Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10, D-10711 Berlin-Wilmersdorf, Germany

Atef Abdel-Aziz Abdel-Hamid
Ralf Borndörfer

On the Complexity of Storage Assignment Problems

Preprint SC–94–14 (June 1994)

On the Complexity of Storage Assignment Problems

Atef Abdel-Aziz Abdel-Hamid* Ralf Borndörfer†

June 20, 1994

Abstract. The *storage assignment problem* asks for the cost minimal assignment of containers with different sizes to storage locations with different capacities. Such problems arise, for instance, in the optimal control of automatic storage devices in flexible manufacturing systems. This problem is known to be $\mathcal{NP}$-hard in the strong sense. We show that the storage assignment problem is $\mathcal{NP}$-hard for *bounded sizes and capacities*, even if the sizes have values 1 and 2 and the capacities value 2 only, a case we encountered in practice. Moreover, we prove that no polynomial time ϵ-approximation algorithm exists. This means that almost all storage assignment problems arising in practice are indeed hard.

Keywords. flexible manufacturing, complexity theory

Mathematics Subject Classification (1991). 68Q25, 90B06, 90C60

1 Introduction

Consider two disjoint sets S and C to be the left and right node sets of a complete bipartite graph $G = (V, E) := (S \cup C, S \times C)$, node weights $s : S \to \mathbb{Z}^+$ and $c : C \to \mathbb{Z}^+$ and edge weights $w : E \to \mathbb{Z}^+$. Denote by $\delta(v)$, $v \in V$, the set of edges incident to v, by $\deg(v) := |\delta(v)|$ its valency. A *feasible assignment* in (G, c, s) is a set $A \subseteq E$ of edges, such that i) $|\ \delta(i) \cap A\ | = 1$ for each $i \in S$ and ii) $\sum_{ij \in \delta(j) \cap A} s_i \leq c_j$ for each $j \in C$. The *storage assignment problem* (SAP) is to find a feasible assignment A^* with minimal weight $\sum_{ij \in A^*} w_{ij}$.
This name is motivated by the following interpretation. S is a set of containers (s_i = size of container i), C a set of capacitated storage locations (c_j = capacity of location j). Storing container i in location j has a cost of w_{ij}. The task is to store all containers in the locations cost minimally obeying the capacity restrictions (sum of the sizes of the containers assigned to a storage location must not exceed its capacity). Introducing binary variables x_{ij}, $ij \in E$, that are 1 if container i is stored in location j and 0 otherwise, the SAP can be modelled as the following integer programming problem, see [Abd94], [AG94].

$$
\text{(SAP)} \qquad
\begin{array}{rlll}
\min \displaystyle\sum_{i \in S} \sum_{j \in C} w_{ij} x_{ij} & & & \\
\displaystyle\sum_{j \in C} & x_{ij} & = 1 & \forall i \in S \\
\displaystyle\sum_{i \in S} & s_i x_{ij} & \leq c_j & \forall j \in C \\
& x_{ij} & \in \{0, 1\} & \forall ij \in E
\end{array}
$$

The SAP came up in a joint project with the production plant "Werk für Arbeitsplatzsysteme" of Siemens Nixdorf Informationssysteme (SNI) in Augsburg, Germany, that manufactures PCs and PC-related products. A bottleneck in the production process was the operation speed of the automatic storage systems of the factory. One of the aspects that influence this speed is the strategy of assigning boxes to free storage

*Konrad Zuse Zentrum für Informationstechnik Berlin, Heilbronner Straße 10, 10711 Berlin-Wilmersdorf, Germany, On leave from Department of Engineering Mathematics & Physics, Cairo University, Egypt

†Konrad Zuse Zentrum für Informationstechnik Berlin, Heilbronner Straße 10, 10711 Berlin-Wilmersdorf, Germany and Technical University of Berlin, Fachbereich 3 Mathematik, Sekretariat MA 6–1, Straße des 17. Juni 136, 10623 Berlin, Germany

locations or how to solve the SAP, see [Abd94], [AG94]. In our application, containers are of sizes 1 and 2 and location capacities vary from 1 to 3, i.e. $s : S \to \{1, 2\}$ and $c : C \to \{1, 2, 3\}$. We call this special SAP *box assignment problem* (BAP).
It is known that SAPs are $\mathcal{NP}$-hard in the strong sense, see e.g. [MT81], [MT91]. But the transformations used in the $\mathcal{NP}$-hardness proofs do not work for bounded sizes and capacities, i.e., the case of the BAP. But this is the practical relevant case. We determine in this article the complexity of the BAP. The following results will be shown in the sequel.

