RAIRO-Oper. Res. 55 (2021) 3379–3397
https://doi.org/10.1051/ro/2021160

RAIRO Operations Research
www.rairo-ro.org

DESIGNING SCREEN LAYOUT IN MULTIMEDIA APPLICATIONS THROUGH INTEGER PROGRAMMING AND METAHEURISTIC

Pedro Henrique González[1,*], Glauco Amorim[1], Ueverton S. Souza[2], Igor Morais[1], Joel dos Santos[1], Vanessa de A. Guimarães[1] and Glaydston M. Ribeiro[3]

Abstract. Binding audiovisual content into multimedia applications requires the specification of each media item, including its size and position, to define a screen layout. The multimedia application author must plan the *application's screen layout* (ASL), considering a variety of screen sizes where the application shall be executed. An ASL that maximizes the area occupied by media items on the screen is essential, given that screen space is a valuable asset for media broadcasters. In this paper, we introduce the Application Screen Layout Optimization Problem, and present its $\mathcal{NP}$-hardness. Besides, two integer programming formulations and an *Iterated Local Search* (ILS) metaheuristic are proposed to solve it. The efficiency of the proposed methods is evaluated, showing that the metaheuristic achieves better results and is at least 12 times faster, on average, than the mathematical formulations. Also, the proposed approaches were compared to a layout design algorithm, showing their effectiveness.

Mathematics Subject Classification. 90-05.

Received April 12, 2021. Accepted October 18, 2021.

1. Introduction

In multimedia applications an *application's screen layout* (ASL) consists of the disposition of visual content on the screen. It is usually defined by indicating the position of each media item relative to the screen, where for each media item its attributes are set in relation to the screen top-left corner. In general, languages enable the application author to define such values either in pixels or percentage [1].

The multimedia application author, usually a content producer, has to plan the ASL and, possibly, consider a variety of screen sizes where the application shall be executed. Analyzing different interactive multimedia applications transmitted in real applications and/or stored in an application repository called Clube NCL [7], we can find presentation characteristics common to most. It was noticed that the media objects were arranged using a grid structure to determine candidate positions. These structure is important because all the media objects arrangement use these points as candidates to create the application layout.

Keywords. Screen layout planning, integer programming, ILS.

[1] CEFET/RJ - Federal Center for Technological Education of Rio de Janeiro, Rio de Janeiro, Brazil.
[2] UFF - Fluminense Federal University, Rio de Janeiro, Brazil.
[3] UFRJ - Rio de Janeiro Federal University, Rio de Janeiro, Brazil.
*Corresponding author: pegonzalez@eic.cefet-rj.br

© The authors. Published by EDP Sciences, ROADEF, SMAI 2021

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

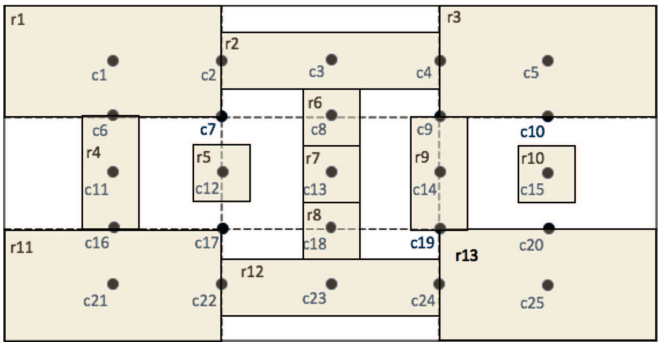


FIGURE 1. An application's screen layout for a screen R with a grid of center points G_C.

Although several tools have been used to rearrange media objects on a screen [3, 9, 10], whether on the smartphone or television, these techniques are not intended to create an optimized layout. The literature's known tools do not guarantee that the layout produced will be maximized concerning the display screen's occupation. Since an ASL that maximizes the area occupied on the screen by media items is essential, due screen space is a valuable asset for media broadcasters, in this paper, we introduce and study the *Application Screen Layout Optimization Problem (ASLOP)*, the problem of finding an ASL (a subset of items and their positions) from a set of media items that, when distributed on the screen, maximize the occupied area and avoids overlap of items. This problem is especially difficult when the number of media items to be distributed on the screen grows. Its description is as follows.

APPLICATION SCREEN LAYOUT OPTIMIZATION PROBLEM

Input: A large rectangle R (the application screen) with integer dimensions, a set of smaller rectangles M (set of media items) also having integer dimensions, and a set C of points within the screen R forming a grid G_C (candidate central points of media items) where the distance between consecutive points of G_C on the x-axis and the y-axis are not necessarily the same.

Goal: Find a subset S of pairs (r, c), $r \in M$, $c \in C$ (media items and their center's position) which fits inside the screen, maximizing the covered area of R without overlapping items.

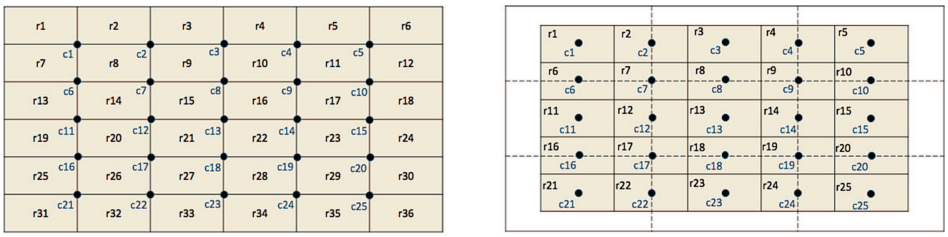
Figure 1 illustrates an application's screen layout obtained from a set of media items for a screen R with a grid of central points G_C.

Following the typology defined in [38], from a combinatorial optimization point of view, ASLOP can be categorized respectively as (i) *two-dimensional*, since it considers both axes x and y; with a (ii) *strongly heterogeneous assortment*, since there are many items of many different shapes or sizes; it has (iii) *one large object* (the screen); and (iv) *regular small items*, given that media items are rectangles.

In fact, ASLOP fits in the Cutting and Packing class of problems. Cutting and Packing (C&P) is a class of optimization problems with a wide range of applications in resource management, such as wood or glass industries, warehousing context, newspaper paging and, most recently, web browser layout [13, 38, 39].

In particular, ASLOP is closely related to widely studied optimization problems such as TWO-DIMENSIONAL KNAPSACK, RECTANGLE PACKING, and GUILLOTINE CUT PROBLEM.

