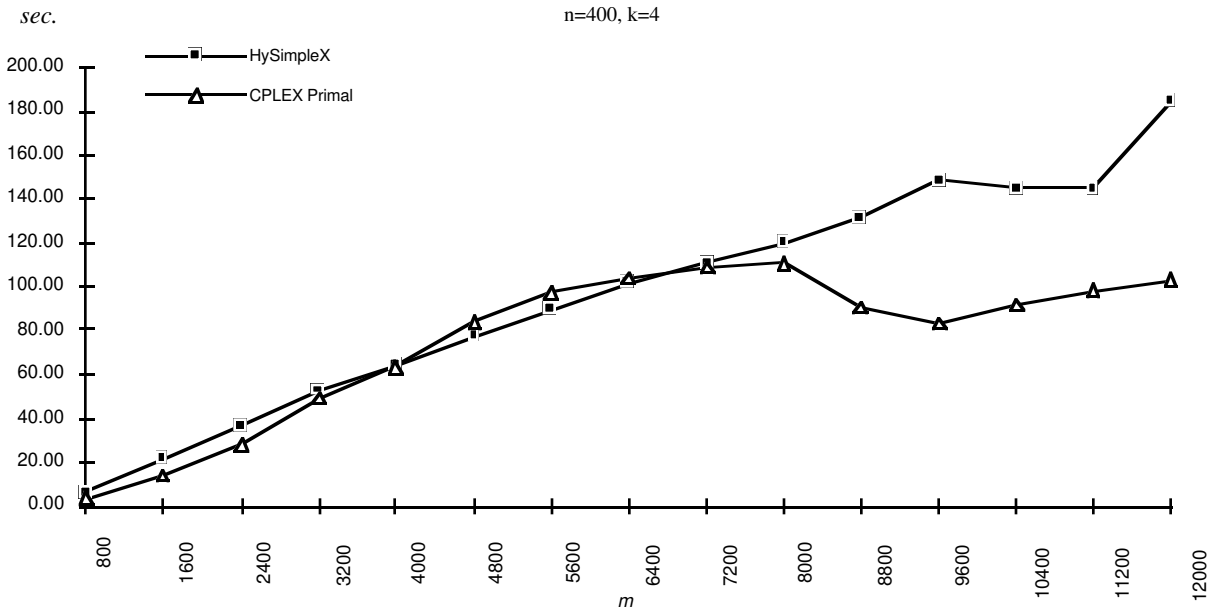

sec.
n=400, k=4
HySimpleX
CPLEX Primal
200.00
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
12000
m

Fig. 3

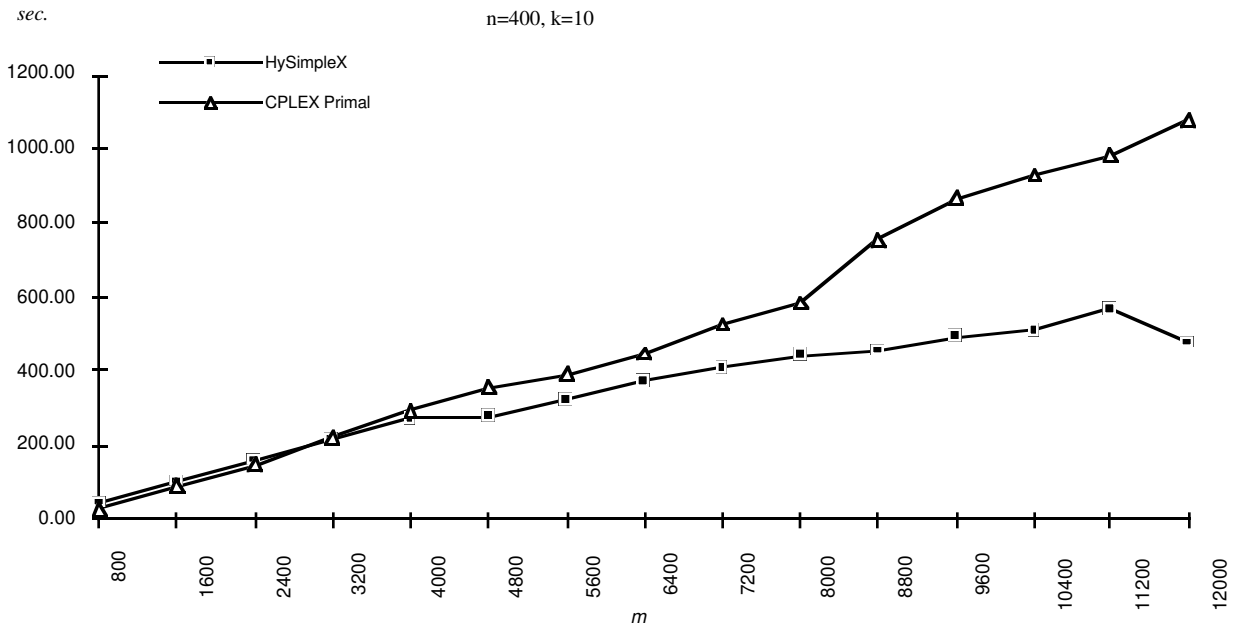

sec.
n=400, k=10
HySimpleX
CPLEX Primal
1200.00
1000.00
800.00
600.00
400.00
200.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
12000
m

Fig. 4

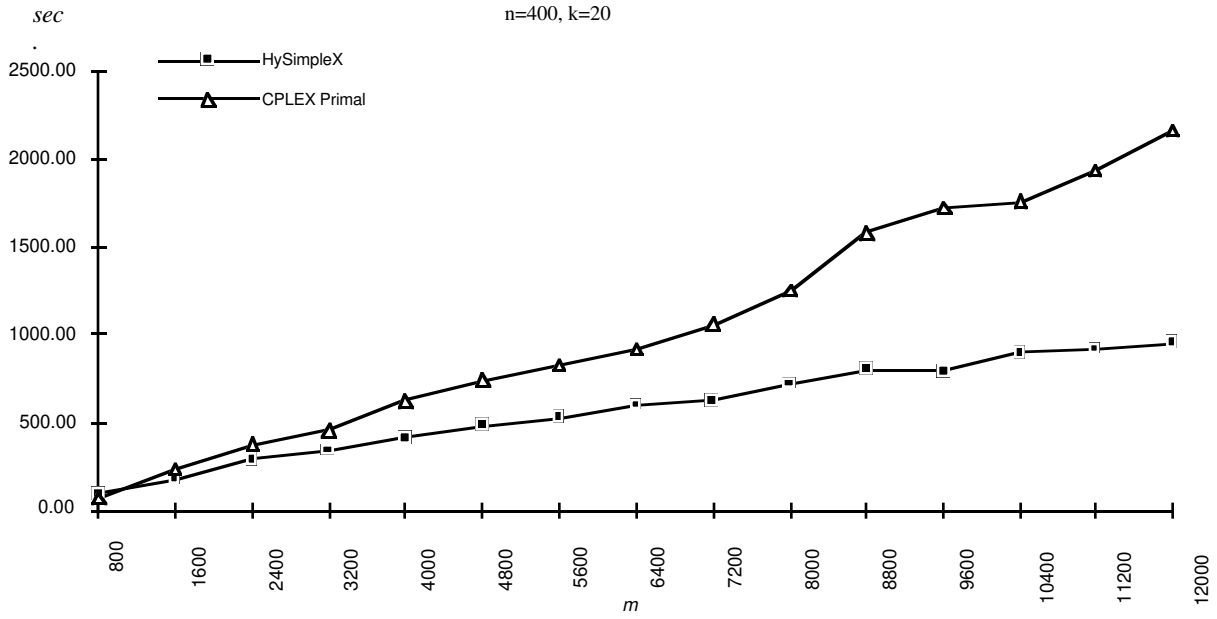

sec
.
n=400, k=20
HySimpleX
CPLEX Primal
2500.00
2000.00
1500.00
1000.00
500.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
12000
m

Fig. 5

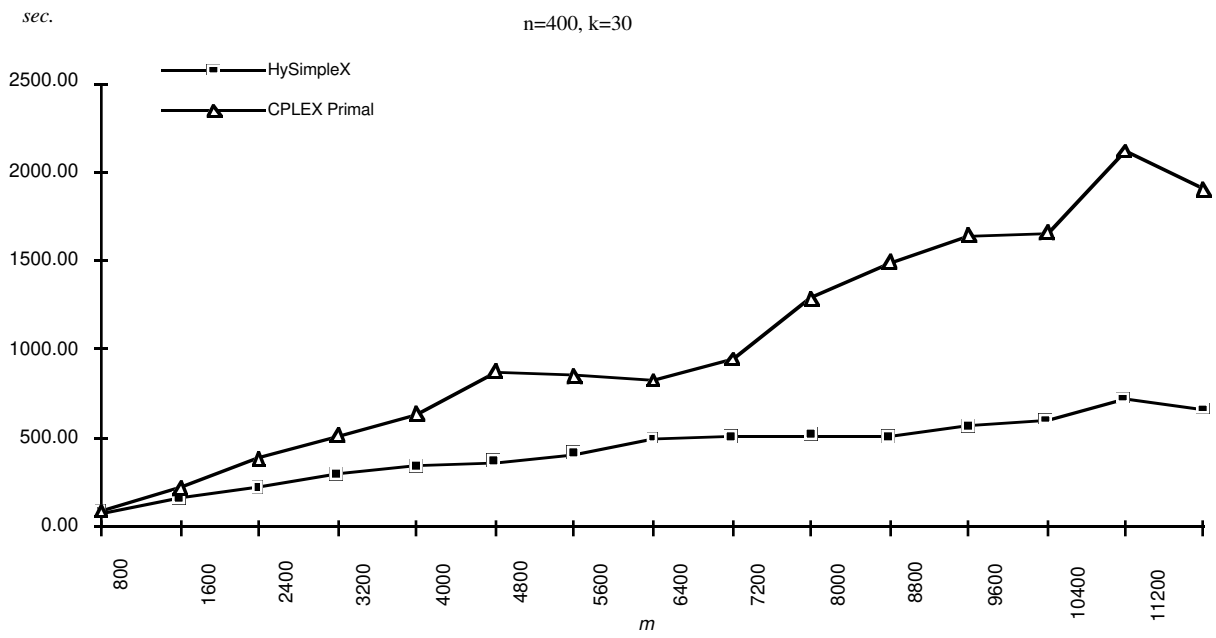

sec.
n=400, k=30
HySimpleX
CPLEX Primal
2500.00
2000.00
1500.00
1000.00
500.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
m