- BAPs (and thus SAPs) are $\mathcal{NP}$-hard, even if all sizes are 1 and 2 and capacities are 2 only.
- No polynomial time ϵ-approximate algorithm for BAPs (and thus for SAPs) exists.
- The problem of deciding whether a SAP has a feasible solution is polynomially solvable for all bounded sizes and capacities.

Thus, basically all storage assignment problems arising in practice, in particular the one we encountered at SNI, are indeed hard. Finding a solution of provable quality guarantee is also hard. These results motivate a *polyhedral* study of the problem to design a cutting plane algorithm. Computational experiments have shown that real-world SAPs, for example the ones at SNI, can be solved to optimality with this approach efficiently, see [Abd94], [AG94].

2 Complexity

The following decision problem BAP_D can be associated with the BAP.

(BAP_D)	Given two disjoint sets S and C, that form the left and right node sets of a bipartite graph $G = (S \cup C, E)$, "sizes" $s : S \to \{1, 2\}$ and "capacities" $c : C \to \{1, 2, 3\}$. Does there exists a feasible assignment in (G, c, s)?

Note that in BAP the graph G is complete. The reason is that —in principle— it should be possible to store any box i in any location j if $s_i \leq c_j$. (To simplify notation, we also include the edges ij with $s_i > c_j$.) In BAP_D this is not the case. It may happen that an edge $ij \notin E$ although $s_i \leq c_j$.
The first step in establishing that BAP is $\mathcal{NP}$-hard, even if all sizes and capacities have values 1 and 2 will be to show that BAP_D is $\mathcal{NP}$-complete. This will be done by reducing the $\mathcal{NP}$-complete *Independent Set Problem* (ISP_D) [GJ79] to BAP_D. A node set of a graph is called *independent* or *stable*, if no two nodes in it are adjacent.

(ISP_D)	Given a graph $G = (V, E)$ and a positive integer k. Does G contain an *independent* node set of size k?

Theorem 1 *BAP_D is $\mathcal{NP}$-complete, even if all sizes and capacities have values* 1 *and* 2*.*

Proof. BAP_D is in $\mathcal{NP}$, because a nondeterministic algorithm need only guess a feasible assignment and check in polynomial time that it is indeed one.
We now transform ISP_D to BAP_D. Let an arbitrary instance of ISP_D be given by the graph $G = (V, E)$ and the positive integer k. We must construct an instance (G', c', s') of BAP_D, i.e., a bipartite graph $G' = (S' \cup C', E')$ with left nodes S' ("box nodes") and right nodes C' ("location nodes"), sizes $s' : S' \to \{1, 2\}$ and capacities $c' : C' \to \{1, 2\}$, such that (G', c', s') has a feasible assignment if and only if G contains an independent set of size k.
The graph G' associated with the instance of BAP_D can be viewed as consisting of components.
First, there will be k "selector nodes" $a_1, \ldots, a_k$ which will be used to select k nodes from V. The selector nodes are box nodes of size 2 ($a_i \in S', s'_{a_i} = 2, i = 1, \ldots, k$).
Second, for each original node v in G, G' contains a "cascade component" γ_v. The main part of each γ_v is a rooted tree with at least $\deg(v)$ leaves. The tree can be viewed as being iteratively composed from a "root component" and one or more "leaf components". The structure of both components is shown in Figure 1.