Although there are several problems related to fitting rectangles within a larger one, due to the constraints concerning the grid of central points, none of these classical problems correctly model the APPLICATION SCREEN



(A) (B)

FIGURE 2. (A) A feasible solution for TWO-DIMENSIONAL KNAPSACK/RECTANGLE PACKING; (B) an optimal solution for ASLOP considering only media items having the same shape.

LAYOUT OPTIMIZATION PROBLEM. The classical problems do not consider that each rectangle in the solution needs to be assigned to a candidate center. Once this characteristic is clear, it is not difficult to verify that ASLOP behaves differently than TWO-DIMENSIONAL KNAPSACK, RECTANGLE PACKING, and GUILLOTINE CUT PROBLEM, for example. Figure 2 shows that a solution for RECTANGLE PACKING as well as TWO-DIMENSIONAL KNAPSACK may not be a solution for ASLOP.

In addition, Figure 1 shows that a solution for ASLOP, may not be a solution for the GUILLOTINE CUT PROBLEM as well as Figure 2 illustrates that an optimal solution for GUILLOTINE CUT PROBLEM may not be a solution for ASLOP.

Therefore, one may conclude that although ASLOP belongs to the same class of problems such as TWO-DIMENSIONAL KNAPSACK, RECTANGLE PACKING and GUILLOTINE CUT PROBLEM, none of them can be used to solve ASLOP. Other researches that work with problems similar to ASLOP can be found in [4,6,11,18,19,23,36]. However, given its specific characteristics, previous techniques may not be directly applied to solve it.

Motivated by its real applicability in the area of Webmedia, as well as its challenging combinatorial nature from the point of view of the Optimization field, in this paper, we analyze the complexity of the ASLOP problem as well as develop tools to solve it. First, we prove that the problem is $\mathcal{NP}$-hard then, from a parameterized complexity perspective, we show that the problem is fixed-parameter tractable with respect to the area to be covered. Next, we present two integer programming formulations. Moreover, considering the limitations of the two proposed integer programming formulations to find high quality solutions and prove optimatility, an *Iterated Local Search* (ILS) metaheuristic was also proposed to solve the ASLOP. The efficiency of both formulations and the ILS was evaluated and compared through the analyzes of the covered area for several data sets.

The remaining of this paper is organized as follows. Section 1.1 presents the problem's motivation inspired by multimedia applications and related works to solve the applied problem in practice. Section 2 shows an $\mathcal{NP}$-hardness proof for ASLOP, and present a kernelization algorithm which allows us to observe that the problem is fixed-parameter tractable ($\mathcal{FPT}$) concerning the area to be covered by the media items as parameter. Section 3 presents two integer programming formulations to solve ASLOP and an ILS metaheuristic. Section 4 presents experiments using the proposed formulations and a comparison between them. Finally, Section 5 concludes this paper and presents future work.

1.1. Motivation and related works

The multimedia production pipeline for digital TV and IPTV includes steps from the planning of the audiovisual content to be presented, their capture and binding into an application latter deployed to the spectator [5]. Binding audiovisual content into a multimedia application is usually performed using specific domain languages, such as HTML5 (*HyperText Markup Language*) [33], SMIL (*Synchronized Multimedia Integration Language*) [32] and NCL (*Nested Context Language*) [20]. Those languages focus on presenting audiovisual content, such as

videos, audios, and images, synchronized in time and distributed along the exhibition device screen. Recently, some languages are able to provide synchronization between audiovisual content and sensory effects [27].

Initiatives such as Hybrid broadcast broadband TV (HbbTv) [17] and Integrated broadcast-broadband systems [21] indicates that Digital TV marketplace is an interesting research area. In this scenario, some studies such as [24, 25] present that broadcasted television shows are becoming more interactive allowing even home viewers to be part of them. Therefore, interactive TV applications with enjoyable layouts are desirable for users.

Many research attempts to address the designing of screen layouts in multimedia applications in different ways. One approach is to provide authoring tools to ease the creation of the ASL, such as [3, 9, 10], where tools are proposed to represent media items and enable the author to rearrange such medias in the tool interface. Once the ASL is created, the tool generates the corresponding code in the language used for the application creation.

CSS Flexible Box Layout enables the author to define a region on the screen (called flex container and flex items) composed of a set of spatial properties, such as direction, wrap, and justify-content, and alignment [34]. It focuses on space distribution in a primary axis using a bottom-up approach to arrange items. It can also use a content-sizebased line-wrapping system to control its secondary axis. CSS Grid Layout controls the sizing and positioning of regions and their contents [35]. Unlike CSS Flexible Box Layout, it works considering the arrangement of items in both dimensions. Using CSS Grid Layout authors can adapt the ASL to changes in the presentation device, such as factors, orientation, and available space. It combines CSS media queries with CSS properties that control the disposition of the grid container and its children during a presentation.

Adaptive layouts are templates where the author defines the ASL by means of predefined arrangements, such as grids and flows[1] [1]. At processing time, media items are distributed on the screen according to the chosen arrangement definition, such as its size, its items size and so on. Such approach works associating media items with a given layout arrangement (grid or flow) in the order they are declared in the document. Therefore, while rendering the layout, presentation characteristics are created according to the number of media items associated with each layout arrangement, following a similar approach to CSS [35].

Depending on the approach used for providing an adaptive layout, some media items may not be displayed because the screen size was not sufficient to accommodate them. W3C [34, 35] requires the presentation of all available items on the available screen. When the available screen area is not sufficient for presenting all items, it does not provide a solution. Amorim *et al.* [1] creates a partition considering the media item's order in the document. CSS uses scroll bars to enable the presentation of all media items.

Although several tools have been used to rearrange media objects on a screen, whether on the smartphone or on television, these techniques are not intended to create an optimized layout. These tools do not guarantee that the layout produced will be maximized in relation to the occupation of the display screen.

To summarize, we propose tools to support multimedia application authors to create an ASL that maximizes the occupied area on the screen, thus solving the ASLOP. We follow an alternative approach where we neither use the order of media items declared in the document to create the spatial arrangement nor create media item partitions when the available space is not sufficient to accommodate them all [1]. Instead, the proposed approach uses integer programming formulations to choose media items that maximize the occupied area. Some media items may not be displayed because the screen size was not sufficient to accommodate them. However, it is possible to perform a second round to choose the next subset of media items to be displayed.

