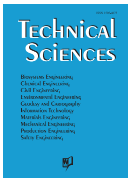
Technical Sciences, 2018, 21(4), 257–269

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE

Quarterly peer-reviewed scientific journal
ISSN 1505-4675
e-ISSN 2083-4527
TECHNICAL SCIENCES
Homepage: www.uwm.edu.pl/techsci/



AN INTEGER OPTIMIZATION MODEL AND ALGORITHMS TO SUPPORT THE COST-REVENUE STUDY AND PROVISORY DESIGNING WAREHOUSES OR OTHER STORAGE OBJECTS

Mikalai Miatselski, Bożena Staruch, Bogdan Staruch

Division of Algebra and Geometry
Faculty of Mathematics and Computer Science
University of Warmia and Mazury in Olsztyn

Received 8 September 2018; accepted 17 October 2018; available online 19 October 2018.

K e y w o r d s: warehouses design, cost-revenue recognition, state space, model and algorithms, nonlinear and integer programming.

A b s t r a c t

An optimization model for the cost–revenue study at the stage of system analysis and preliminary designs of storage objects such as warehouses, containers, packs and similar objects are developed. Our assumptions motivated by warehouses design lead us to a nonlinear integer optimization problem with the only basic constraint. We present algorithmic methods for obtaining the exact solution to the general problem with emphasizing the special case when both the objective and the constraint functions are increasing. The results of the paper may be used in developing software tools intended for supporting designers.

Introduction

An optimization model for the cost–revenue study at the stage of system analysis and provisory designs of storage objects such as warehouses, containers, packs and similar objects are developed. The design includes both: single objects and object complexes.

Correspondence: Bożena Staruch, Katedra Algebry i Geometrii, Wydział Matematyki i Informatyki, Uniwersytet Warmińsko-Mazurski, ul. Słoneczna 54, 10-710 Olsztyn, e-mail: bostar@matman.uwm.edu.pl

The problem of warehouses design is the strategic factor in the success of many businesses and hence there is a need of developing different types of warehouse storage solutions. A lot of companies offer their support in warehouse and layout design (see, e.g., Mecalux, Logistics Bureau). Scientific articles concern mainly layout design (see CHITTRATANAWAT 1999, SINGH, SHARMA 2006, VAN CAMP et al. 1991). A design project of a warehouse should compile as much information as possible, so that the installation fulfils its function and can even adapt to any future needs that may arise. It is essential that designers clearly understand all the characteristics of storing goods: the load unit used, its dimensions and the required dimensions of the shelves and the installation work areas, as well. The dimensions and characteristics of the warehouse infrastructure are essential and must be very accurate information. They are required for the design of shelves, to calculate the capacity of the installations and the distribution of the goods inside the warehouse.

The following simplified sequence of relationships underlies our approach: entrepreneur's revenue from selling storage services is roughly proportional to the total loading of all the storage objects the latter, in turn, is roughly proportional to the entire interior volume of the storage objects and, finally, the interior volume of all the predesigned objects is a function of their dimensions. Such a function may be defined either by an analytical formula or even by an algorithm.

Naturally, the objective function is maximized subject to constraints that express budget limitation derived from landscape peculiarities or by structural or technological norms and regulations. The major financial constraint takes into account costs (prices) of the building materials, raw and fabricated, letting the other construction expenses (e.g. caused by purchasing and installing the equipment or facilities, labor costs etc.) be represented by a collective evaluation. There are also limitations that must be included: access, floors, windows, columns, boxes, lines and power lines are all examples of parts of an installation that influence its design. Furthermore, there are the building regulations that directly affect the calculations of storage structures.

Structural and technology requirements imply that we mainly deal with a nonlinear integer optimization problem. The main version is proved to be an $\mathcal{NP}$-hard problem. We study basic properties of this problem which underlie our algorithms. Examples and applications are given as well.

Although our algorithms are based on searching the state space and have exponential complexity instances of the presented problem with a small number of variables are solved efficiently. In case of a large number of variables it is worth looking for fast sub-optimal algorithms enriched with the appropriate tests for optimality.

The paper is organized as follows. First, we present assumption and conventions that are motivated by practice. Next an optimization problem for three decision variables is described together with its algorithmic solution. Later we generalize this problem to n decision variables. We also discuss the role of the choice of the set of parameters (that corresponds to decision variables) characterizing the predesign shape of construction. Finally, we consider increasing property of both the objective function and the constraint function and we give an algorithm based on this property.