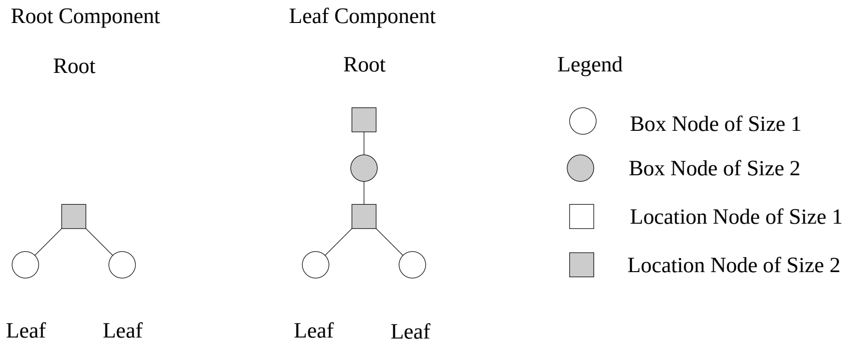


Figure 1: Components of a Cascade Tree

Initially, the tree consists of a root component only. This root component forms the top of the tree in any cascade, its root is also the root of the tree. If $\deg(v) \leq 2$ the root component is already the whole tree. Otherwise, a copy of a leaf component is attached to each leaf of the tree in such a way that an edge joins its root to the corresponding leaf. In this way the number of leaves doubles. This step is repeated until the number of leaves exceeds $\deg(v)$. This happens after $\lceil \log_2 \deg(v) \rceil$ steps when the tree has $2^{\lceil \log_2 \deg(v) \rceil} \leq 2\deg(v)$ leaves. We now pick $\deg(v)$ leaves and label them with the edges incident to v. The cascade is completed by adding a "garbage collection node" for each of the $2^{\lceil \log_2 \deg(v) \rceil} - \deg(v)$ remaining unlabelled leaves that is joined by an edge to its leaf. Garbage collection nodes are location nodes of capacity 1. An example of a cascade component for a node of degree six incident to six edges a, b, c, d, e and f is shown in Figure 2.

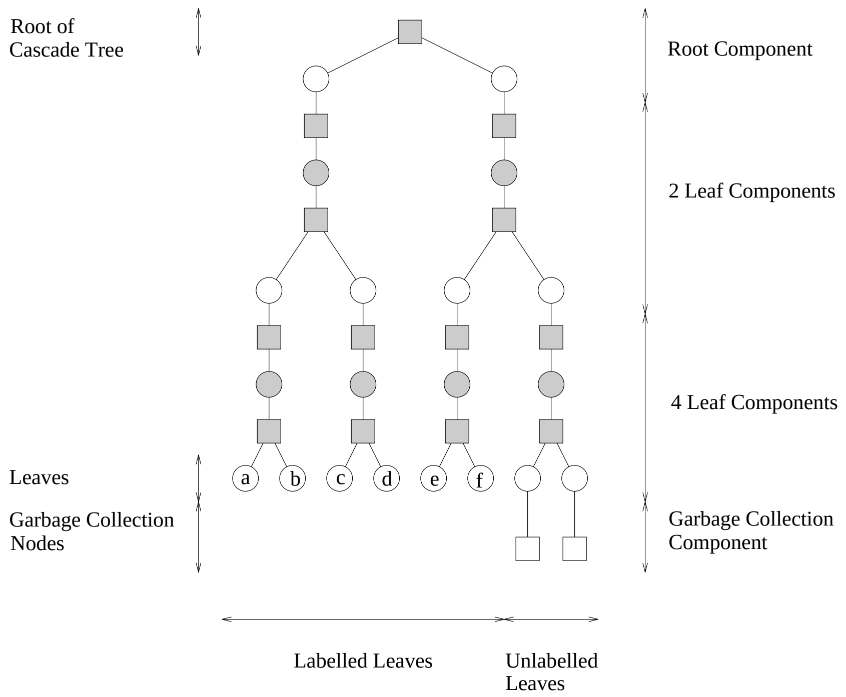
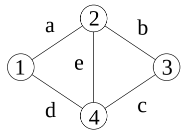


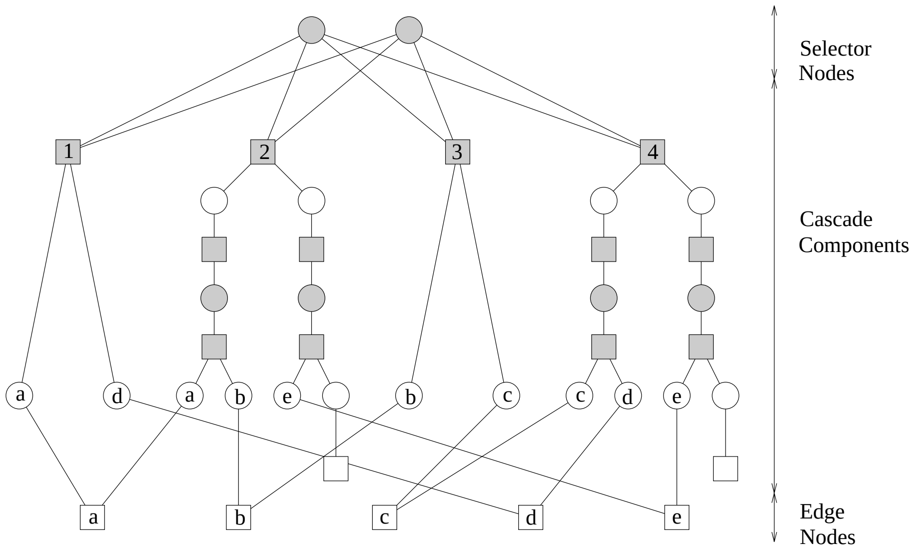
Figure 2: Cascade Component for a Node with Degree six