From a practical standpoint, the rational use of the screen becomes even more relevant during the COVID-19 pandemic, since most parts of the daily activities are being performed online. These activities are diverse, including from e-commerce to home-office. As a result, platforms used to work meetings grew vertiginously: Google Hangout, for instance, grew to 60% daily in March, while Zoom jumped from 10M to 220M users from 12/2019 to 03/2020 [28].

Regarding trade, according to a Brazilian report about Economy and Consumption during the pandemic, although the Trade Sales Index of Brazilian Retailer registered 25.1% retraction in the same month of the

[1]The paper also proposes other predefined arrangements, but for simplicity, we do not address them here.

previous year, the e-commerce has grown 32.6% in the number of orders [28]. Then, considering that the maximization of the screen use could promote a more significant number of ads incentivizing e-commerce, this research becomes relevant to help, albeit indirectly, in the marketing and sales areas.

2. Computational complexity

Once having explained the importance of ASLOP to the field of study and the applied area, it is necessary to show the difficulty in solving it, which combined with its relevance, would justify our study.

Theorem 2.1. *The Application Screen Layout Optimization Problem is $\mathcal{NP}$-hard.*

Proof. The proof uses a reduction from the Subset Sum problem, shown to be NP-hard by [22]. In Subset Sum we are given n elements, $\{a_1, \ldots, a_n\}$ and a target b. We are asked to determine whether there is a subset of elements a_i which adds up exactly to b.

Let us consider an instance of Subset Sum, formed by a finite set A, such that $|A| = n$ and each $a_i \in A$ is an integer greater than zero. Moreover, consider b a positive integer. From this instance we construct an instance of ASLOP as follows:

- create a rectangular screen M of dimension $1 \times b$;
- for each $a_i \in A$, create a rectangular media item m_i of dimension $1 \times a_i$;

Now, consider the bottom-left and top-right corners of the screen R at points $(0, 0)$ and $(1, b)$ of the Cartesian space:

- Let G_C be the grid formed starting at the point $(0, 0)$ in such a way that the distance between consecutive points of G_C on the x-axis as well as the y-axis is 0.5, and the top-right point is $(1, b)$.

Since, by construction, the height of the screen and of all media items are equal to 1, it is easy to see that there is a subset $X \subseteq A$ with $\sum_{x \in X} x = b$ if and only if there is a subset Z of media items that occupy the entire screen area.

If the reader feels more comfortable having elements with both height and width of non-unitary value, to modify the proof, just create a media of size $r + 1 \times b$, where $r = \sum_{a_i \in A} a_i$ (the sum of the values in A), and b is the target value of Subset Sum. In addition, we make the screen size equal to $r + 2 \times b$, and increase the grid proportionately. In this way, a solution occupying the entire screen will exist if and only if, in addition to using the media of size $r + 1 \times b$, there is a set of medias that correspond to a Subset Sum solution. □

As shown in Theorem 2.1, the problem remains NP-hard even when both the screen and the input medias have height equal to one. Next, we present an analysis of the relationship between the complexity of the problem and the size of the solution. More precisely, we show that the problem is *fixed-parameter tractable* concerning the size of the solution as parameter. More details on Parameterized Complexity can be found in [8] and [12].

Theorem 2.2. *The problem of determining whether there is a set of media items that cover an area of size at least k of the application screen without overlapping them is fixed-parameter tractable when parameterized by k.*

Proof. Considering the problem of determining whether there is a solution covering an area of size at least k (the size of the solution to be found), a *kernel* can be obtained by applying the following reduction rules:

(1) Remove all media that do not fit the screen;
(2) If the sum of the remaining items areas is less than k, return *No*;
(3) If any remaining item has an area of size at least k, return *Yes*.

If Rules 2 or 3 were applied then the problem was solved. Otherwise, the number of different media items shapes is bounded by k^2. Since the height/width of the medias are integers, a minimal set of media covering an area of size at least k contains no more than k items. Therefore, the following reduction rule can be safely applied.

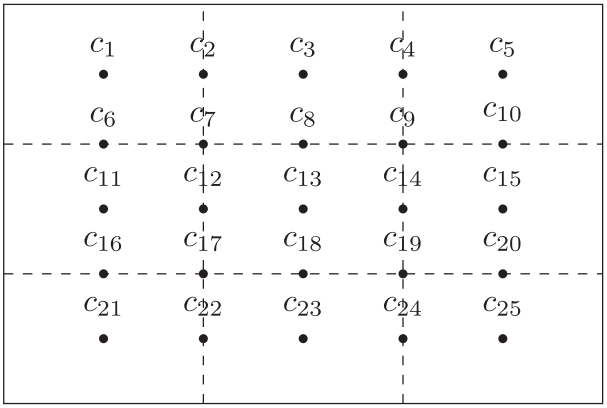


FIGURE 3. Screen grid and possible centers. The figure presents a 3x3 grid, considering the smallest dimensions find in media items. Centers are distributed from the upper-left corner to the bottom-right corner.

(4) For each media item shape remove all but k items.
Now, the number of remaining media items is bounded by k^3.
(5) If the distance between consecutive points of G_C on the x-axis (as well as the y-axis) is greater than k, contract it to k by applying a directional scaling.
Note that if on a given axis the distance between any two points on the grid is at least k, then there will be no overlap with respect to that axis, since the area of each item is less than k and the points of the grid are central positions. Thus, contraction up to k is safe.
(6) Similarly to the previous rule, we reduce up to $\frac{k}{2}$ (if necessary) the distance between the edges of the border of R to the bottom-left and top-right points of the grid.
Finally, it remains to apply the last reduction rule:
(7) If the screen has a height or width greater than $2k^2 - k$, return *Yes*.
The safety of Rule 5 follows from the fact that all the remaining media items fit the screen and have a height and width less than k. Note that $2k$ is a safe distance between two central points on the same line as $\frac{k}{2}$ is a safe distance from a central point to the border. Thus, if the height of the screen is greater than $2k^2 - k$ then we can insert it into the screen, in a top-down manner, the remaining k media of greater area. Analogously, if the width of the screen is greater than k^2 then we can insert on the screen, in a left-right manner, the remaining k media of greater area.

After applying the reduction rules above, we either assert *Yes* or *No* to the decision problem, or we conclude that the screen has a height and width less than $2k^2 - k$ and return an instance with at most k^3 smaller media items composing a polynomial-sized kernel for the problem.

Since a problem is $\mathcal{FPT}$ if and only if it has a kernel (see [12]), the claim holds. □

Theorem 2.2 shows that if the threshold size of the desired area to be covered is small then the problem can be efficiently solved.