Fig. 6

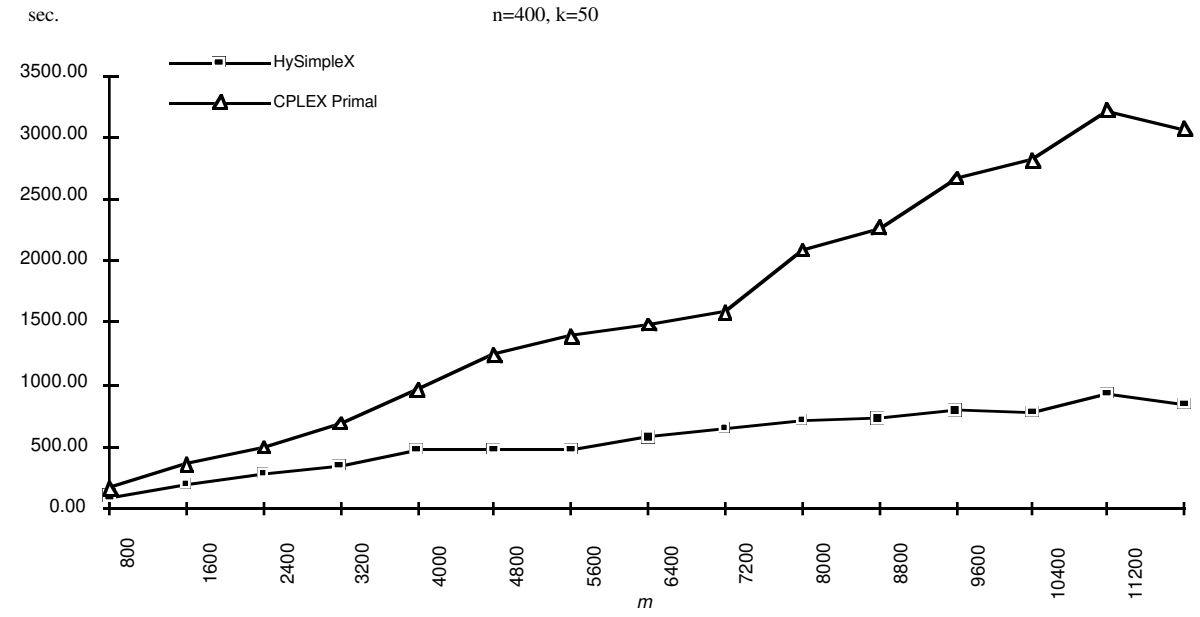

sec.
n=400, k=50
HySimpleX
CPLEX Primal
3500.00
3000.00
2500.00
2000.00
1500.00
1000.00
500.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
m

Fig. 7

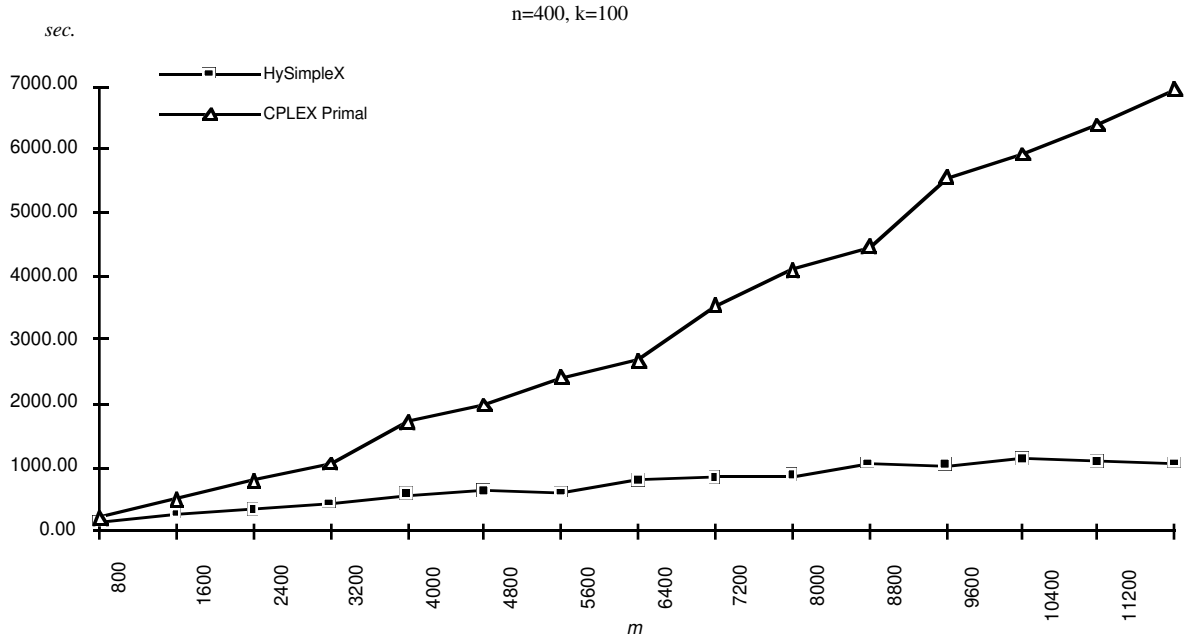

n=400, k=100
sec.
HySimpleX
CPLEX Primal
7000.00
6000.00
5000.00
4000.00
3000.00
2000.00
1000.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
m

Fig. 8

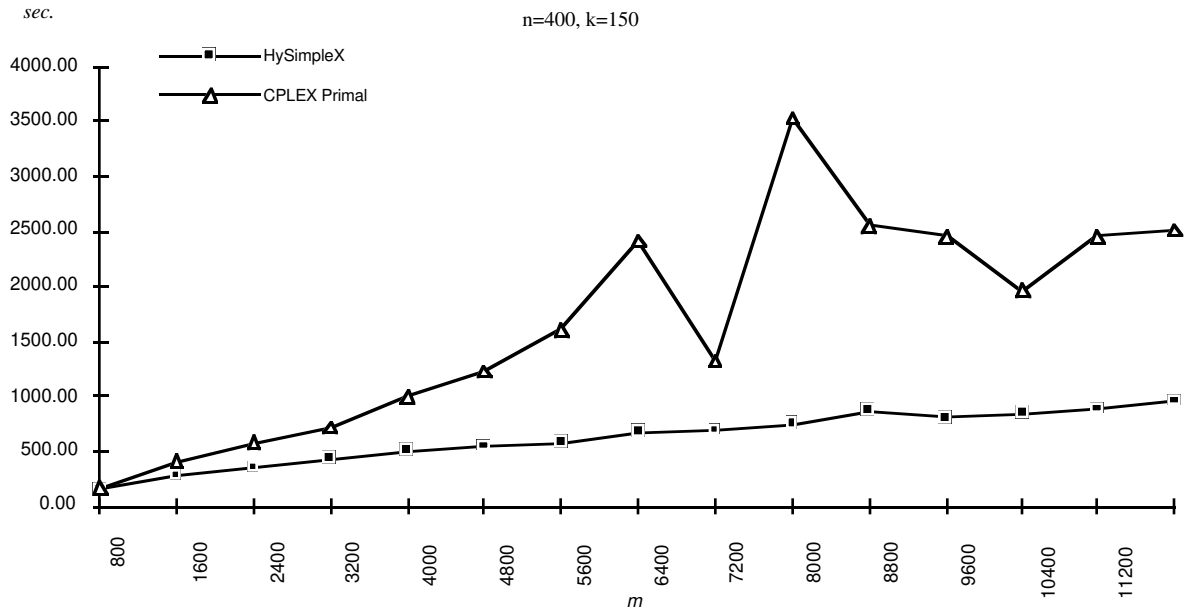

sec.
n=400, k=150
HySimpleX
CPLEX Primal
4000.00
3500.00
3000.00
2500.00
2000.00
1500.00
1000.00
500.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
m

Fig. 9

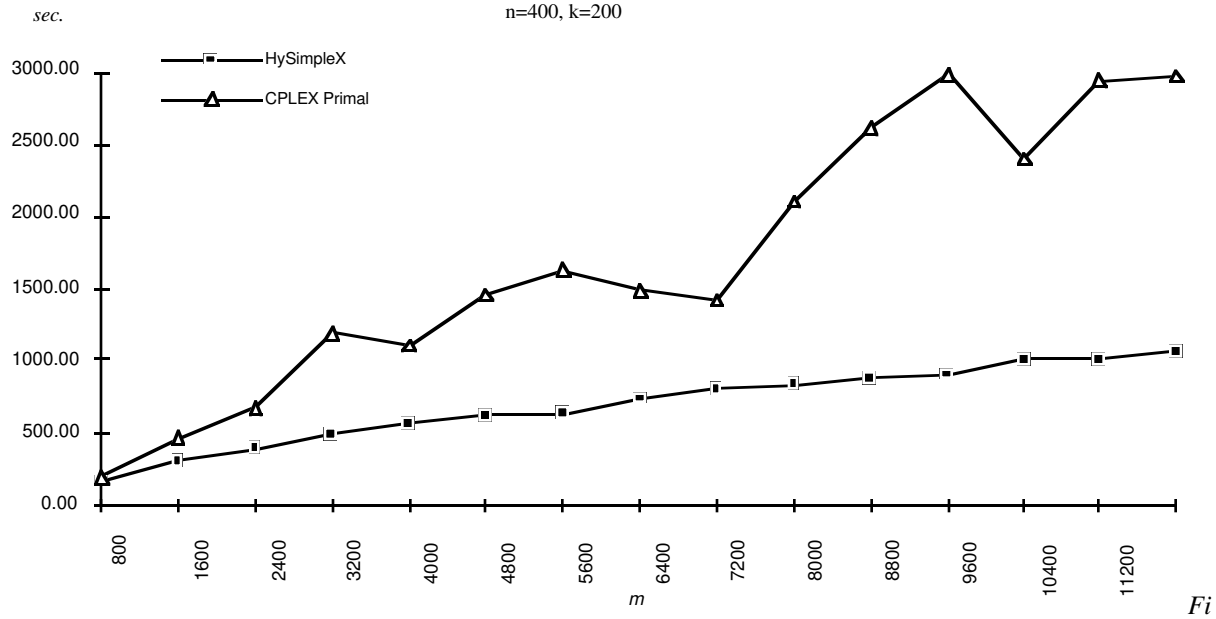

sec.
n=400, k=200
HySimpleX
CPLEX Primal
3000.00
2500.00
2000.00
1500.00
1000.00
500.00
0.00
800
1600
2400
3200
4000
4800
5600
6400
7200
8000
8800
9600
10400
11200
m