The last component of G' is an "adjacency testing component". Together with the cascade components it will be used to test whether two selected nodes are adjacent. It consists of an "edge node" for each edge in G which is a location node of size 1.
The construction is completed by connecting the components by additional edges.

First, each selector node is connected to all root nodes of the cascade components.
Second, each edge node corresponding to an edge $ij \in E$ is connected to the leaf nodes labelled ij of the two cascades γ_i and γ_j.
This completes the construction of the instance (G', c', s') of BAP_D. Figure 3 shows a graph G and the associated instance (G', c', s') of BAP_D for $k = 2$.



(G, k = 2)



(G', c', s')

Figure 3: A Graph and the Associated Instance of BAP_D for $k = 2$

It is not hard to see that this transformation can be done in polynomial time.
We show now that (G', c', s') contains a feasible assignment if and only if G has an independent set of size k.
Suppose $V^* = \{v_1, \ldots, v_k\}$ is an independent set in G. A feasible assignment A' in G' can be constructed as follows. Adopting standard terminology, the father of a node v in a rooted tree is the next node on the unique path from v to the root, the sons are all nodes adjacent to v but the father. Consider the cascades $\gamma_v, v \in V \setminus V^*$. It is easy to see that all box nodes in γ_v can be assigned to their fathers inside γ_v. For the remaining cascades $\gamma_{v_i}, i = 1, \ldots, k$ assign selector node a_i to the root node of cascade γ_{v_i}. This forces all box nodes in γ_{v_i} to be assigned to their sons. Then, the unlabelled leaves have to be assigned to their corresponding garbage collection nodes and the labelled leaves have to be assigned to their corresponding edge nodes. This completes the construction of the assignment A'. A' is feasible, because at most one labelled leaf node (of size 1) is assigned to each edge node (of capacity 1). Suppose the contrary, i.e., there is an edge node v_iv_j and two labelled leaves of the cascades γ_{v_i} and γ_{v_j} that are both assigned to v_iv_j. The reason that the leaf nodes are assigned to v_iv_j is that two selector nodes are assigned to the root nodes of the cascades γ_{v_i} and γ_{v_j}. But then v_i and v_j belong to the stable set V^* and the edge v_iv_j cannot exist in G, a contradiction.
Conversely, suppose G' has a feasible assignment A'. Then the selector nodes $a_1, \ldots, a_k$ are assigned to the root nodes of cascades $\gamma_{v_1}, \ldots, \gamma_{v_k}$. Then the labelled leaves of cascades $\gamma_{v_1}, \ldots, \gamma_{v_k}$ have to be assigned to their corresponding edge nodes. We will show that the selected nodes $v_1, \ldots, v_k$ form a stable

set in G. Suppose v_i and v_j are adjacent. Each cascade γ_{v_i} and γ_{v_j} contains a labelled leaf $v_i v_j$ of size 1 and both of them are assigned to the edge node $v_i v_j$ of capacity 1 — a contradiction.
Since we used sizes and capacities having values 1 and 2 only, this completes the proof of the theorem. □

Increasing the capacities of the edge and garbage collection nodes to 2 and connecting each of them to one additional corresponding "dummy" box node of size 1, one can easily prove

Corollary 1 *BAP_D is $\mathcal{NP}$-complete, even if the box sizes have values 1 and 2 and the capacities have value 2 only.*

3 Optimization and Feasibility

Corollary 2 *There is no polynomial time ϵ-approximate algorithm for BAP for any $\epsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$, even if all sizes have values 1 and 2 and capacities have value 2 only.*