3. TOOLS TO SOLVE ASLOP

The ASLOP presents several difficulties in its modeling and solving. First of all, if each possible coordinate inside the screen is a candidate to be the center of a media, the non-overlapping constraints would necessarily

be non-convex constraints, which is not a good thing when working with integer programming problems [37]. In order to avoid that kind of situation, it was decided to use a discretization of the screen, creating a grid where each point would be a center candidate. Figure 3 represent this idea for a 3x3 grid.

Considering that each media item $r \in R$ has an associated width w^r and height h^r, the generated grid has its points created using Δ_w and Δ_h, which are, respectively, the smallest width and the smallest height of the candidate media items.

$$\Delta_w = \{w^r \mid w^r \leq w^{\bar{r}} \; \forall \, r, \bar{r} \in R\} \tag{3.1}$$

$$\Delta_h = \{h^r \mid h^r \leq h^{\bar{r}} \; \forall \, r, \bar{r} \in R\}. \tag{3.2}$$

This means that after centering the origin $(0,0)$ in the upper-left corner of the screen, every candidate center can be found through the expression $(h_1 \frac{\Delta_w}{2}, h_2 \frac{\Delta_h}{2})$, where $h_1 \in [1, 2\frac{m}{\Delta_w} - 1] \cap \mathbb{Z}$ and $h_2 \in [1, 2\frac{n}{\Delta_h} - 1] \cap \mathbb{Z}$. Having defined the grid and consequently a set C of possible centers, the two integer programming formulations can be defined.

The remainder of this section is organized as follow: Subsection 3.1 presents the first formulation proposed, called Distance Formulation. In Subsection 3.2, a second integer programming formulation based on the independent set problem [31] is presented. At last, Subsection 3.3 presents the soft computing techniques.

3.1. Distance formulation

Let $c.x$ and $c.y$ be defined as the x-coordinate and y-coordinate of center $c \in C$. Through a preprocessing procedure, one can create sets $R(c)$ which contains all media items r that may be centered in center c without trespassing the borders of the screen. Following the same idea, it is possible to define $C(r)$ as the set of all possible centers for media item r, for whom the border of the screen is not trespassed if r is placed in center c. In order to verify whether an overlapping happens, given $r, \bar{r} \in R$, $c \in C(r)$ and $\bar{c} \in C(\bar{r})$ whenever the following expressions hold true, items r and $\bar{r}$ cannot be placed at centers c and $\bar{c}$ simultaneously:

$$\left(r.x + w^r \geq \bar{r}.x \;\wedge\; \bar{r}.x + w^{\bar{r}} \geq r.x\right) \tag{3.3}$$

$$\left(r.y + h^r \geq \bar{r}.y \;\wedge\; \bar{r}.y + h^{\bar{r}} \geq r.y\right) \tag{3.4}$$

where $r.x$, $r.y$, $\bar{r}.x$ and $\bar{r}.y$ can be defined as:

$$r.x = c.x - \frac{w^r}{2} \qquad \bar{r}.x = \bar{c}.x - \frac{w^{\bar{r}}}{2} \tag{3.5}$$

$$r.y = c.y - \frac{h^r}{2} \qquad \bar{r}.y = \bar{c}.y - \frac{h^{\bar{r}}}{2}. \tag{3.6}$$

The relations described by equations (3.3) and (3.4) can be seen graphically in Figure 4.

To facilitate the notation lets create the function $overlapping(r, \bar{r}, c, \bar{c})$ that returns *true* if both equations (3.3) and (3.4) are true, otherwise, false. At last, let $p_r^c \in \{0,1\}, \forall r \in R, \forall c \in C(r)$, be a decision variable associated to the media item $r \in R$ and center $c \in C$, which represents whether media item r is placed in center c. Having defined all the necessary elements, the ASLOP can be formulated as an integer programming problem as follows.

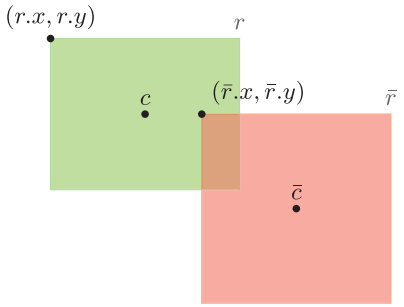


FIGURE 4. Graphical representation of two overlapping media items r and $\bar{r}$. Overlapping is calculated in both axes x and y using equations (3.3) and (3.4). Initial points $(r.x, r.y)$ and $(\bar{r}.x, \bar{r}.y)$ are obtained using equations (3.5) and (3.6).

$$\max \quad \sum_{r \in R} \sum_{c \in C} A_r p_r^c \tag{3.7}$$

$$\text{s.t.} \quad p_r^c + p_{\bar{r}}^{\bar{c}} \leq 1, \qquad \forall r \neq \bar{r} \in R, c \in C(r), \bar{c} \in C(\bar{r}) | \; overlapping(r, \bar{r}, c, \bar{c}) = true \tag{3.8}$$

$$\sum_{r \in R(c)} p_r^c \leq 1, \qquad \forall c \in C \tag{3.9}$$

$$\sum_{c \in C(r)} p_r^c \leq 1, \qquad \forall r \in R \tag{3.10}$$

$$p_r^c \in \{0, 1\}, \qquad \forall r \in R, \forall c \in C(r). \tag{3.11}$$

The objective function (3.7) states the maximization of the sum of the chosen media items' area. In this function, A_r is the area of a rectangle. Constraints (3.8) ensure that no overlapping is allowed between chosen media items. Constraints (3.9) ensure that each center c can have assigned to it at most one media item r. Constraints (3.10) ensure that one media item r can be used at most one time in the solution. At last, Constraint (3.11) define the domain of the decision variables.

3.2. Independent set formulation

In order to use this formulation one must build a conflict graph [2] $G = (V, E)$, such that each vertex $i \in V$ is associated to a pair (r, c) and has a cost $v_i = A_r$, where A_r remains the area of a rectangle. In this graph, given two vertex $i \sim (r, c)$ and $j \sim (\bar{r}, \bar{c})$, whenever at least one of the expressions $overlapping(r, \bar{r}, c, \bar{c})$, or $r = \bar{r}$, or $c = \bar{c}$ is true, an edge (i, j) is added to E. Let $\pi \in \{0, 1\}^{|V|}$ be decision variables which represent whether a vertex is in the solution. Once this graph was build and the decision variables were defined, another integer programming formulation to represent the ASLOP can be defined as follows.

$$\max \quad \sum_{i \in V} v_i \pi_i \tag{3.12}$$

$$\text{s.t.} \quad \pi_i + \pi_j \leq 1, \qquad \forall (i, j) \in E \tag{3.13}$$

$$\pi \in \{0, 1\}^{|V|}. \tag{3.14}$$

The objective function (3.12) states the maximization of the sum of the chosen media item's area. Constraints (3.13) ensure that overlapping and multiple use of the same media item is not allowed. At last, Constraints (3.14) define the domain of the decision variables.