Fig. 10

As graphs reveal, for small instances, characterized by small values of m and of k, the two algorithms present a similar performance; in some cases CPLEX runs faster, thanks to its pricing strategy (usually the Devex one), which allows it to limit the total number of simplex iterations. However, when m and k increase, *HySimplex* runs faster than CPLEX. In particular, while the *HySimplex* solution time is a rather slow increasing function of the hypergraph size, both the number and the running time of the CPLEX iterations increase with the size of the instances. This seems to suggest that *HySimplex* works well on dense instances of large size.

4. Applications

Here, we will describe a variety of hypergraph models, which are all special cases of the Minimum cost hyperflow problem presented in Section 2. These models include Multicommodity flows, applications into the area of flexible manufacturing systems like the Cutting Stock Problem and the Assembly Problem, and some Economic applications.

4.1. Multicommodity flows

Given a directed graph $G=(N,A)$, assume to have h flow demand vectors associated with the nodes of G, say b^1, b^2, .., b^h, and h upper capacity and cost vectors associated with the arcs of G, say u^1, u^2, .., u^h and c^1, c^2, .., c^h, respectively. The *Multicommodity Flow Problem* consists in finding h feasible flows or *commodities* on G, say $f^1, f^2, ..., f^h$, each one relative to the flow demand vector b^i and to the upper capacity vector u^i, $i= 1, ..., h$, in such a way to minimize the total flow cost, i.e. $\sum_{i=1}^{h} c^i f^i$. Global upper capacities $u(v,w)$ are associated with each arc $(v, w)\in A$ as well, stating that the overall flow along (v,w) must be bounded by $u(v,w)$. The h commodities are thus required to satisfy these global upper capacity constraints.

The problem can be formulated as a Minimum Cost Hyperflow Problem on a hypergraph $\mathtt{H}_G$ defined as follows. Firstly, for each node v and for each commodity i in G, let us introduce a node v_i in $\mathtt{H}_G$, representing node v when traversed by commodity i. In addition, for each arc (v,w) in G, let us introduce a node v_w in $\mathtt{H}_G$, whose meaning will be explained below. The first type of nodes of $\mathtt{H}_G$ correspond to the original nodes of G: node v_i has in fact a flow demand $b^i(v)$, i.e. the original demand of node v with respect to commodity i. On the other hand, the nodes of type v_w are transhipment nodes.

As far as the hyperarcs of H_G are concerned, for each arc $e=(v,w)\in A$ and for each commodity i, let us introduce a hyperarc e_i in H_G going from the nodes v_i and v_w to the node w_i. In addition, for each node of type v_w, let us introduce a tailless hyperarc entering v_w.

Then, if we associate the upper capacity $u^i(v,w)$ and $u(v,w)$ with $e_i=(\{v_i, v_w\}, w_i)$ and with $(\emptyset, v_w)$, respectively, each feasible flow on H_G corresponds to a set of h feasible commodities on G which satisfy the global upper capacity constraints. So, if the cost $c^i(v,w)$ is associated with hyperarc e_i, while the tailless hyperarcs of H_G are given zero cost, the Minimum cost hyperflow problem on H_G models the Multicommodity flow problem on G. Note that H_G has $(m + nh)$ nodes and $(mh + m)$ hyperarcs with maximum cardinality of the tail sets ≤ 2. Moreover, H_G is a particularly structured hypergraph.

The proposed hypergraph model is illustrated in Figure 11. By assuming h=2, i.e. a two-commodity flow problem, each arc (v, w) in the original graph G is modelled by the following sub-hypergraph of H_G:

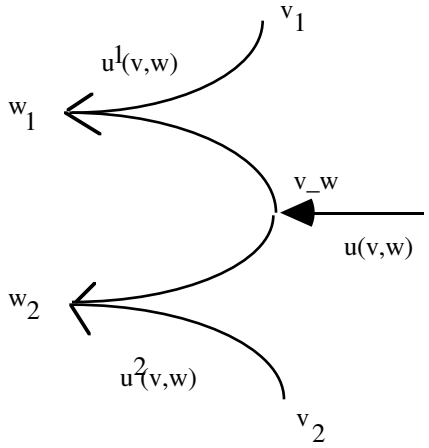


Fig. 11

In our experimentation, we have studied the *HySimplex* behavior also on the structured hypergraphs which model multicommodity flow in terms of hyperflows. In particular, we have run a set of test instances obtained from C.R.T. of Montreal, which are quite sparse, and we have compared the computational results obtained by our code both with the primal version of CPLEX code, and with an "ad hoc" dual code, named MMCFB, which is based on cost decomposition and which uses bundle techniques for solving the Lagrangean dual problem (see [Capitani, Catoni, Frangioni, Gallo&Saviozzi, 1995}). The algorithm behavior is illustrated by graphs, where the products $m*h$, estimating the number of hyperarcs into the hypergraph formulation, are reported on the x-axis, while

the computational times are reported on the *y*-axis by using a logarithmic scale. These times are the ones required for solving the problem instances, i.e. the transformation times are not included here (in Figure 12, the computational times of CPLEX Primal are not reported since this code has shown to be very slow with respect to the other codes (more precisely, its computational times vary between 74 and 163 seconds)).