Assumptions and conventions

Assume that an entrepreneur has a land area sufficient to build a logistics center, the main object of which would be a warehouse to provide regular revenue, and therefore to gain profit from the provision of warehousing and storage services for goods, raw materials, semi-finished products, etc., generally called "loading" of the magazine. We also make the following assumptions:

– regular income is quite stable and proportional to the amount of the whole loading stored;

– the size of the loading is related to the interior volume: larger volume creates a predisposition for greater loading. In any case, the lack of space at the occurrence of demand for storage services clearly leads to revenue losses;

– storage capacity depends on the geometric shape and individual dimensions of the warehouse. The most common shapes of real structures are quite simple: a cuboid (possibly with a "gable roof" or a shade), a pyramid or pyramid truncated;

– assuming the geometric shape of the storage structure to be already defined, we have that the values of the basic dimensions of the construction determine the final design of the interior space of the warehouse. Such a project is designed to provide the largest volume of interior space and thereby maximize the potential loading of the warehouse. This means that the dimensions are variable values, selectable in certain intervals and subject to some constraining conditions (in other words, they are supposed to be *decision variables* in appropriate optimization mathematical models);

– as the basic limiting condition, we require that the pre-determined amount of money for warehouse construction be not exceeded. We assume that the costs are dependent on the prices of building materials used for the structural components, i.e. walls, floor, roof and so on. Also, other costs such as design, land preparation and development, energy infrastructure for machines, energy costs of equipment exploitation, labor, etc. are under consideration. The entire sum of costs is presented as a function dependent on decision variables;

– summing up, we would like to develop and explore a model for achieving best business results based on the provision of stock warehousing services at the design stage of storage facilities, depending on the structural components and dimensions of the building.

Formulation of optimization model

We start with an example of an optimization model for a cuboid warehouse. The problem is to determine the size of the warehouse with the greatest volume under budget. We take the following assumptions:

– the warehouse is a cuboidal building described by three parameters: length, width and height denoted, respectively, by x_1, x_2, x_3. In our optimization problem these parameters measured e.g. in meters will play the role of decision variables. Our purpose is to maximize volume of the cuboid, hence the function $\mathrm{Vol}(x_1,\ x_2,\ x_3) = x_1x_2x_3$ will constitute the objective function;

– the parameters values are restricted to intervals determined by lower bounds $l_1,\ l_2,\ l_3$ and upper bounds $u_1,\ u_2,\ u_3$;

– the construction expenses are obtained by summing costs of the main parts of the building such as walls, floor, roof and others which are supposed to be proportional to their surfaces. Let $a,\ b,\ c,\ d$ denote the given unit costs of construction of one square meter of the floor, side and rear walls, roof, and the front wall, respectively;

– the total cost (denoted by Cost) of constructing the warehouse is the sum of all the main parts and some costs e that are independent of the size of the building. Hence, we obtain a mathematical expression for the constraint function:

$$\mathrm{Cost}(x_1, x_2, x_3) = ax_1x_2 + b(2x_2x_3 + x_1x_3) + cx_1x_2 + dx_1x_3 + e;$$

– let B denote the budget limitation for the investment. Thus, we obtain the following constraint on the size of the building: $\mathrm{Cost}(x_1, x_2, x_3) \le B$;

– finally, the problem can be stated as follows: maximize $\mathrm{Vol}(x_1, x_2, x_3)$ subject to:

C1 $\mathrm{Cost}(x_1, x_2, x_3) \le B$,

C2 $l_i \le x_i \le u_i$, $l_i < u_i$ and l_i, u_i are nonnegative reals for every $i = 1, 2, 3$;

– the integer form of this problem assumes that x_i, l_i, u_i are nonnegative integers.

The integer optimization problem with three decision variables will be abbreviated as IP3 (Integer Problem with 3 decision variables). Certainly, Vol and Cost in IP3 can be different from those presented in above.