3.3. Soft computing techniques

In this section an Iterated Local Search [26] to solve the ASLOP is presented. This subsection is organized in such way that the following three subsection present the components of the ILS and the last one present how these components are combined in the proposed approach.

3.3.1. Constructive heuristic

Soft computing techniques are being used to deal with many layout optimization problems [16,40]. The constructive heuristic applied here uses the black-box LocalSolver framework[2], which has been used successfully in covering problems [16]. The LocalSolver framework has a hybrid approach of neighborhood search, which allows it to combine different optimization techniques in a dynamic way during the resolution process. As described in their website, this framework may combine local search techniques, constraint propagation and inference techniques, linear mixed-integer programming techniques, as well as non-linear programming techniques. Since it is a black-box, how these techniques are combined is not available to general public.

Given a valid representation of the problem, the black-box LocalSolver framework applies several techniques in order to find high quality solutions. In this paper, the Independent Set Formulation (as described in Sect. 3.2) was used as a representation of the problem in the LocalSolvers' API for C++. This was done since the Independent Set Formulation provided better results than the ones obtained when using Distance Formulation together with LocalSolver. An important remark is that, since LocalSolver is dependent of the representation used, a better representation may lead to better solutions. In the following sections, LSH is used to refer to the use of LocalSolver as a constructive heuristic.

3.3.2. Perturbation

The perturbation, defined as a component of the proposed ILS, consists in adding a new item or removing an item from the solution s. In order to decide whether the addition (fA) or the removal (fR) will be used, a random number between zero and one is generated. If this number is smaller than or equal to the threshold γ the removal happens, otherwise the addition procedure is selected. When there is no media item to be removed, removal procedure cannot be performed. So instead of removal, an addition procedure is selected. When most of the region is covered by multiple media items and no more can be added, a media item is removed executing the removal procedure. The perturbation, called *AddDrop*, is described in more details in Algorithm 1.

In Algorithm 1, the methods RemoveItem(s) and AddItem(s), receives a solution and return whether it was possible to make the movement. If it was possible, the AddItem returns the new solution, otherwise RemoveItem returns a new solution.

3.3.3. Local search

In order to improved a given feasible solution, a Local Search heuristic is applied. The local search used in the proposed ILS uses as neighborhood all solutions that have exactly one item different from the current explored solution. Mathematically speaking, the neighborhood $\mathcal{N}$ of a solution $s \in S$, where S is the solution space, can be define as:

$$\mathcal{N}(s) = \{s' \in S | \rho(s, s') = 1\} \tag{3.15}$$

where $\rho(s, s')$ represents the number of different elements between these two solutions.

Having defined that, the local search procedure may be represented by Algorithm 2.

[2]LocalSolver Framework - http://www.localsolver.com.

Algorithm 1: Swap Drop.

```
Input: s,γ
begin
    opt ← rand(0, 1);
    {fR, fA} ← true;
    if opt ≤ γ then
        fR ←RemoveItem(s);
        if fR = false then
            AddItem(s);
        end
    else
        fA ←AddItem(s);
        if fA = false then
            RemoveItem(s);
        end
    end
end
return s;
```

Algorithm 2: Local Search.

```
Input: s
begin
    s̄ ← s;
    f* ← cost(s);
    nimp ← false;
    repeat
        for i ∈ s̄ do
            for j ∉ s̄ do
                if feas(s̄, i, j) = true then
                    if (cost((s̄ \ {i}) ∪ {j}) ≥ f*) then
                        in ← j;
                        out ← i;
                        f* ← cost((s̄ \ {i}) ∪ {j});
                        nimp ← true;
                    end
                end
            end
        end
        if nimp = true then
            swap(s̄, in, out)
        end
    until nimp = false;
end
return s̄;
```

Algorithm 2 receives as a parameter a feasible solution and tries to improve it by swapping one item in the solution with another one that is not in the solution. In order to do that, it verifies all feasible solutions s' in the current neighborhood and move to the best one. In order to verify feasibility, the function $feas(\bar{s}, i, j)$ is used. The function $feas(\bar{s}, i, j)$ verify whether it is possible to exchange items i and j. After verifying feasibility, Algorithm 2 verifies whether the cost of the this solution is better than the current best one. This calculation is

done through function $cost(s) = \sum_{r \in s} A_r$, which receives a solution s as parameter and returns the covered area. At last, after completely exploring the neighborhood, the $swap(.,.,.)$ function is called, in order to move to the newly best solution found, if one was found ($nimp = \text{true}$). This procedure repeats until no further improvement can be done.

3.3.4. Iterated local search - ASLOP

The Iterated Local Search (ILS) [15, 26, 29], is a metaheuristic that repeatedly applies local search procedures to solutions obtained by perturbing previously visited local optimal solutions. The ILS presented here uses as its components the LSH presented in Section 3.3.1, the Local Search and Perturbation presented in Sections 3.3.3 and 3.3.2, respectively. The methods are applied in a straightforward way. First we run the LSH to get a feasible solution. Secondly we try to improve the quality of the previously found solution by applying the Local Search and the Perturbation. So, the algorithm is described in Algorithm 3.

Algorithm 3: ILS-ASLOP.

```
1  Input: γ,StopCriterion
2  begin
3      s ← LSH();
4      s ← LocalSearch(s);
5      s_best ← UpdateBest(s);
6      while StopCriterion = false do
7          s ← AddDrop(s, γ);
8          s ← LocalSearch(s);
9          s_best ← UpdateBest(s);
10     end
11     return s_best
12 end
```

In the proposed ILS, the initial solution is generated by the LSH method. Then, the LocalSearch function performs the local search procedure and the AddDrop performs a perturbation. The UpdateBest, is responsible to store the best solution found.