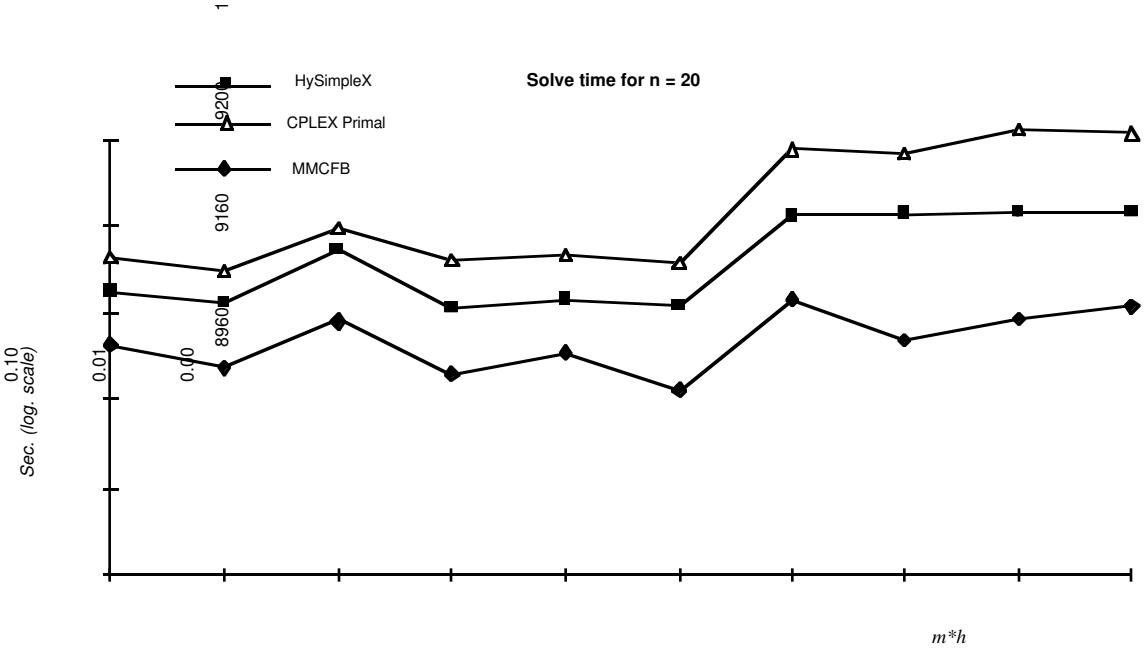


Fig. 12

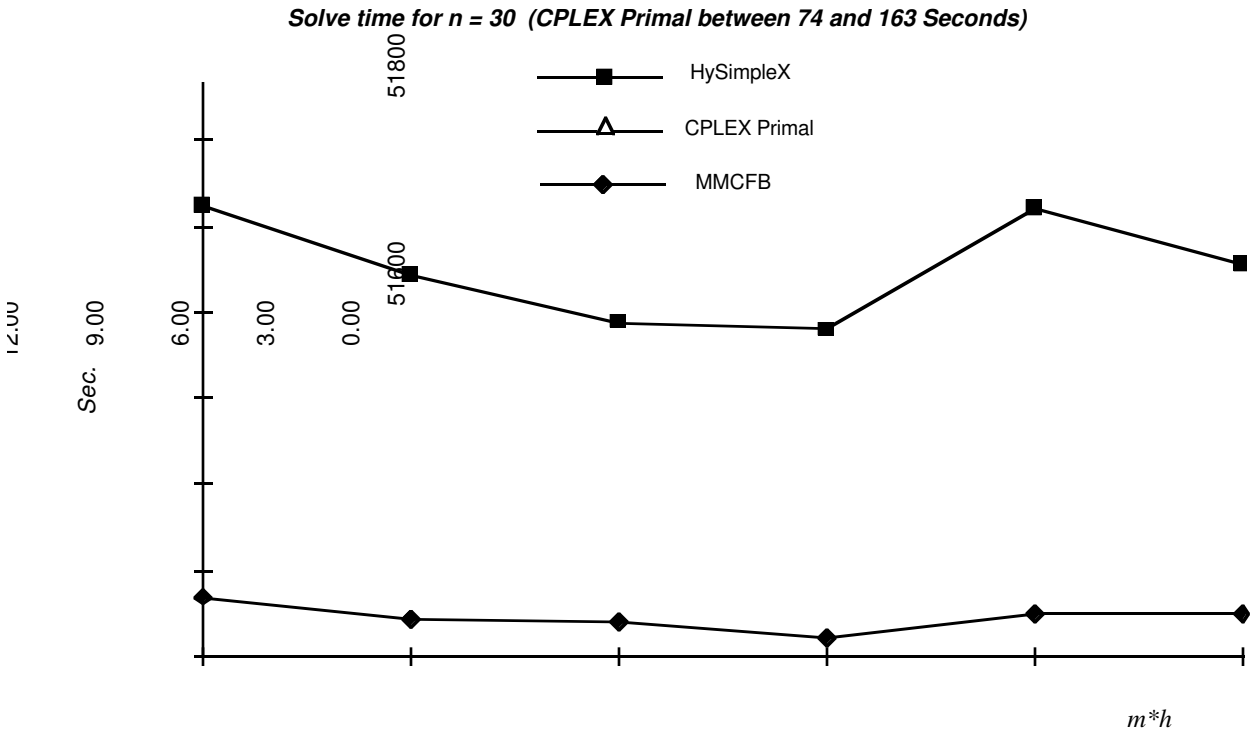

Solve time for n = 30 (CPLEX Primal between 74 and 163 Seconds)
HySimpleX
CPLEX Primal
MMCFB
Sec.
12.00
9.00
6.00
3.00
0.00
51800
51600
m*h

Fig. 13

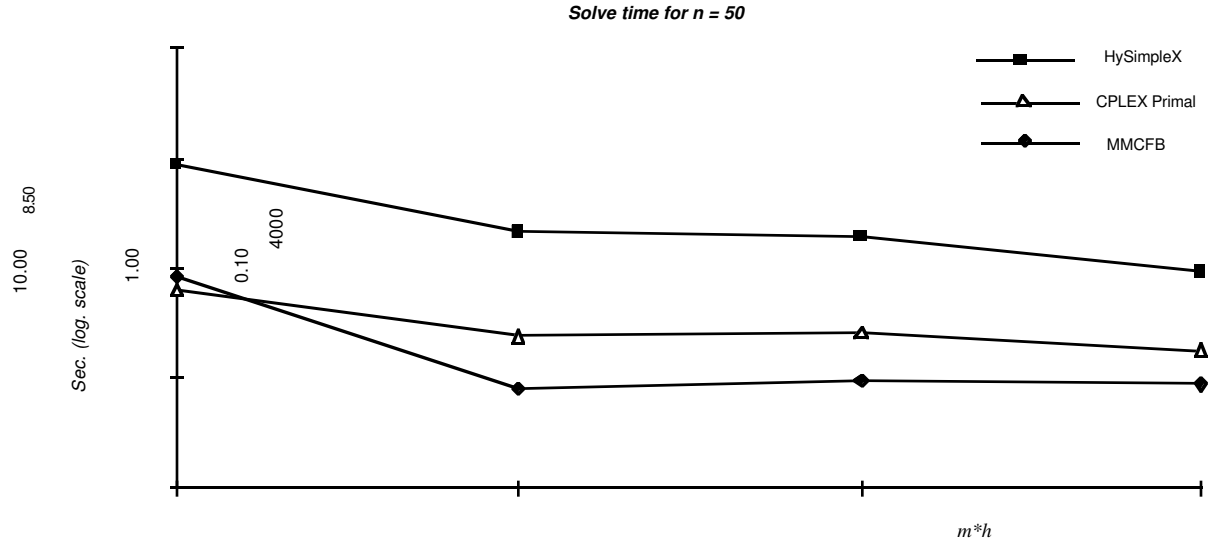


Fig. 14

It is possible to observe that, over all the considered instances, *HySimple* is in general faster than the Primal CPLEX code, according to the results obtained for random hypergraphs, while the "ad hoc" code MMCFB has the best behavior. Also in this case, however, *HySimplex* is competitive, and its solution time is a rather slow increasing function of the hypergraph size.

4.2. Cutting Stock

In this section, we consider the One-Dimensional Cutting Stock Problem (CSP), and show how to model its linear programming relaxation in terms of hyperflows. Based on this hyperflow formulation, a hypergraph simplex algorithm has been developed in [Gambale, Nonato, Scutellà, 1995]. The general Hypergraph Simplex algorithm has in fact been tailored to capture the particular structure of the CSP hypergraphs. Preliminary results, obtained by means of a LISP implementation of the algorithm which has been developed for an Italian steel factory, are quite interesting: large instances are solved in rather short time, often providing an integer solution; moreover, stock requirements seem to constitute a critical factor just up to a certain value, over which they become of no influence. In addition, the hyperflow model for CSP is very flexible, since it allows to deals with multiple standard lengths, to add stock lengths and capacity constraints, and to manage cutting knife limitations. All these results suggest that an approach based on hyperflows is an innovative and valid alternative to

standard LP codes for facing and solving the linear relaxation of CSP, mainly when stock requirements are very large. A wider experimentation, which includes a C++ implementation of the algorithm and a comparison with more classical approaches, is in progress and will be the object of a subsequent work. Here, the Cutting Stock Problem and the corresponding hypergrflow model are described. For more details about the Hypergraph Simplex algorithm for CSP and its LISP implementation we refer to the above mentioned paper.

Consider an unlimited number of *pieces* of stock, characterized by different *stock lengths* s_k, k=1, .., K, and assume to have b_i requirements of stock with *demand length* d_i, i= 1, .. I (all the input data are assumed to be integer). The One-Dimensional Cutting Stock Problem consists of cutting the pieces of stock in such a way that the demand length requirements are satisfied, while the overall residual length is minimized (in alternative formulations, the total number of the used pieces of stock is minimized).

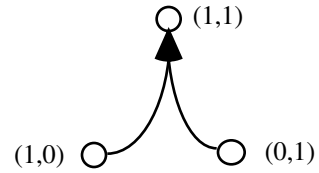
Consider now the linear programming relaxation of CSP. The classical approach to formulate and solve this relaxation is based on the definition of cutting patterns. A *cutting pattern* (or simply *pattern*) is an integer vector $(a_1, a_2, .., a_I)$, denoting that a piece of stock of length s_k will be cut in such a way that exactly a_i pieces of demand length d_i, i= 1, .. I, will be obtained. An *elementary pattern* is a cutting pattern having 1 in one only position, and 0 otherwise; hereafter, we shall denote the i^{th} elementary cutting pattern, having 1 in position i, by P^i, i= 1, .. I. A cutting pattern $(a_1,a_2, .., a_I)$ is said to be *feasible* if $\sum_i a_i d_i$ is less than or equal to some stock length s_k, producing a leftover of $s_k - \sum_i a_i d_i$.

The approach described here relies on the concept of cutting pattern as well. The innovative aspect is that the linear programming relaxation is rewritten and solved in terms of hyperflows. Given P as an instance of CSP, let us associate a hypergraph H_P with it in the following way. Firstly, let us impose a one-to-one correspondence between feasible patterns of P and nodes of H_P. In order to define the hyperarc set, assume that the order relation $d_1 < d_2 < .. < d_I$ holds among the demand lengths. Then, each feasible pattern $A=(a_1, a_2, .., a_j , 0,..0)$, where $a_j, j \le I$, is the maximum (with respect to the given ordering) nonzero component, can be uniquely defined according to the following recursive rule:

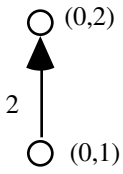
$$(a_1, a_2, .. , a_j, 0,...0) = (a_1, a_2, .. , a_j -1, 0,...0) + P^j, \qquad (*)$$

where P^j is the j^{th} elementary pattern. Therefore, by recoursively applying rule (*), each pattern A can be viewed as the result of the successive composition of elementary patterns, according to the ordering previously introduced.

Based on rule (*), for each feasible pattern A let us introduce a hyperarc of H_P having the node corresponding to A as the head, and the nodes corresponding to the two patterns in (*) as the tail set nodes. This hyperarc represents the composition rule (*) relative to A. For instance, let s=(4,5,8) be the list of the stock lengths and d=(2,3) be the ordered list of the demand lengths; then the composition rule relative to the feasible pattern A=(1,1), obtainable from the stock length 5, is described by means of the following hyperarc:



From the above definitions, it follows that exactly one hyperarc enters each node of H_P, with the exception of the nodes corresponding to elementary patterns, for which no entering hyperarc exists. Moreover, all these entering hyperarcs have two nodes into the tail set, at least one of which is an elementary pattern, except for the case in which (a_1, a_2, ... , a_j -1, 0,...0) coincides with P^j in rule (*). In this case, the hyperarc having the node corresponding to P^j as the unique tail node and the one corresponding to (a_1, a_2, .. , a_j, 0,...0) as the head node is inserted into H_P; in order to model (*), a multiplier equal to 2 is associated with the hyperarc, denoting that two occurencies of the elementary pattern P^j need to be used. These particular hyperarcs will be also referred to as *multiple* hyperarcs. For instance, in the sample instance previously introduced, the feasible pattern (0,2)=(0,1)+(0,1) is modeled by the following multiple hyperarc:



Let us further investigate the properties of H_P. By defining as $Level_k$ the set of patterns (or, equivalently, nodes) A such that $\sum_i a_i = k$, H_P can be viewed as a layered graph; in fact, for each node v in $Level_k$, the nodes in the tail set of the unique hyperarc entering v belong to $Level_{k-1}$ and to $Level_1$, where $Level_1$ is the set of the nodes corresponding to the elementary patterns. H_P is thus acyclic. This property is illustrated in Figure 15, where a portion of the "sample instance" hypergraph is reported.