The above problem can be stated for continuous variables and hence certain attempts to apply real analysis optimization methods may be done. However, as we assume that the objective function describes the volume not only of a warehouse but also of a container or a collection of containers, the Vol function may have many different types and forms. Practically, any arbitrary function, even defined by an algorithm (with no explicit formula given) can be considered as an objective function. The constraint function Cost is assumed to express the sum of all the construction costs. Therefore, there is no reason to assume any ‘nice’ analytic property such as differentiability, continuity or convexity of any of these two functions.

In this situation, an approach based on partial enumeration of feasible solutions, seems to be the only practically efficient technique. Certainly, we realize that even decision problem of simple membership in the feasible region may turn out to be hard. Nevertheless, strong variability of objective functions justifies this approach in our work. Moreover, we assume that, as a rule, in practical applications the variables are integer. For example, the walls are made of some normalized components (modules) of a given size or the material is stored in containers of a given size. For this reason, we assume that the parameters are measured in some units depending on a specific situation and we consider the values of variables to be integer multiples of these units. Consequently, these units are used to calculate costs. Hence, we focus our attention on integer optimization problems such as IP3 or more generally, IPn, where n is the number of decision variables.

Algorithmic solution of IP3

The set of all the integer points satisfying C2 will be called the state space. To solve the IP3 problem we need to scan the state space S:

$$S = [l_1, u_1] \cdot [l_2, u_2] \cdot [l_3, u_3],$$

where $[l_i, u_i] = \{l_i, l_i + 1, \ldots, u_i\}$ for every $i = 1, 2, 3$.

If a point in S satisfies the constraint C1 i.e. is a feasible solution, then the value of Vol is calculated, and finally, the points with the greatest value of Vol are the optimal solutions.

Below the reader will find the pseudo-code of an algorithm which for every point in S checks if this point is a feasible solution, and if the answer is “yes” it calculates the value of Vol. The points with currently the highest value of the variable V are remembered (REMEMBER) and they are cleared (CLEAR) as soon as V becomes greater.

Algorithm-3DmaxVolume

```
Vol$:=0$; $V:=0$
FOR $l := u_l$ DOWNTO $l_l$
    FOR $w := u_w$ DOWNTO $l_w$
        FOR $h := u_h$ DOWNTO $l_h$
            IF (Cost$(l, w, h) \leq B$) THEN
                $V :=$ Vol$(l, w, h)$
```

```
                    IF $(V = \mathrm{Vol})$ THEN
                    REMEMBER $(l, w, h)$
                    END IF
                    IF $(V > \mathrm{Vol})$ THEN
                    $\mathrm{Vol} := V$
                    CLEAR
                    REMEMBER $(l, w, h)$
                    END IF
                END IF
            END FOR
        END FOR
    END FOR
    END
```

The above algorithm scans the whole state space, so it makes $(1 + u_l - l_l) \cdot (1 + u_w - l_w) \cdot (1 + u_h - l_h)$ of steps. To improve effectivity of solving IC3 some deeper analysis of the functions Vol and Cost should be done.

The IPn model

We present here a general optimization model IPn. By assuming a general point of view, we are able to consider any solid figure (or even a collection of solid figures) the volume of which is being maximized. We assume that such a figure is described by a set of parameters (*describing parameters*) that characterizes this figure in the sense that there is a function on the set of parameters returning its volume. Assume that we have:

– describing parameters $x_1, \dots, x_n$ that fully characterize the figure as a rigid body;

– the volume function $\mathrm{Vol}(x_1, \dots, x_n)$ to be maximized;

– the cost function $\mathrm{Cost}(x_1, \dots, x_n)$ limited by a fixed number B;

– integer values of describing parameters that are constrained by the integer lower and upper bounds $0 \le l_i \le u_i$ so that $l_i \le x_i \le u_i$ for every $i = 1, \dots, n$.

The state space of this problem consists of integer points of the n-orthotope S:

$$S = [l_1, u_1] \cdot \dots \cdot [l_n, u_n],$$

where $[l_i, u_i] = \{l_i, l_i + 1, \dots, u_i\}$ for every $i = 1, \dots, n$.

Then the IPn problem is formulated as follows:

Maximize $\mathrm{Vol}(x_1, \dots, x_n)$, subject to

C1 $\mathrm{Cost}(x_1, \dots, x_n) \le B$, where B is a positive real, C2 $l_i \le x_i \le u_i$ and $0 \le l_i \le u_i$, where x_i, l_i, u_i are nonnegative integers for $i = 1, \dots, n$

C3 $S \subseteq D_{\mathsf{Vol}} \cap D_{\mathsf{Cost}}$, where D_{Vol}, D_{Cost} are the domains of Vol and Cost, respectively.