4. Computational experiments

This section presents computational experiments on the formulations presented in Sections 3.1 and 3.2, and the ILS described in Section 3.3.4. Experiments considered four common screen sizes (640×480, 800×600, 1024×768 and 1920×1080) and five sets of media items (containing 14, 20, 28, 34 and 42 media items) with different dimensions to be distributed along the screen. Thus, 20 ASL instances were created for each combination of screen size and media item quantity.

Each ASLOP instance contains 64% of small media items and 36% of big media items. Small and big media items are created considering a base size calculated as a function of the screen size and considering a grid composed by 10, 15, 20, 25 and 30 media items, respectively. One should notice that the number of media items generated for each ASL instance is approximately 40% higher than the grid size so that the formulation has to choose among the available media items.

Let bs be the base size in a given dimension (width or height), the size of a small media item in that dimension is given by $0.7 \cdot bs + random(0.6 \cdot bs)$. That means that small media items are generated with sizes that are in the proximity of the base size by a factor of 30%. Moreover, let ss be the screen size in a given dimension, the size of a big media item in that dimension is given by $bs + random(ss - bs)$. That means that

TABLE 1. Results obtained by the mathematical formulations.

Instances	Distance Formulation			Independent Set Formulation		
	BSF	Time	GAP(%)	BSF	Time	GAP(%)
640_480_1	76.597	65.010	0.000	76.597	501.260	0.000
640_480_2	72.183	3600.000	0.780	67.420	3600.000	25.174
640_480_3	84.551	3600.000	1.830	73.262	3600.000	50.930
640_480_4	71.156	3600.000	4.799	0.000	3600.000	–
640_480_5	66.374	2400.000	7.141	0.000	3600.000	–
800_600_1	76.415	39.940	0.000	76.415	117.040	0.000
800_600_2	74.543	3600.000	0.714	74.543	3600.000	12.859
800_600_3	71.631	3600.000	3.321	71.631	3600.000	46.704
800_600_4	85.117	3600.000	2.734	0.000	3600.000	–
800_600_5	82.503	3600.000	6.929	0.000	3600.000	–
1024_780_1	81.214	19.380	0.000	81.214	360.710	0.000
1024_780_2	75.523	3600.000	0.858	76.733	3600.000	18.240
1024_780_3	74.186	3600.000	3.741	73.458	3600.000	48.030
1024_780_4	74.434	2787.534	6.437	63.198	3600.000	152.120
1024_780_5	79.715	3600.000	6.156	0.000	3600.000	–
1920_1080_1	69.423	26.880	0.000	69.423	41.430	0.000
1920_1080_2	73.503	3600.000	0.646	70.235	3600.000	20.790
1920_1080_3	75.528	3600.000	3.185	74.360	3600.000	34.777
1920_1080_4	77.873	3600.000	3.850	0.000	3600.000	–
1920_1080_5	78.832	3600.000	6.490	78.832	3600.000	174.854
Average	76.065	2786.937	2.981	51.366	2931.022	41.748

big media items have its size between the base size and the screen size. All generated instances are available at https://github.com/oca-cefetrj.

This section is organized as follows. The first subsection presents and analyze the results obtained by both mathematical formulations. The second subsection presents the results obtained by the proposed ILS and compare them with the results obtained by the mathematical formulation.

4.1. Mathematical formulation experiments

Both formulations were implemented in *C++* and compiled with *g++ 7.2.0* using the `-O3` optimization flag. The experiments were carried out on an Intel(R) Core i7-7740X CPU 4.30GHz machine with 32GB of RAM. For the implementation and execution of the formulations, CPLEX Studio 12.8 framework was used with its cuts, pre-processing and heuristics disabled. A one-hour processing time limit was determined and no parallel option of the processor was activated. Table 1 presents the results obtained by the mathematical formulations. The first column (Instance) presents the name of the instance. Columns (BSF - Best Solution Found) and (Time), respectively, present the coverage, in percentage, obtained by the mathematical formulation and the CPU time in seconds spent to prove optimality (or reach the time limit). Column (GAP) present the final GAP obtained by CPLEX. The last line presents the average of columns Time and GAP. Whenever column GAP presents a dash ("-"), it means that the mathematical formulation was not able find a feasible solution.

Analyzing Table 1, one can verify that the Distance Formulation was able to find a coverage over 70% in all the instances, except by `640_480_5` and `1920_1080_1` (whose the coverage was 66.34% and 69.42%, respectively). The Independent Set Formulation, by its turn, had a worse performance: it has not found a feasible solution in six out of twenty instances. Nevertheless, the other instances achieved a coverage index over 70%. In average, the Distance Formulation achieved a cover of 76.065% of the screen, while the Independent Set Formulation was able to find a cover, in average, of just 51.366%.

TABLE 2. ILS results.

Instances	LS	BSF	AvgSol	StdSol	AvgTime	StdTime
640_480_1	68.305	74.245	74.245	0.000	206.586	12.878
640_480_2	63.296	71.182	71.120	0.124	217.099	19.888
640_480_3	71.923	84.627	84.627	0.000	211.157	17.749
640_480_4	76.074	87.442	87.442	0.000	214.698	29.826
640_480_5	72.730	82.648	82.648	0.000	262.918	151.697
800_600_1	65.041	72.268	72.268	0.000	202.954	3.665
800_600_2	70.815	74.543	74.543	0.000	203.700	5.160
800_600_3	61.602	71.631	71.631	0.000	206.262	5.727
800_600_4	79.158	85.117	85.117	0.000	211.650	20.050
800_600_5	72.602	82.503	82.503	0.000	205.152	5.203
1024_768_1	71.468	81.214	81.214	0.000	201.112	3.138
1024_768_2	69.229	81.446	81.446	0.000	209.666	16.742
1024_768_3	68.964	78.654	77.488	1.829	264.646	88.899
1024_768_4	75.660	85.678	85.678	0.000	253.712	126.868
1024_768_5	79.544	85.532	85.532	0.000	230.586	73.568
1920_1080_1	61.786	69.423	69.423	0.000	196.274	4.580
1920_1080_2	66.152	73.503	73.503	0.000	217.213	35.352
1920_1080_3	75.672	80.831	80.503	0.402	264.356	94.295
1920_1080_4	72.911	80.123	80.123	0.000	234.385	78.215
1920_1080_5	72.879	80.977	80.977	0.000	228.692	65.588
Average	70,791	79.179	79.102	0.118	222.141	42.954

Now, analyzing the average computational time spent by the methods, it is possible to verify that both methods had a similar performance stopping when the time limit was reached (3600 s). In this case, Distance Formulation took less time to solve the problem to all number of media (ranging from 1.54 to 18.61 times faster). Nevertheless, even using the whole time available, the Independent Set has not found a feasible result when adopting a larger number of media items. The Distance Formulation was, on average, 200s faster than the Independent Set Formulation. In two cases, instances `640_480_5` and `1021_780_4`, the Distance Formulation had its process stopped because the process reached the maximum RAM capacity. Nonetheless, even in this case, the Distance Formulation was able to find better coverage than the Independent Set Formulation. It is interesting to remark that the Independent Set Formulation was not able to find a single feasible solution for 6 out of 20 instances.