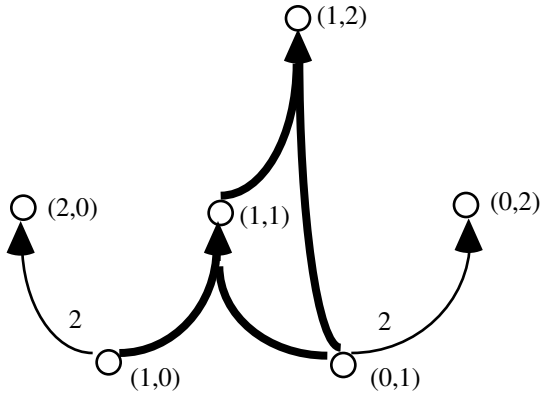


Fig.15. The acyclic hypergraph H_P

In Figure 15, the nodes corresponding to the elementary patterns (1,0) and (0,1) belong to $Level_1$, the ones corresponding to (2,0), (1,1) and (0,2) are in $Level_2$, while the node representing (1,2) belongs to $Level_3$. Multipliers equal to 1 are omitted. It appears that, for each node v, corresponding to a non-elementary pattern $A=(a_1, a_2, .., a_I)$, there is a unique directed hyperpath in H_P from the nodes in $Level_1$ to v. This directed hyperpath completely describes the composition of pattern A in terms of the elementary patterns. As an exemplification, the directed hyperpath from the nodes in $Level_1$ to node (1,2) is shadowed in Figure 15.

In order to formulate the Cutting Stock Problem relaxation in terms of minimum cost hyperflow problem on H_P, let us first introduce some other particular hyperarcs, and specify costs, flow multipliers and flow demands as follows.

Firstly, let us introduce a *headless hyperarc* $(v,\emptyset)$ for each node v, i.e. a hyperarc outgoing from v and having empty head. To simplify the notations, the other hyperarcs of H_P will also be referred to as *complete*. Note that the properties of H_P stated above still hold after this addition. In our

hyperflow model, the flow on a headless hyperarc $(v,\emptyset)$ will represent how many times the feasible pattern corresponding to v is used to satisfy the stock requirements.

Now, let us associate a flow multiplier equal to 1 with each node $v \in T_e$, for each hyperarc e in H_P, with the exception of multiple hyperarcs, having 2 as a flow multiplier (recall that they model feasible patterns made up of the same two elementary patterns). In addition, let us associate flow demands with the nodes as follows: for each node v in $Level_1$, if it corresponds to the i^{th} elementary pattern, and so to the i^{th} demand length d_i, let us associate a flow demand $b(v)= -b_i$, while for each other node v let $b(v)=0$.

Then, if integer constraints are imposed on the flow variables, each feasible hyperflow on H_P uniquely identifies a feasible way to cut the pieces of stock in order to satisfy all the demand length requirements. To be more precise, the cutting stock solution is given by the flows associated with the headless hyperarcs: the flow on a headless hyperarc $(v,\emptyset)$ gives the number of times we use the pattern corresponding to v. An example is provided in Figure 16, which extends the example of Figure 15 with the stock requirement list b=(1,4), which is the flow demand vector of $Level_1$, depicted within square brackets []. An integer feasible hyperflow is made in evidence within {} brackets: it corresponds to using once the pattern (1,2) plus the pattern (0,2), so satisfying the stock requirements.

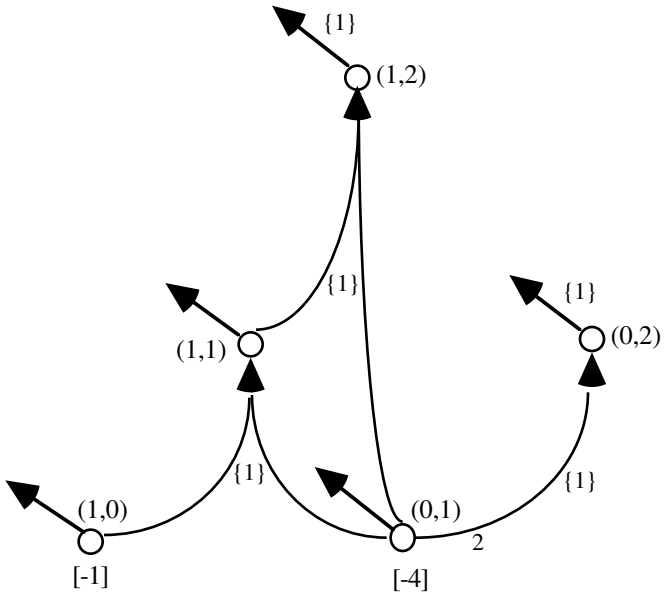


Fig. 16. A feasible cutting

Let us define now the hyperarc costs. Let v be a node in H_P, corresponding to a pattern $A=(a_1, a_2, ..,a_I)$. Let us associate with the headless hyperarc $(v,\emptyset)$ outgoing from it the cost $c((v,\emptyset))=min\{s_k: s_k \geq \sum_i a_i d_i, k=1, .., K\}- \sum_i a_i d_i$; this cost represents the minimal residual length obtained using pattern A once. The remaining hyperarcs of H_P are given zero cost. In our "sample instance", the headless hyperarcs ((1,0),Ø), ((0,1),Ø) and ((0,2),Ø) are given the cost 2,1 and 2, respectively, while the other hyperarcs have a zero cost. For this setting, the (uncapacitated) integer minimum cost hyperflow problem on H_P models the Cutting Stock Problem P. In fact, an integer hyperflow, i.e. a feasible cutting strategy, that minimizes the overall residual length of the cut pieces of stock is looked for. By relaxing the integrality constraints, the resulting (linear) minimum cost hyperflow problem on H_P models the linear programming relaxation of CSP. Due to its size, only a portion of H_P is actually maintained by the Hypergraph Simplex algorithm for CSP, which works on a reduced hypergraph by using a column generation procedure to devise new patterns at each step.

4.3. Assembly problems

In robotic assembly systems, one or more types of items are produced by assembling together several parts given in input, according to a properly defined assembly plan. An *assembly plan* is a sequence of feasible assembly operations which allow to derive the desired product from the given parts. A *feasible assembly operation* is an operation which connects together two sub-assemblies in order to obtain a new sub-assembly or the final product itself; note that the input parts can be considered as sub-assemblies of cardinality one.

Systems of this type should be characterised by a high degree of flexibility in order to allow for quick switches from one type of product to another, or to recover easily from errors or unexpected events which may require changes in the assembly plan. Crucial to this point is the capability of generating and updating assembly plans in an efficient way.

Several alternative ways to represent relations among different feasible assembly operations and several approaches to find in the space of feasible assembly operations an optimal sequence yielding the desired product have been proposed in the literature and used in practice (De Fazio&Whitney, 1987; Baldwin, Abell, Lui, De Fazio & Whitney, 1991). Recently, the use of hypergraph models has been proposed to represent and derive assembly plans (Homem De Mello&Sanderson, 1990; Gallo&Pallottino, 1992). In [Gallo&Pallottino, 1992], the problem of finding an optimal plan, under a certain set of assumption, is formulated and solved as a shortest hyperpath problem. Here

we show that this problem can be formulated as a Minimum cost hyperflow problem, and that this formulation, to a certain extent, results more powerful than the hyperpath one.

Let $Q = \{1, 2, \ldots, n\}$ be a set of distinct *parts*. An *assembly* is an item obtained by properly connecting some of the parts; the cardinality of an assembly is the number of parts it contains; a part is an assembly of cardinality 1. In the following, for the sake of the simplicity, we shall consider only binary operations, i.e. an assembly is obtained by connecting two sub-assemblies. The generalization to the case in which connections involving three or more items are allowed is straightforward. We do not make any assumption on how the parts are assembled together (screwed, soldered, …). We just assume that, for any two assemblies which can be connected in order to yield a larger assembly, an *assembly operation* is defined. For the sake of the simplicity, we assume that for any two assemblies, say π_1 and π_2, there is a unique way to connect them together, and, if π is the resulting assembly, we shall denote by $\omega=(\{\pi_1,\pi_2\}, \pi)$ the corresponding assembly operation which will be called a *feasible operation*; such an assumption does not lead to any loss of generality: in fact, in order to deal with the case in which an assembly can be obtained by connecting in several alternative ways two given sub-assemblies, it is enough to replace the assembly by a number of copies equal to the number of different possible connection modes. We shall denote by Π the set of all the assemblies which can be obtained from the parts in Q, by means of feasible operations, and by Ω the corresponding set of feasible operations. Given a set of parts, Q, a set of feasible operations, Ω, and a distinguished assembly $\pi^* \in \Pi$, an *assembly plan*, A, is a minimal[1] set of feasible operations which, starting from the parts in Q, leads to the production of the assemby π^*. By *assembly problem*, (Q, Ω, π^*, ϕ) we mean the problem of finding an assembly plan which satisfies a given criterion ϕ.

An assembly operation $\omega=(\{\pi_1,\pi_2\}, \pi)$ can be, quite naturally, represented as a hyperarc where $\{\pi_1,\pi_2\}$ is the tail set and π is the head. Thus we can represent the data of an assembly problem by means of a hypergraph $H=(V,E)$, where:

$$V = \Pi,$$

$$E = \Omega \cup \{(\emptyset, q): q \in Q\}^2.$$

---

[1] By minimal we mean that none of its proper subsets is an assembly plan.

[2] If in the operation $(\{\pi_1,\pi_2\},\pi_3)$ it is $\pi_1 = \pi_2 = \pi$, in order to represent the operation as a hyperarc we must consider the assemblies π_1 and π_2 as distinct assemblies, so duplicating the node corresponding to π.