Any solution of the above problem is based on searching (exhaustively or partially) the state space, checking constraints and choosing the optimal solution (or solutions). Obviously, the cardinality of S is equal to:

$$\mathrm{card}(S) = (1 + u_1 - l_1) \cdot \dots \cdot (1 + u_n - l_n) \le [\max\{(1 + u_1 - l_1), \dots, (1 + u_n - l_n)\}]^n .$$

Therefore, it is important that the chosen set of describing parameters be minimal. The choice of describing parameters is crucial in faster methods for solving the problem.

The 3DmaxVolume algorithm easily generalizes (n nested loops FOR) to the algorithm nDmaxVolume solving IPn. Notice that the condition C3 guarantees the correctness of this generalization. If C3 is not satisfied, the nDmaxVolume algorithm should be improved by introducing the mechanism for checking if the current point belongs to $D_{\mathsf{Vol}} \cap D_{\mathsf{Cost}}$. If the answer is “no” the next point is taken. As the nDmaxVolume algorithm is of exponential time complexity, other quick methods for solving IPn are worth of considering including methods based on some kind of heuristics. For example, genetic or other evolutionary algorithm would bring a suboptimal solution in better time. It depends on preferences of the entrepreneur if the exact solution with bigger cost is required, or if a non-exact suboptimal solution is good enough to use.

Choosing describing parameters

The next two examples show that even for a fixed solid figure there are various choices of describing parameters. Every choice has some advantages and disadvantages, as well. The solid figure in Example 1 is a cone and the value of the cost function is given as the value of the lateral surface area. It can be characterized by two parameters e.g. radius r and height h, or radius r and slant height l.

Example 1 (Cone):

– $\mathsf{Vol}(r,h) = \frac{1}{3}\pi r^2 h$, $\mathsf{Cost}(r,h) = \pi r\sqrt{r^2 + h^2}$. Here, the constraint C3 is satisfied:

– $\mathsf{Vol}(r,l) = \frac{1}{3}\pi r^2\sqrt{l^2 - r^2}$, $\mathsf{Cost}(r,l) = \pi r l$. Here, the domain of Vol is restricted to pairs $l > r$. In this case, the improved version of *2*DmaxVolume should be used or some change in describing parameters should be done. Let us introduce a new decision variable $x = l - r$, $x \ge 0$. Then $\mathsf{Vol}(r,x) = \frac{1}{3}\pi r^2\sqrt{(x + r)^2 - r^2}$, $\mathsf{Cost}(r,x) = \pi r(x + r)$ and C3 is satisfied.

When the solid figure is a conical frustum, created by slicing the top off a cone with a cut parallel to the base, we need at least three parameters.

Example 2 (Conical Frustum). Describing parameters: L, r, R, where R is the radius of the base, r is the radius of the top, and L is the slant height. Then $\mathrm{Vol}(L,r,R) = \frac{1}{3}\pi\sqrt{L^2 - (R-r)^2}(R^2 + Rr + r^2)$, and $\mathrm{Cost}(L,r,R) = a\pi R^2 + b\pi r^2 + c\pi(R+r)L$, where a, b, c are unit costs of constructing the base, the top and the lateral surface, respectively. Here the domain assumptions that $R > r$ and $L > R - r$ can be used. By substituting new variables $x = R - r$, $x > 0$ and $y = L - x$, $y > 0$ we obtain a model with describing parameters x, y, r that satisfies C3.

NP-hardness of IPn

To show that IPn is NP-hard we present a polynomial (linear) reduction of the very known KNAPSACK problem) to IPn. To learn more on KNAPSACK problems see e.g. KELLERER et al. (2004).