From a theoretical point of view, both formulations are strictly the same since one can map each variable p_r^c to one variable π_i. Constraints (3.8) are in Constraints (3.13), while Constraints (3.9) and (3.10) are stronger version of Constraints (3.13) \ Constraints (3.8). Considering that, one may verify that the relaxation of Distance Formulation is tighter than the other.

4.2. ILS experiments

The proposed ILS was implemented in *C++* and compiled with *g++ 7.2.0* using the `-O3` optimization flag. The experiments were carried out on the same machine presented in Section 4.1. After testing different values for γ, $\{0.35, 0.5, 0.75\}$, we chose $\gamma = 0.5$. As for the stopping criterion, the number of iterations was 100, after testing $\{50, 100, 150, 200, 250\}$.

Table 2 is organized in such way to show the coverage of solution of the constructive phase (LS), the coverage of the best solution found by ILS (BSF), the average coverage obtained by ILS (AvgSol), the standard deviation for the obtained solutions (StdSol), the average time spent in seconds (AvgTime) and the standard deviation for the computational time (StdTime).

It is possible to note that the coverage of the ILS ranges from 69.423% to 85.678%, achieving an average coverage of 79.102%, which is better than the average solution found by the Distance and Independent Set

TABLE 3. ILS *vs.* Formulations.

Instances	GAP DF	Time DF	GAP ISF	Time ISF
640_480_1	0.031	0.315	0.031	2.426
640_480_2	0.014	16.582	−0.056	16.582
640_480_3	−0.001	17.049	−0.155	17.049
640_480_4	−0.229	16.768	–	16.768
640_480_5	−0.245	9.128	–	13.693
800_600_1	0.054	0.197	0.054	0.577
800_600_2	0.000	17.673	0.000	17.673
800_600_3	0.000	17.454	0.000	17.454
800_600_4	0.000	17.009	–	17.009
800_600_5	0.000	17.548	–	17.548
1024_780_1	0.000	0.096	0.000	1.794
1024_780_2	−0.078	17.170	−0.061	17.170
1024_780_3	−0.060	13.603	−0.071	13.603
1024_780_4	−0.151	10.987	−0.356	14.189
1024_780_5	−0.073	15.612	–	15.612
1920_1080_1	0.000	0.137	0.000	0.211
1920_1080_2	0.000	16.574	−0.047	16.574
1920_1080_3	−0.070	13.618	−0.087	13.618
1920_1080_4	−0.029	15.359	–	15.359
1920_1080_5	−0.027	15.742	−0.027	15.742
Average	−0.043	12.431	−0.055	13.033

formulations. It is also possible to note that the standard deviation is small, which shows that the ILS seems to be a stable method.

In order to improve the analysis, Table 3 shows the results of the comparison between the ILS and the mathematical formulations. Columns GAP DF and GAP ISF show the GAP of the the average solution found by the ILS in relation to the solutions obtained by the Distance Formulation (GAP DF) and the Independent Set Formulation (GAP ISF).

The GAP was calculated using the solution obtained by one of the mathematical formulations (MFsol) and the solution of ILS (ILSsol). equation (4.1) define how the GAP was computed:

$$GAP = \frac{MFsol - ILSsol}{ILSsol} \tag{4.1}$$

In addition, the column Time DF shows the quotient between the computational time spent by Distance Formulation (Time DF) and the ILS; while the column Time ISF shows the quotient between the Independent Set Formulation (Time ISF) and ILS.

The experimental results presented in Table 3 indicates that the proposed ILS not only was able to find, in average, better solutions than the mathematical formulations, but also was, in average, 12 times faster than the Distance Formulation and 13 times faster than the Independent Set Formulation. By comparing ILS with DF, the two methods with better performance, it was possible to realize that ILS had a better coverage in eleven out of twenty cases, being tied in seven of twenty. Regarding the time, ILS is faster in sixteen out of twenty instances. In the four instances that ILS is slower, the coverage is the same in two of them and it is worse in the other two).

Another comparison can be seen in Figure 5, where the average solutions obtained by the ILS are compared with the best solutions found by the two mathematical formulations. Figure 5 is a comparison, presenting three dimensions, first the transparence that represents the best solution, the opaque color presents the average solution and the time is represented by the width of the section. This representation shows how fast the ILS

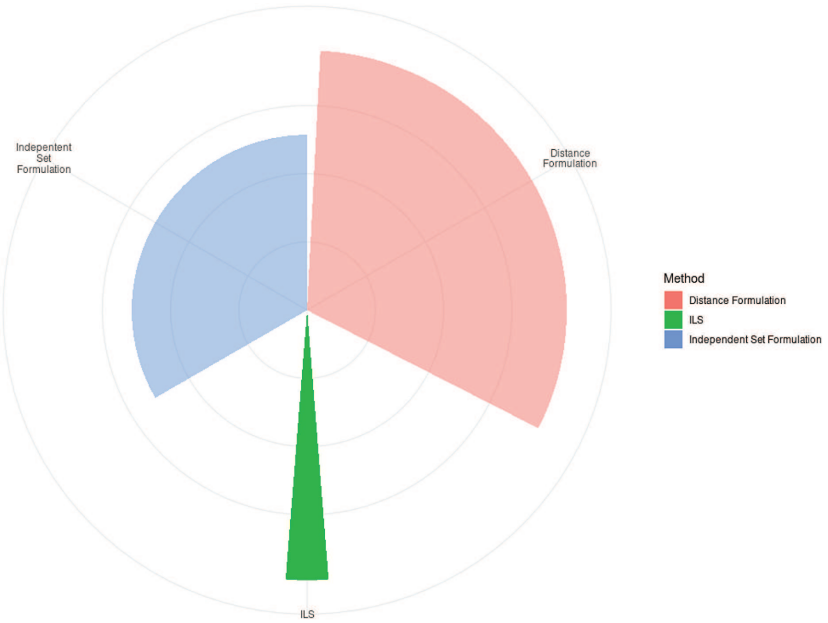


FIGURE 5. ILS *vs.* best solutions found by the mathematical formulations.

is in comparison to the formulations and how ILS is close and yet better in solution quality to the Distance Formulation.