Proof. Suppose $\mathcal{A}$ is an ϵ-approximate algorithm for BAP for some fixed, but arbitrary $\epsilon > 0$. We can use $\mathcal{A}$ to solve BAP_D (sizes and capacities restricted to $\{1, 2\}$ and $\{2\}$ respectively) in polynomial time. Let an arbitrary instance (G, c, s) of BAP_D with restricted sizes and capacities be given by the graph $G = (V, E) = (S \cup C, E)$, sizes $s : S \to \{1, 2\}$ and capacities $c : C \to \{2\}$. Add all missing edges F from S to C to obtain the complete bipartite graph $G' = (V, E \cup F)$. Define edge weights

$$w_{ij} = \begin{cases} 1, & ij \in E \\ \lceil (1+\epsilon)|V|^2 + 1 \rceil, & ij \in F. \end{cases}$$

(G', c, s, w) is an instance of BAP. The transformation is polynomial.
Denote by w_{opt} the optimal objective value of (G', c, s, w) and let A be an assignment found by the algorithm $\mathcal{A}$ with quality guarantee ϵ.
If (G, c, s) has a feasible assignment A', A' will also be a feasible assignment for (G', c, s, w), $A' \subseteq E$ and thus A' has costs $|A'|$. This implies for the costs of A

$$\sum_{ij \in A} w_{ij} \le (1+\epsilon) w_{\mathrm{opt}} \le (1+\epsilon)|A'| \le (1+\epsilon)|V|^2.$$

$|V|^2 \le |A'|$ since $|A'| \le |S \times C| \le |S||C| \le |V|^2$.
If (G, c, s) has no feasible assignment, A contains at least one edge from the expensive edge set F and we have

$$\sum_{ij \in A} w_{ij} \ge w_{\mathrm{opt}} \ge \lceil (1+\epsilon)|V|^2 + 1 \rceil > (1+\epsilon)|V|^2.$$

Thus,

$$(G, c, s) \text{ has a feasible assignment } \iff \sum_{ij \in A} w_{ij} \le (1+\epsilon)|V|^2$$

and we can solve BAP_D in polynomial time — a contradiction. □

Corollary 3 *BAP is $\mathcal{NP}$-hard in the strong sense, even if all box sizes have values 1 and 2 and capacities have value 2 only.*

The graphs associated with the instances of BAP_D are not complete. Theorem 1 states that it is $\mathcal{NP}$-complete to determine whether an instance of BAP_D has a feasible solution. In SAPs, the graphs are complete. We will show now that it is then possible to decide in polynomial time the existence of a feasible solution provided the sizes and capacities are bounded.
For any positive integer M, the following decision problem SAP_D^M can be associated with the feasibility question of SAP.

(SAP_D^M) Given an instance (G, c, s, w) of SAP with both $\max_{i \in S} s_i \leq M$ and $\max_{j \in C} c_j \leq M$. Does there exists a feasible assignment in (G, c, s, w)?

To show that SAP_D^M is polynomially solvable for any arbitrary, but fixed M, we will transform SAP_D^M to the *bin packing problem* BP_D^B. It is known that the bin packing problem is polynomially solvable for any fixed "bin capacity" $B \in \mathbb{Z}^+$, see [GJ79].

(BP_D^B) Given a finite set U of items, sizes $s : U \to \mathbb{Z}^+$ for each $u \in U$ and a positive integer k. Does there exists a partition of U into disjoint sets $U_1, \ldots, U_k$ such that the sum of the sizes of the items in each U_i is B or less?

We will call a partition $U_1, \ldots, U_k$, such that the sum of the sizes of the items in each "bin" U_i is B or less a *feasible partition*.

Theorem 2 *SAP_D^M is polynomially solvable for any fixed, but arbitrary bound $M \in \mathbb{Z}^+$.*

Proof. Let $B := 2M + 1$. Then B is fixed for fixed M and BP_D^B is polynomially solvable.
We transform SAP_D^M to BP_D^B. Let an arbitrary instance (G, c, s, w) of SAP_D^M be given by the complete graph $G = (S \cup C, E)$, sizes $s : S \to \{1, 2, \ldots, M\}$, capacities $c : C \to \{1, 2, \ldots, M\}$ and weights $w : E \to \mathbb{Z}^+$ (we can ignore the weights here). We must construct an instance (k, U, s') of BP_D^B, i.e., a set U of items, sizes $s' : U \to \mathbb{Z}^+$ and an integer k, such that (k, U, s') has a feasible partition if and only if (G, c, s, w) has a feasible assignment.
We construct (k, U, s') as follows.
First, we will use $k := |C|$ bins. Each bin j corresponds in one to one manner to the storage location j.
Second, the set of items consists of two types of items. There will be $|C|$ items called "location items" $a_1, \ldots, a_{|C|}$, that will be used to reduce the capacities of the bins to become exactly the capacities of the corresponding storage locations. Each location item a_j has size $B - c_j \geq M + 1 > B/2$. The remainder of the items, called "box items", is formed by the set of boxes S. The box items keep their original sizes $s'(i) = s(i)$ for all $i \in S$. Thus, $U := \{a_1, \ldots, a_{|C|}\} \cup S$ and

$$B \geq s'(i) = \begin{cases} B - c_j \geq M + 1, & \text{if } i = a_j \\ s(i), & \text{if } i \in S. \end{cases}$$