Based on a KNAPSACK instance:

maximize $a_1x_1 + \cdots + a_nx_n$, subject to

$$c_1x_1 + \cdots + c_nx_n \leq C,$$

where x_i are nonnegative integers for every $i = 1, \ldots, n$

create the following instance of IPn:

maximize $\mathrm{Vol}(x_1, \ldots, x_n) = a_1x_1 + \cdots + a_nx_n$, subject to

$$\mathrm{Cost}(x_1, \ldots, x_n) = b_1(2a_1 + 2)x_1 + \cdots + b_n(2a_n + 2)x_n \leq B,$$

where x_i are positive integers and $b_i = \frac{c_i}{2a_i+2}$ for every $i = 1, \ldots, n$ and $B = C + (a_1 + \cdots + a_n)$.

It is easy to see that the model IPn describes the situation of designing a collection of n bins of sizes $1, a_i, x_i$, $i = 1, \ldots, n$. Such a collection may be intended for serial batch manufacturing. The very creation of the model instance uses linear time $O(n)$. Any optimal solution of the IPn is an optimal solution of the KNAPSACK instance. NP-hardness of KNAPSACK implies NP-hardness of IPn.

Increasing assumption

In this section we consider an assumption that the functions Vol and Cost are increasing. This allows as to propose an algorithm solving IP2 in linear time and to lower time complexity of IPn.

Basic definitions and properties

Let $\breve{x} = (x_1, \dots, x_n)$ and let $(\breve{x}, {}^{x_i}/_a) = (x_1, \dots, x_{i-1}, a, x_{i+1}, \dots, x_n)$ denote an n-tuple obtained from $\breve{x}$ by substitution a for x_i, where $1 \le i \le n$. Analogously, $(\breve{x}, {}^{x_i}/_a, {}^{x_j}/_b)$ stands for an n-tuple obtained from $\breve{x}$ by substitution a for x_i and b for x_j, where $1 \le i < j \le n$.

Let f be a real function of n variables $x_1, \dots, x_n$ and let $D_f \subseteq R^n$. We say that:

– f is increasing on variable x_i on a set $A \subseteq D_f$ if and only if for any $a < b$ with $(\breve{x}, {}^{x_i}/_a), (\breve{x}, {}^{x_i}/_b) \in A$ it holds that $f(\breve{x}, {}^{x_i}/_a) \le f(\breve{x}, {}^{x_i}/_b)$;

– f is strictly increasing on variable x_i on a set $A \subseteq D_f$ if and only if for any $a < b$ with $(\breve{x}, {}^{x_i}/_a), (\breve{x}, {}^{x_i}/_b) \in A$ it holds that $f(\breve{x}, {}^{x_i}/_a) < f(\breve{x}, {}^{x_i}/_b)$;

– f is increasing on a set $A \subseteq D_f$ if and only if f is increasing on a set A on every variable;

– f is strictly increasing on a set $A \subseteq D_f$ if and only if f is strictly increasing on a set A on every variable.

Example 3:

– in Example 1.1, the functions Vol and Cost are strictly increasing on their domains;

– in Example 1.2, the function $\mathrm{Vol}(r, l)$ is strictly increasing on l and it would not be increasing on r. After substitution, Vol(r, x) is strictly increasing on its domain;

– in Example 2, $\mathrm{Vol}(L, r, R)$ is strictly increasing on variables r, L and would not be increasing on R. The cost function Cost is strictly increasing on its domain. After substitution, $\mathrm{Vol}(x, r, y)$ is strictly increasing on its domain;

– as the state space S is a Cartesian product, the natural ordering $\preccurlyeq$ (state space ordering) on S is determined. Namely, for any $(x_1, \dots, x_n), (y_1, \dots, y_n) \in S$;

– $(x_1, \dots, x_n) \preccurlyeq (y_1, \dots, y_n)$ *if* $x_i \le y_i$ for every $i = 1, \dots, n$;

– $(x_1, \dots, x_n) \prec (y_1, \dots, y_n)$ *if* $x_i \le y_i$ for every $i = 1, \dots, n$ and there is $j \le n$ with $x_j < y_j$.

Directly from definition of the order $\preccurlyeq$ we have that a function $f: R^n \to R$ is increasing (strictly increasing) on the state space S if and only if $S \subseteq D_f$ and for any $\breve{x}, \breve{y} \in S$, if $\breve{x} \prec \breve{y}$ then $f(\breve{x}) \leq f(\breve{y})$ ($f(\breve{x}) < f(\breve{y})$).