It is straightforward to see that there is a one-to-one correspondence between the set of feasible assembly plans and the set of integer feasible flows on H, when the demand vector b has 1 in the position corresponding to node π^* and 0 elsewhere. In fact:

- under these assumptions, the flow on hyperarc $(\{\pi_1,\pi_2\}, \pi)$, for each $(\{\pi_1,\pi_2\}, \pi)\in \Omega$ corresponds to the activity of producing the item π, and its value gives the number of times this operation is performed and hence the number of copies of π produced in the process of producing the item π^*;

- the flow conservation constraints assure that all the items (assemblies) produced in the process are used in subsequent operations, but the last one, π^*, and that no operation can be performed unless all the items it needs as input have been produced;

- finally, the flow on arc $(\emptyset, q)$, $\forall\ q\in Q$, gives the number of copies of part q needed in the production of π^*.

If all the operations (hyperarcs) have been assigned a cost, the *production cost* of an assembly plan is defined as the sum of the costs of all the operations performed; clearly if an operation is performed p (>1) times, then its contribution to the total cost is p times its cost. If the costs are set equal to the operations' durations, then the production cost gives the total time needed to produce one unity of π^* in the case in which no parallelism is allowed, i.e. all the operations are performed in series. In this assumption a minimum cost (or duration) assembly plan can be found simply by solving an uncapacitated Minimum cost hyperflow problem. Being the problem uncapacitated, it is equivalent to solving a Shortest hyperpath problem, which can be done in linear time (Gallo&Pallottino, 1992).

More interesting is the case in which we want to find the minimum duration assembly plan with a parallelism of a given degree k, i.e. when k parallel identical machines are available for the execution of the operations.

Assume first all the operations to have equal duration, say one unit of time. This assumption will be relaxed later. If T is an upper bound on the total time needed to perform all the operations in the optimal assembly plan, we can expand the hypergraph H over time, obtaining the new *time-expanded hypergraph* $\tilde{H} = (\tilde{V}, \tilde{E})$, where:

$$\tilde{V}=\{(\pi, t): \pi\in \Pi, t= 0, 1, \ldots, T\}\cup\{o_0, o_1, \ldots,o_T\}\cup\{d\}$$

$$\tilde{E}=\{(\{(\pi_1, t), (\pi_2, t), o_t\}, (\pi, t+1)): \forall(\{\pi_1,\pi_2\}, \pi)\in \Omega, t=0, 1, \ldots, T-1\}$$

$$\cup\{((\pi,t),(\pi,t+1)):\pi\in \Pi,t=0,1,\ldots,T-1\}$$

$$\cup\{(\emptyset, (q, t)): q\in Q, t= 0, 1, \ldots, T\}\cup\{(\emptyset, o_t): t=0, 1, \ldots, T-1\}$$

$\cup\{((\pi^*, t), d): t=1, 2, \ldots, T\}$.

All the hyperarcs have unbounded capacity, except the arcs of the type $(\emptyset, o_t)$ which have capacity equal to k, i.e. to the number of available machines.

Extending the interpretation of flows given previously, here, a flow of value f on hyperarc $(\{(\pi_1, t), (\pi_2, t), o_t\}, (\pi, t+1))$ means that operation $\omega=(\{\pi_1,\pi_2\}, \pi)$ is performed f times, starting at time t and ending at time $t+1$. Being k the capacity of arc $(\emptyset, o_t)$, for all t, no more than k parallel operations can be performed at the same time. Thus, with respect to the original model, here a feasible flow corrisponds not only to a feasible assembly plan, but also to a feasible schedule of the assemby plan on the given k machines.

Different problems can be modelled by a proper choice of the hyperarcs' costs. If each hyperarc of the type $(\{(\pi_1, t), (\pi_2, t), o_t\}, (\pi, t+1))$ is assigned the cost of operation $\omega=(\{\pi_1,\pi_2\}, \pi)$, and all the other hyperarcs have zero cost, then the minimum cost hyperflow gives the minimun cost assembly plan which can be performed with k parallel machines within T units of time. On the other side, if hyperarc $((\pi^*, t), d)$ is assigned cost t, for all t, while all other hyperarcs have zero cost, then the minimum cost hyperflow gives the minimum makespan assembly plan when only k parallel machines are available.

The case of different durations can be reduced to the previous one simply by replacing in H any hyperarc $(\{\pi_1,\pi_2\}, \pi)$, whose duration is the integer $p>1$ with the p hyperarcs $(\{\pi_1,\pi_2\}, \pi^1)$, $(\{\pi^1\}, \pi^2)$, $(\{\pi^2\}, \pi^3)$, $\ldots,(\{\pi^{p-1}\}, \pi^p)$, each of unit duration. Then the time-expanded hypergraph $\tilde{H}$ is constructed as seen before. Clearly, in this case the solution obtained by solving a minimum cost hyperflow problem is a solution in which preemption is permitted, that is it is possible that the execution of an operation be interrupted before completion and resumed later. In the case in which preemption is not allowed, the solution obtained provides a lower bound on the optimum and can be used as the starting solution of a heuristic algorithm.

In order to illustrate the time-expanded hypergraph and its properties, consider De Fazio and Whitney's transmission model as reported in Figure 17. It is the model of a transmission with 11 parts (by ignoring bolts), which is symmetric around an axis of revolution (see [Homem de Mello and Lee, 1991]). Figure 18 shows the corresponding assembly's *liaison diagram*.

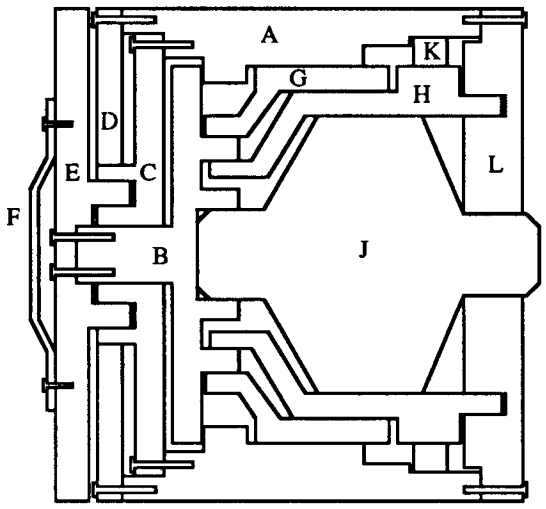


Fig. 17: De Fazio and Whitney's transmission model

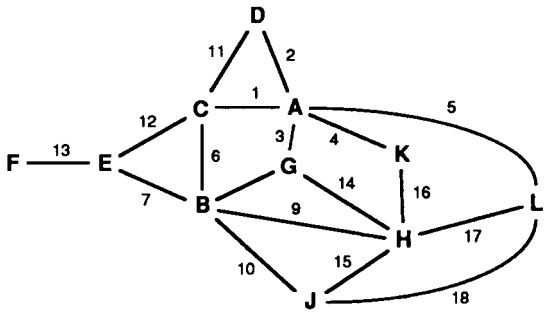


Fig. 18: The assembly's liaison diagram

An assembly plan for the transmission model together with the corresponding hyperflow on a time-expanded hypergraph with a parallelism of degree 4 is illustrated in Figure 19. In this figure, each assembly is represented as the sequence of the parts it contains, and notations like $(HKL)^2$ mean that the sub-assembly under consideration (HKL in our example) is contained twice.

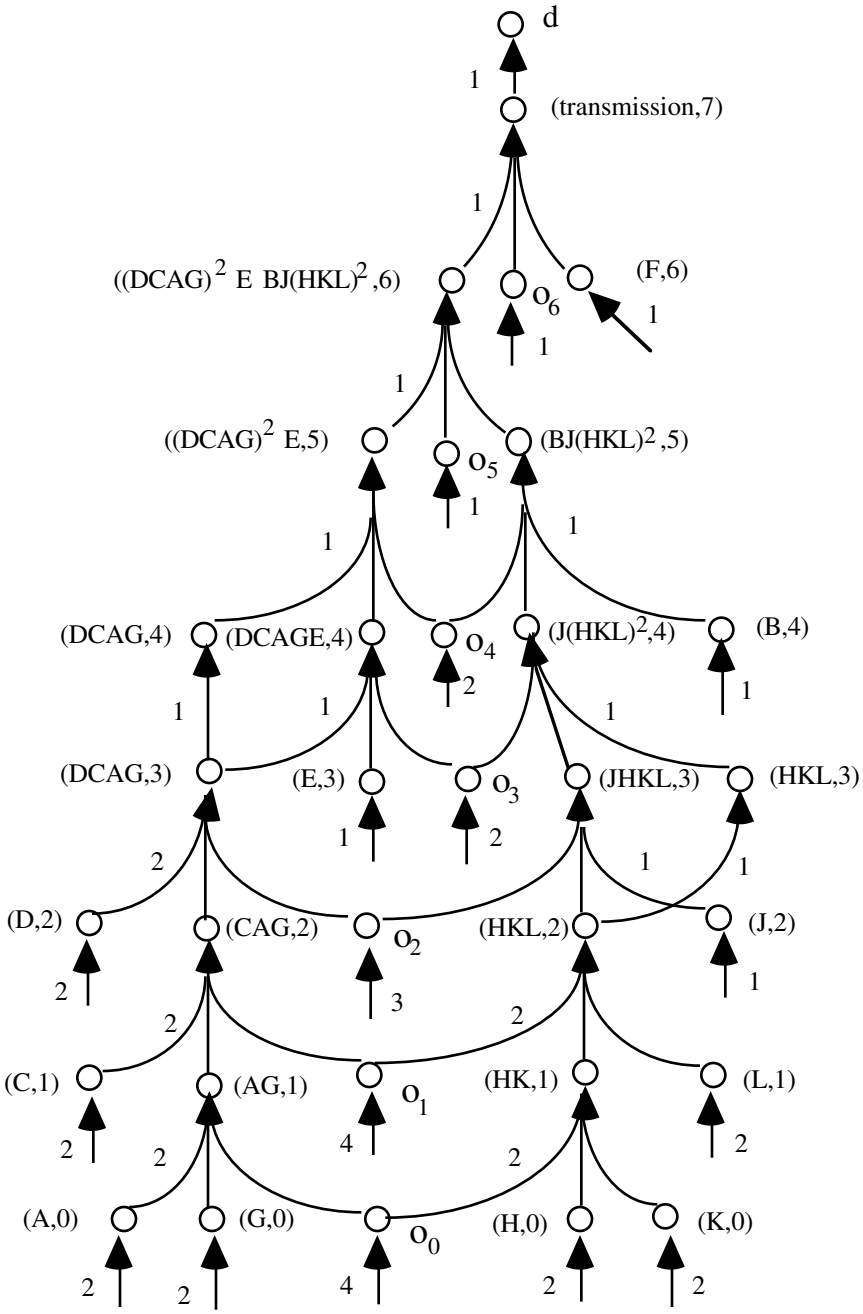


Fig. 19: An assembly plan on a time-expanded hypergraph with k=4

4.4. Economic applications

In this section, linear production models will be described in terms of hyperflows. A *linear production model* involves *goods* consumed (*inputs*) or produced (*outputs*) by *activities*, and describes a certain economic system (Gale, 1960; Henderson&Quandt, 1971; Takayama, 1985). The model is said to be *simple* if each activity produces exactly one output, while it is called *general* if at least one activity produces more than one good. Generally, the involved goods are classified as follows:

primary goods: they are goods like plant equipment and labor (sometimes, they are also referred to as *factors of production*), generally characterized by the following two properties: they cannot be outputs of activities of the model, and they are available in a limited amount (Gale, 1960); however, they can be considered as outputs of fictitious activities, which produce exactly one primary good and consume nothing;

target goods: they are the goods required to the system by consumers;

collateral goods (*externalities*): they are goods like discards, wastes and pollution, which are generally produced by activities but are unrequired and undesiderable.

According to the above classification, the linear production models generally involve q primary goods, say $P_1,\ldots,P_q$, n target goods, $G_1,\ldots,G_n$, h collateral goods, $V_1,\ldots,V_h$, m activities, $W_1,\ldots,W_m$ and r ($r \geq q$) fictitious activities, $F_1,\ldots,F_r$, offering primary goods. By assuming that each activity is relative to a given finite interval of time T, and that all the activities are able to operate at different production levels, each linear production model is then completely described by the following non-negative matrices:

the *input matrices* $A=[\alpha_{ij}]$, $B=[\beta_{ij}]$ and $C=[\chi_{ij}]$, where α_{ij}, β_{ij} and χ_{ij} are, respectively, the amount of inputs P_i, G_i and V_i needed by the activity W_j in order to work at its unitary production level;

the *output matrices* $D=[\delta_{ij}]$ and $E=[\varepsilon_{ij}]$, where δ_{ij} and ε_{ij} are, respectively, the amounts of outputs G_i and V_i produced by W_j at its unitary production level;

the *auxiliary matrix* $R=[\rho_{ik}]$, that denotes which primary goods are offered by the fictitious activities; namely, $\rho_{ik}=1$ if F_k offers P_i, and $\rho_{ik}=0$ otherwise.

Note that both β_{ij} and δ_{ij} (or χ_{ij} and ε_{ij}) may be positive. For example, it may be necessary to consume a certain amount of steel in order to produce more steel; also χ_{ij} may be positive, i.e. a certain discard V_i can be consumed by an activity W_j, if W_j is a recycling activity.

Now, let us consider a *simple linear production model*, i.e. a linear production model where each activity produces exactly one good, and assume that each activity W_j works at its unitary level of

production when it produces one net unit of output, i.e. if $\delta_{ij}>0$ then $\delta_{ij} - \beta_{ij}=1$. Observe that, in this kind of model, we can ignore the collateral goods (i.e.h=0); in fact, it is a non sense the presence of activities producing only wastes or discards.

Let $x_{W_j}\in[0,u_{W_j}]$ be the net amount of output produced by W_j, j=1,…,m, $x_{F_k}\in[0,u_{F_k}]$ be the amount of primary good given by F_k, k=1,…,r, and suppose that b_{P_i} and b_{G_j} represent the amount of P_i and G_j required by the consumers, i=1,…,q and j=1,…,n. Then, if c_{F_k} and c_{W_j} are the costs for the production of one unit of output of F_k and W_j, respectively, the problem of satisfying the consumer requirements by minimizing the total production cost can be formalized as:

$$\begin{cases} \min \ cx \\ Mx=b \\ 0 \le x \le u \end{cases}$$

where $M=\begin{bmatrix} R & -A \\ 0 & D-B \end{bmatrix}$, $b=\begin{bmatrix} b_P \\ b_G \end{bmatrix}$, $c=\begin{bmatrix} c_F \\ c_W \end{bmatrix}$, $u=\begin{bmatrix} u_F \\ u_W \end{bmatrix}$ and $x=\begin{bmatrix} x_F \\ x_W \end{bmatrix}$.

By the previous assumption, this is a Minimum cost hyperflow problem. Consider for example a computer factory. In order to produce the final good, motherboards, cases and drives are needed. Assuming that drives and labor are primary goods, the productive processes can be described as shown in Figure 20, where “input_case” and “input_motherboard” indicate the set of further primary goods needed to produce cases and motherboards, respectively.

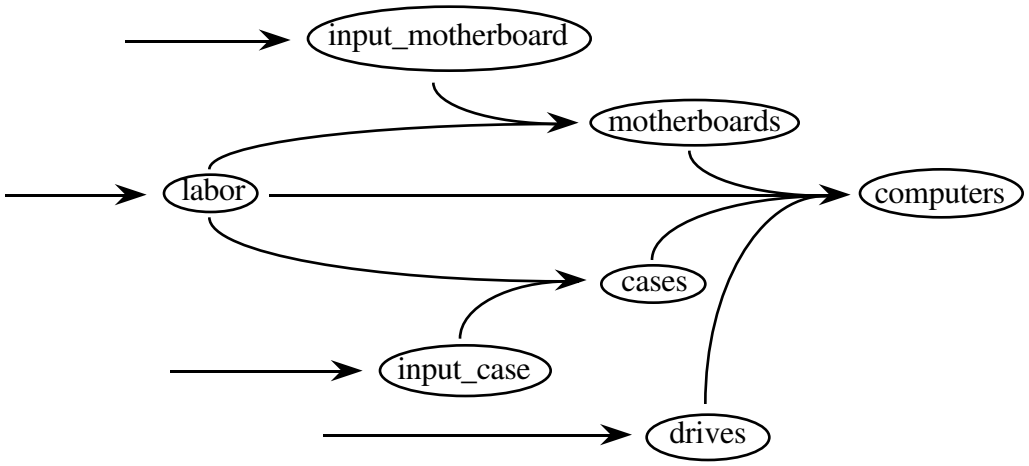


Fig. 20

As the example reveals, each activity can be represented by a hyperarc; in particular, each fictitious activity corresponds to a tailless hyperarc. The entire economic system can thus be

described by a hypergraph, and the problem can be modelled as a minimum cost hyperflow problem.

The most unnatural assumption of the previous model is that each activity produces exactly one good; so let us generalize it by allowing that each activity produces more than one good. If $x_{w_j} \in [0,u_{w_j}]$ represents the production level of W_j, $j=1,\ldots,m$, $x_{F_k} \in [0,u_{F_k}]$ is the amount of primary good given by F_k, $k=1,\ldots,r$, the non-negative vectors b_P and b_G are the consumer demands of primary and target goods, respectively (clearly, the requirement of collateral goods is zero), and c_F and c_w are the cost vectors of the activities per unit of production, then the *general linear production* problem can be formalized as:

$$\begin{cases} \min cx \\ Mx=b \\ 0 \le x \le u \end{cases}$$

where $M=\begin{bmatrix} R & -A \\ 0 & D-B \\ 0 & E-C \end{bmatrix}$, $b=\begin{bmatrix} b_P \\ b_G \\ 0 \end{bmatrix}$, $c=\begin{bmatrix} c_F \\ c_w \end{bmatrix}$, $u=\begin{bmatrix} u_F \\ u_w \end{bmatrix}$ and $x=\begin{bmatrix} x_F \\ x_w \end{bmatrix}$.

Now, the columns of M relative to the activities W_j can have more than one positive component. So, in order to interpret the problem as a minimum cost hyperflow problem, we have to reformulate the problem in such a way to have exactly one positive element for each column. This is obtained by the variable transformation $x'_w=u_w - x_w$ This transformation is illustrated in Figure 21 for the simple case in which only one arc, i.e. W_j, is transformed (the demands are indicated in parenthesis).

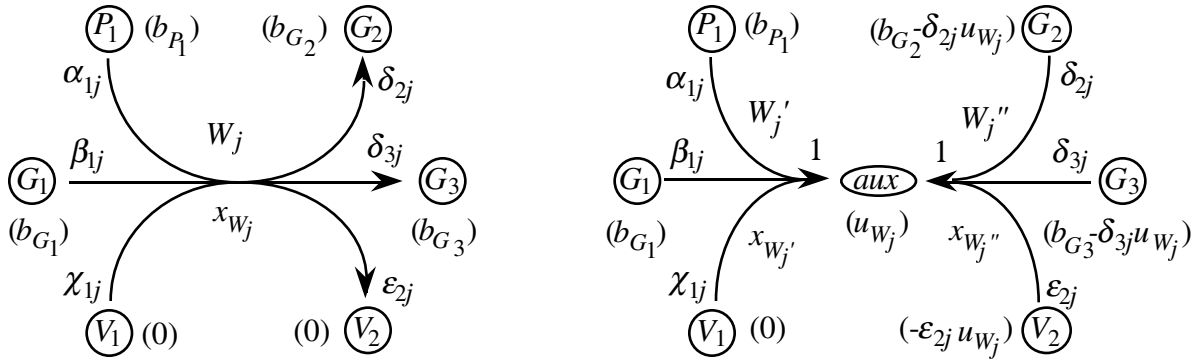


Fig. 21

In the original hyperarc, on the left, an activity level of value $x_{w_j} \in [0,u_{w_j}]$ induces a demand at the tail nodes of $-x_{w_j}\alpha_{1j}$, $-x_{w_j}\beta_{1j}$ and $-x_{w_j}\chi_{1j}$, respectively. These demands can be satisfied by

means of either input flows or flows arriving from other portions of the network. At the head, it produces a flow of value $x_{w_j}\delta_{2j}$, $x_{w_j}\delta_{3j}$ and $x_{w_j}\varepsilon_{2j}$. Considering the demands, the flow which is made available to the rest of the network is $x_{w_j}\delta_{2j}$-b_{G_2}, $x_{w_j}\delta_{3j}$-b_{G3} and $x_{w_j}\varepsilon_{2j}$, respectively.