This completes the construction of the instance of BP_D^B. The transformation is polynomial because B is a constant.
We now show that (k, U, s') has a feasible partition if and only if (G, c, s, w) has a feasible assignment.
Suppose A is a feasible assignment in (G, c, s, w). We can construct a feasible partition $U_1, \ldots, U_{|C|}$ as follows. Let $U_j := \{a_j\} \cup \{i \in S : ij \in \delta(j) \cap A\}$, $j = 1, \ldots, |C|$. So each U_j comprises its corresponding location item and the box items, whose corresponding boxes are assigned to location j. Then each item is contained in exactly one bin and the sum of the sizes of the items in each bin U_j is

$$\sum_{i \in U_j} s'(i) = s'(a_j) + \sum_{i \in S : ij \in \delta(j) \cap A} s'(i) \leq B - c_j + c_j = B$$

and $U_1, \ldots, U_{|C|}$ is a feasible partition.
Conversely, suppose (k, U, s') has a feasible partition. Since the sum of the sizes of any two location items a_i and a_j is $s_{a_i} + s_{a_j} = B - c_i + B - c_j = 4M + 2 - c_i - c_j > 2M + 1 = B$, each bin U_j contains exactly one location item. Without loss of generality, we can assume $a_j \in U_j$. Consider the remainder of the items in bin U_j. Their sizes add up to

$$\sum_{i \in S \cap U_j} s'(i) = \sum_{i \in S \cap U_j} s(i) \leq B - s'(a_j) = B - (B - c_j) = c_j.$$

We can thus assign the boxes in $S \cap U_j$ to location j and obtain $A := \bigcup_{j \in C} \{ij : i \in S \cap U_j\}$ as a feasible assignment in (G, c, s, w).
This means that we can obtain a polynomial algorithm for SAP_D^M by first polynomially transforming the instances of this problem to instances of BP_D^B and then solve these instances with a polynomial time algorithm for BP_D^B. $\square$

4 Summary

The complexity status of SAPs is summarized in the following table.

SAP	Maximal Capacity	
Maximal Size	1	≥ 2
1	$\mathcal{P}$ (Assignment Problem)	$\mathcal{P}$ (Transportation Problem)
≥ 2	$\mathcal{P}$ (Assignment Problem)	$\mathcal{NP}$-hard

Table 1: Complexity Status of Storage Assignment Problems

Moreover, the ϵ-approximation problem for arbitrary quality guarantee ϵ is also not polynomially solvable, even if sizes and capacities are restricted to $\{1, 2\}$ and $\{2\}$ respectively, unless $\mathcal{P} = \mathcal{NP}$. This shows, that basically all storage assignment arising in practice are indeed difficult if one wants to solve them to optimality or to proven quality.
On the other hand, it is possible to solve the feasibility problem for SAPs in polynomial time for bounded sizes and capacities.
The complexity results in this article motivate a polyhedral investigation of storage assignment problems to design a cutting plane algorithm.

References

[Abd94] A. Abdel-Aziz Abdel-Hamid, *Combinatorial optimization problems arising in the design and management of an automatic storage system*, PhD dissertation, Technical University of Berlin, 1994.

[AG94] A. Abdel-Aziz Abdel-Hamid and M. Grötschel, *Storage assignment in flexible manufacturing: Polyhedral investigations and a cutting plane algorithm*, Tech. Rep. SC–94–13, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Heilbronner Straße 10, D–10711 Berlin-Wilmersdorf, Germany, 1994. Submitted to SIAM Journal on Optimization.

[GJ79] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[MT81] S. Martello and P. Toth, *An algorithm for the generalized assignment problem*, in Operational Research'81, J. Brans, ed., North-Holland, Amsterdam, 1981, pp. 589–603.

[MT91] S. Martello and P. Toth, *Knapsack Problems — Algorithms and Computer Implementations*, John Wiley and Sons, New York, 1991.