Proposition 1. If the objective function Vol is:

– increasing, then for any optimal solution $\breve{a}$ there exists a maximal (in the order $\preccurlyeq$ restricted to feasible solutions) element $\breve{m}$ which is an optimal solution of IPn;

– strictly increasing, then any optimal solution of IPn is a maximal element among feasible solutions.

To see the correctness of the above proposition, let $\breve{a}$ be an optimal solution of IPn with $\breve{a} \prec \breve{x}$ for a feasible solution $\breve{x}$. If Vol is increasing, then $\mathrm{Vol}(\breve{a}) \leq \mathrm{Vol}(\breve{x})$ and, as $\mathrm{Vol}(\breve{a})$ is maximal we have $\mathrm{Vol}(\breve{a}) = \mathrm{Vol}(\breve{x})$, which means that $\breve{x}$ is an optimal solution. If Vol is strictly increasing, then $\mathrm{Vol}(\breve{a}) < \mathrm{Vol}(\breve{x})$, which yields a contradiction.

A linear algorithmic solution of IP2

Proposition 1 and the next observation will be used in our algorithms in the sequel. Let us say that $\breve{x} \in S$ satisfies the cost constraint if $\mathrm{Cost}(\breve{x}) \leq B$ and $\breve{x}$ does not satisfy the cost constraint, in the opposite case.

Proposition 2. If the function Cost is increasing on its domain and $S \subseteq D_{\mathrm{Cost}}$ then for any $\breve{x}, \breve{y} \in S$

– if $\breve{x}$ satisfies the cost constraint and $\breve{y} \preccurlyeq \breve{x}$ then $\breve{y}$ satisfies the cost constraint;

– if $\breve{x}$ does not satisfy the cost constraint and $\breve{x} \preccurlyeq \breve{y}$ then $\breve{y}$ does not satisfy the cost constraint, either.

This proposition can be used to fit upper bounds u_i in IPn C2 as follows:

Algorithm-Fitting (x_i)

```
WHILE (($\mathrm{Cost}(\breve{l}, {}^{l_i}/_{u_i}) > B$) AND $(u_i > l_i)$)
        $u_i := u_i - 1$
END WHILE
OUTPUT $u_i$
END
```

Assume from now on that the upper bound for every decision variable is set by the above algorithm, and that Vol is strictly increasing and Cost is increasing on S (increasing assumptions). First, we will present the algorithm for IP2 (based on Proposition 2) that finds maximal feasible solutions which are, at the same time, optimal solutions (by Proposition 1).

Let $S \subseteq D_{\mathrm{Vol}}$ be a 2-D state space. Let x, y be the decision variables with $k \le x \le K$, $l \le y \le L$. We present an algorithm solving IP2 in at most $(1 + K - k) + (1 + L - l)$ steps.

Algorithm 2D-IncreasingAssumptions (2D-IA)

```
$x := k; y := L$
WHILE (($x \le K$) AND ($y > l$))
    WHILE (($\mathrm{Cost}(x, y) \le B$) AND ($x \le K$))
    $x := x + 1$
    END WHILE
    REMEMBER $(x - 1, y)$
    WHILE (($\mathrm{Cost}(x, y) > B$) AND ($y > l$))
    $y := y - 1$
    END WHILE
END WHILE
END
```

The algorithm starts in the North-West corner ($x = k; y = L$) of the rectangle $[k, K] \cdot [l, L]$. As (k, L) satisfies the cost constraint, (k, b) satisfies the cost constraint for every $b < L$. In this situation, we should move right ($x = x + 1$) and repeat this procedure until (x, y) does not satisfy the cost constraint. And again, we move down then right and so on. The algorithm ends when we get to the bottom or to the right-side boundary of the rectangle.

Time complexity of this algorithm (the number of visited points) is not greater than the length of the path starting in the North-West corner and ending in the South-East corner, which is equal to $(1 + K - k) + (1 + L - l) - 1$. The usage of space is not greater than $\min(1 + K - k; 1 + L - l)$, because every line (row or column) contains at most one maximal element. To obtain optimal solutions it is enough to calculate the values of Vol for every remembered point and choose the best ones.

An algorithmic solution of IP3

Consider IP3 and choose a variable (say, z with $m \le z \le M$) and repeat the 2D-IA algorithm for x, y and every fixed z. For simplicity, let

2D-IA$(x; k; K; y; l; L; \mathrm{Cost}(x, y, z))$

mean that 2D-IA runs for variables x, y with the cost function $\mathrm{Cost}_z(x, y) = \mathrm{Cost}(x, y, z)$ is for a fixed value z, bounded as $k \le x \le K$, $l \le y \le L$.