In the transformed network, on the rigth, the same value x_{w_j} on arc W'_j gives rise to the same demands -$x_{w_j}\alpha_{1j}$, -$x_{w_j}\beta_{1j}$ and -$x_{w_j}\chi_{1j}$ at nodes P_1, G_1 and V_1, and produces a flow of value x_{w_j} at the dummy node *aux*. Being u_{w_j} the demand at *aux*, we need now a level of activity u_{w_j}-x_{w_j} at arc W''_j, which induces demands -$(u_{w_j}$-$x_{w_j})\delta_{2j}$, -$(u_{w_j}$-$x_{w_j})\delta_{3j}$ and -$(u_{w_j}$-$x_{w_j})\varepsilon_{2j}$ at nodes G_2, G_3 and V_2, respectively. Then what is available to the rest of the network at G_2, G_3 and V_2 is -$(u_{w_j}$-$x_{w_j})\delta_{2j}$-$(b_{G_2}$-$\delta_{2j}u_{w_j})$=$x_{w_j}\delta_{2j}$-b_{G_2}, -$(u_{w_j}$-$x_{w_j})\delta_{3j}$-$(b_{G_3}$-$\delta_{3j}u_{w_j})$=$x_{w_j}\delta_{3j}$-b_{G_3} and -$(u_{w_j}$-$x_{w_j})\varepsilon_{2j}$-$(-\varepsilon_{2j}u_{w_j})$=$x_{w_j}\varepsilon_{2j}$, respectively, as it was in the original network.

The general linear production model considered so far can be further generalized. Firstly, it is reasonable that the consumer demands are not fixed, but they vary within fixed intervals. In addition, *underproductions* and/or *overproductions* may be allowed, if this is economically convenient. For example, a computer factory can fix its production target in [100, 110] units at day, but it can decide to pay for an overproduction, if this can derive an economical advantage. Then, assume that the consumer requirements of the generic primary good P_i and of the generic target good G_j can vary in the interval [b_{P_i} ,b_{P_i}+Δb_{P_i}] and in [b_{G_j} ,b_{G_j}+Δb_{G_j}], respectively, where $\Delta b_{P_i}\geq 0$ and $\Delta b_{G_j}\geq 0$. Assume also that up_{P_i} and up_{G_j} denote the maximum allowable underproductions of goods P_i and G_j, respectively, and that op_{P_i} , op_{G_j} and op_{V_k} denote the maximum overproductions of goods P_i, G_j and V_k, with non-negative linear costs uc_{P_i}, uc_{G_j}, oc_{P_i}, oc_{G_j} and oc_{V_k}, respectively. By introducing the vectors $y_P\in[0,\Delta b_P]$, $y_G\in[0,\Delta b_G]$, $uz_P\in[0,up_P]$, $uz_G\in[0,up_G]$, $oz_P\in[0,op_P]$, $oz_G\in[0,op_G]$ and $oz_V\in[0,op_V]$, the following minimum cost hyperflow model can then be obtained:

$$\begin{cases} \min \ \overline{c}\,\overline{x} \\ \overline{M}\,\overline{x}=\overline{b} \\ 0 \leq \overline{x} \leq \overline{u} \end{cases}$$

where $\overline{M}=\begin{bmatrix} R & 0 & -A & -I_q & 0 & I_q & 0 & -I_q & 0 & 0 \\ 0 & -D & -B & 0 & -I_n & 0 & I_n & 0 & -I_n & 0 \\ 0 & -E & -C & 0 & 0 & 0 & 0 & 0 & 0 & -I_h \\ 0 & I_m & I_m & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, I_i is the ixi identity matrix,

$i=m, n, q, h$, $\overline{b}=\begin{bmatrix} b_P \\ b_G-Du_w \\ -Eu_w \\ u_w \end{bmatrix}$, $\overline{u}=[u_F,u_w,u_w,\Delta b_P,\Delta b_G,up_P,up_G,op_P,op_G,op_V]^{\mathrm{T}}$,

$\overline{c}=[c_F,0,c_w,0,0,uc_P,uc_G,oc_P,oc_G,oc_V]^{\mathrm{T}}$ and $\overline{x}=[x_F,x'_w,x_w,y_P,y_G,uz_P,uz_G,oz_P,oz_G,oz_V]^{\mathrm{T}}$.

Finally, dynamic components can be modelled as well. Assume for instance that overproduced goods can be stored and used later. For example, consider the simple linear production model previously described (the extension to the general production model is trivial), and assume that the time interval T is one year. In order to study the economic system in a period of two years, let $M_i=\begin{bmatrix} R_i & -A_i \\ 0 & D_i-B_i \end{bmatrix}$, $b_i=\begin{bmatrix} b_{i_P} \\ b_{i_G} \end{bmatrix}$, $c_i=\begin{bmatrix} c_{i_F} \\ c_{i_W} \end{bmatrix}$, $u_i=\begin{bmatrix} u_{i_F} \\ u_{i_W} \end{bmatrix}$ and $x_i=\begin{bmatrix} x_{i_F} \\ x_{i_W} \end{bmatrix}$ be the data relative to the i^{th} year of production, i=1, 2. In addition, let c_P and c_G be the linear non-negative cost vectors for storing the overproduction of primary and target goods, respectively. Denoting by $s_P \in [0,max_P]$ and $s_G \in [0,max_G]$ the stored amounts of primary and target goods, respectively, the following minimum cost hyperflow problem is then obtained:

$$\begin{cases} \min \ \overline{c}\,\overline{x} \\ \overline{M}\,\overline{x}=\overline{b} \\ 0 \le \overline{x} \le \overline{u} \end{cases}$$

where $\overline{M}=\begin{bmatrix} R_1 & -A_1 & -I_q & 0 & 0 & 0 \\ 0 & D_1-B_1 & 0 & -I_n & 0 & 0 \\ 0 & 0 & I_q & 0 & R_2 & -A_2 \\ 0 & 0 & 0 & I_n & 0 & D_2-B_2 \end{bmatrix}$, $\overline{c}=\begin{bmatrix} c_1 \\ c_P \\ c_G \\ c_2 \end{bmatrix}$, $\overline{u}=\begin{bmatrix} u_1 \\ max_P \\ max_G \\ u_2 \end{bmatrix}$, $\overline{x}=\begin{bmatrix} x_1 \\ s_P \\ s_G \\ x_2 \end{bmatrix}$ and $\overline{b}=\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$.

References

Bala M., and K. Martin, “A Mathematical Programming Approach to Data Base Normalization”, Tech. Report, Graduate School of Business, University of Chicago (1993).

Baldwin, D.F., T.E. Abell, M.-C. Lui, T.L. De Fazio and D.E. Whitney: "An integrated computer aid for generating and evaluating assembly sequences for mechanical products", *IEEE Trans. on Robotics and Automation* **7** (1991) 78-94.

Cambini R., G. Gallo and M.G. Scutellà, “Minimum Cost Flows on Hypergraphs”, TR- 1/92, Dipartimento di Informatica, Università di Pisa (1992).

Cambini R., G. Gallo and M.G. Scutellà, “Flows on Hypergraphs”, TR-4-96, Dipartimento di Informatica, Università di Pisa (1996), to appear in *Mathematical Programming B*.

Capitani G., O. Catoni, A. Frangioni, G. Gallo and G. Saviozzi, “A Bundle type Dual-Ascent approach to Linear Multicommodity MinCost Flow Problems”, presented at HPSNO International Conference, Capri (1995).

De Fazio, T.L. and D.E. Whitney, "Simplified generation of all mechanical assembly sequences", *I.E.E.E. J. oof Robotics and Automation* **RA-3** (1987) 640-658.

Gale D., *The Theory of Linear Economic Models*, McGraw-Hill (1960).

Gallo G. and S. Pallottino, “Hypergraph models and Algorithms for the Assembly Problem”, TR-6/92, Dipartimento di Informatica, Università di Pisa (1992).

Gallo G., N. Licheri & M.G. Scutellà, "The Hypergraph Simplex approach: some experimental results", TR-12/95, Dipartimento di Informatica, Università di Pisa (1995), submitted to *Ricerca Operativa*.

Gambale M., M. Nonato and M.G. Scutellà, “The Cutting Stock Problem: a new model based on hypergraph flows”, *Ricerca Operativa* **74**, anno **25** (Estate 1995).

Henderson J.M., and R.E. Quandt, *Microeconomic Theory*, *a Mathematical approach*, McGraw-Hill (1971).

Homem De Mello, L.S. and A.C. Sanderson: “And/Or Graph Representation of Assembly Plans”, *IEEE Trans. on Robotics and Automation* **6** (1990) 188-199.

Homem De Mello, L.S. and S. Lee: *Computer-Aided Mechanical Assembly Planning*, Kluwer Academic Publishers (1991).

Jeroslow R.G., R.K. Martin, R.R. Rardin and J. Wang, “Gainfree Leontief substitution flow problems”, *Mathematical Programming* 57 (1992) 375-414.

Licheri F., *Implementazione e Sperimentazione di un prototipo di Simplesso su Ipergrafi*, Tesi di Laurea, Dip. di Informatica, Università di Pisa (1994).

CPLEX Opt., *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, Incline Village, Nevada (1992).

L. Schrage, *LINDO*, *An Optimization Modelling System*, The Scientific Press (1991).

A. Takayama, *Mathematical Economics*, Cambridge University Press (1985).