Algorithm 3D-IncreasingAssumptions (3D-IA)

```
FOR $z: = m$ TO $M$
    RUN 2D-IA$(x; k; K; y; l; L; \mathrm{Cost}(x, y, z))$
END FOR
END
```

This algorithm needs at most $(1 + M - m) \cdot \big((K - k) + (L - l) + 1\big)$ steps. Algorithm 3D-IA easily generalizes to higher dimensions (nD-IA algorithm) by using the appropriate number of FOR loops. Notice that the nD-IA algorithm can be used under assumption that the Vol and the Cost function are increasing only on a pair of decision variables.

A slight improvement of the nD-IA algorithm

Assume that Vol and the Cost are increasing functions on *S*. Notice that n D-IA solves IPn in time $s_1 s_2 \dots s_{n-2}(s_{n-1} + s_n - 1)$, where $s_i = 1 + u_i - l_i$ for $i = 1, 2, \dots, n$. Moreover, if the variables are ordered according to increasing values of s_i i.e. $s_1 \le s_2 \le \dots \le s_n$ then

$$s_1 s_2 \dots s_{n-2}(s_{n-1} + s_n - 1) \le s_{\pi(1)} s_{\pi(2)} \dots s_{\pi(n-2)}\big(s_{\pi(n-1)} + s_{\pi(n)} - 1\big),$$

where π is a permutation of n.

To show the last property consider $E_1 = s_1 s_2 \dots s_{n-2}(s_{n-1} + s_n - 1)$ and
$E_2 = s_1 s_2 \dots s_{i-1} s_{i+1} \dots s_{n-1}(s_i + s_n - 1)$ for some $1 \le i \le n - 2$. Then
$E_1 = s_1 s_2 \dots s_{n-2}(s_{n-1} - 1) + s_1 s_2 \dots s_{n-2} s_n$ and
$E_2 = s_1 s_2 \dots s_{n-2}(s_{n-1} - 1) + s_1 s_2 \dots s_{i-1} s_{i+1} \dots s_{n-1} s_n$

As the first components are equal, we compare the second ones:

$$\frac{s_1 s_2 \dots s_{n-2} s_n}{s_1 s_2 \dots s_{i-1} s_{i+1} \dots s_{n-1} s_n} = \frac{s_i}{s_{n-1}} \le 1.$$

This yields that $E_1 \le E_2$.

Conclusions and remarks

In the paper, we introduced into consideration and investigated a problem of warehouse design under budget limitation. Certainly, this motivation leads to the IPn problems which can be also used in other applications e.g. allocation problem (see IBARAKI and KATOH 1988). The proposed IPn model has a single constraint however it may be extended to a multi-constraint model.

When we make increasing assumptions on the IPn problem we obtain an instance of the nonlinear integer knapsack problem (see LI DUAN, SUN XIAOLING

2006). Therefore, methods for solving IPn can be used in a very wider class of problems and we think that developing methods presented in this paper is worth of effort (for example, an effective generalization of 2D-IA to higher dimensions is desirable).

References

CAMP D. VAN, CARTER M., VANNELLI A. 1991. *A nonlinear optimization approach for solving facility layout problems*. European Journal of Operations Research, 57: 174–189.

CHITTRATANAWAT S. 1999. *An integrated approach for facility layout, P/D location and material handling system design*. International Journal of Production Research, 37(3): 683-706.

IBARAKI T., KATOH N. 1988. *Resource allocation problems: algorithmic approaches*. MIT Press, Cambridge, Mass.

KELLERER H., PFERSCHY U., PISINGER D. 2004. *Knapsack Problems*. Springer Science & Business Media, Berlin.

LI D., SUN X. 2006. *Nonlinear Integer Programming*. International Series in Operations Research & Management. Springer US.

Logistics Bureau, https://www.logisticsbureau.com.

Mecalux, https://www.mecalux.com.

SINGH S.P., SHARMA R.R.K. 2006. *A review of different approaches to the facility layout problems*. The International Journal of Advanced Manufacturing Technology, 30: 425–433.