It is worth mentioning that grids bigger than 5×5 are uncommon in digital TV applications, as well as media items smaller than 20% of the screen size. Given that the proposed solution was able to provide results for all the experimental scenarios, it points to the possible use of the ILS in real-world scenarios, *e.g.*, for defining a TV program ASL.

It is the technique used by the main tools for building multimedia applications. The technique is simple to facilitate the application author, who is usually not an experienced programmer.

In order to improve the study, we compared the ILS against a technique for positioning media objects used by the main tools for building multimedia applications [1, 27]. This technique is widely used in multimedia applications because it is simple which is called Arrangement in Order (AiO). The operation of the AiO is described as follows: the media objects that will be positioned on the display screen are declared in a text file. This file is analyzed, and each media object is evaluated in the order in which it was declared. The technique tries to position the first analyzed media object in the first available center. The centers are traversed from right to left and from top to bottom. If possible, both the media object and the center are marked as used and move on to the next media object. If it is impossible to place the media object in that center, a next available center is sought. This action is done until there are no more available centers or media objects to allocate.

The Figure 6 shows the comparison between the ILS and the AiO technique. The metric used for the comparison was the area covered by the media objects.

It is possible to notice that the ILS metaheuristic wins in all instances with significant differences. This situation was already expected since the aim of the ILS is to optimize the covered area. However, it is possible to note that the AiO technique depends a lot on how media objects are declared. The average coverage of the AiO technique is approximately 48%, while ILS has an average of 79% coverage. The average difference found between the approaches is approximately 31%, a significant value. It is noticed that the use of the ILS heuristic enhances the use of the space for presenting media objects resulting in a benefit that cannot be disregarded by content providers.

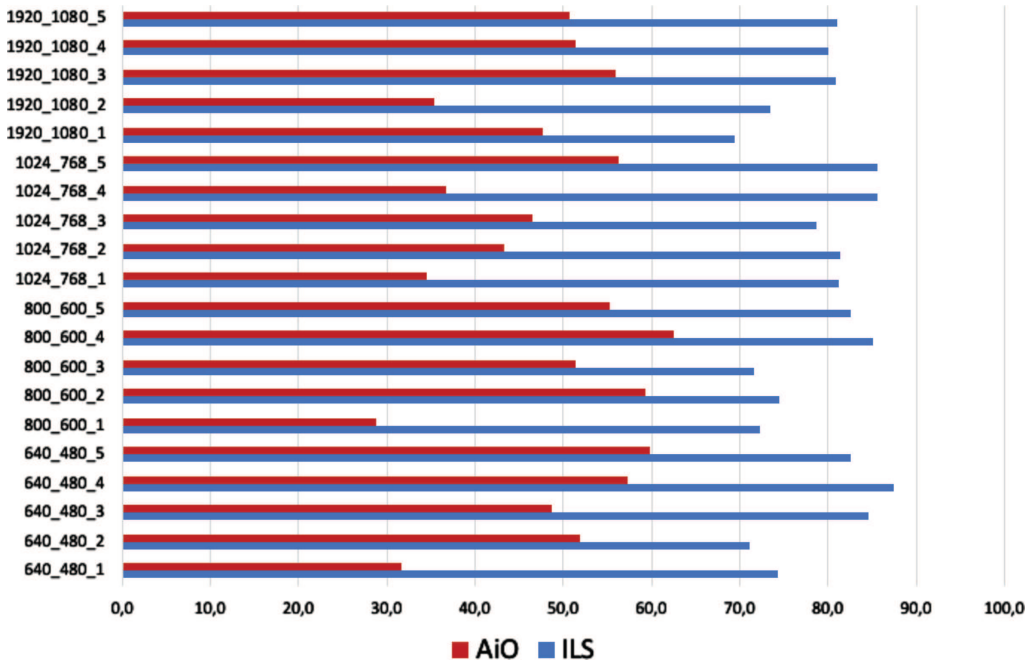


FIGURE 6. Comparison between the ILS and the AiO in the area covered. The instances are presented in the axis Y and the covered area (%) is presented in the axis X.

5. Conclusions and future works

Binding audiovisual content into a multimedia application in the production pipeline of digital TV and IPTV require authors (usually content producers) to specify for each media item its size and position when defining the application screen layout. This process requires much authoring effort since the author has to plan the ASL and consider a variety of screen sizes. Besides, it is important to maximize the area occupied on the screen, given that screen space is a valuable asset for media broadcasters. We called this problem as Application Screen Layout Optimization Problem (ASLOP). This paper proposed a new approach to deal with (ASLOP) and its applications for digital TV and IPTV. A formal definition for ASLOP and its proof of $\mathcal{NP}$-hardness was provided. A kernelization algorithm which allows us to observe that the problem is $\mathcal{FPT}$ concerning the area to be covered by the media items as a parameter is also presented. In addition, two integer programming formulations and a metaheuristic were presented to solve ASLOP: the Distance Formulation, the Independent Set Formulation and the ILS metaheuristic. All methods were tested with different screen sizes and number of media items to be distributed along the screen. The computational cost and time spent were presented for each method.

The computational experiments showed that, in practice, the Distance Formulation outperformed the Independent Set Formulation. The Distance Formulation found better solutions than the Independent Set Formulation for 12 out the 20 instances and an equal solution in 7 out the 20 instances. Now comparing the formulations with ILS, one may verify that the ILS outperformed the mathematical formulations in 10 out the 20 instances, tied in 9 out the 20 instances and lost in just one. In addition it is important to remark that, in average, the ILS was at least 12 times faster than the mathematical formulations.

In addition, it was possible to show that the ILS heuristic has a much better covered area rate than the AiO technique, normally used by multimedia applications for building layouts. ILS demonstrated an average coverage rate of 79% *versus* 48% for the AiO technique.

5.1. Future works

Future researches could focus on strengthening the formulations and developing efficient cutting-plane algorithms, so a branch and cut technique could be developed. Considering the structure of the problem, one may focus on using clique cuts to do the above proposed. It is also possible to combine the ILS with the formulations in a way to improve the starting solution generating a positive effect on branch and bound and branch and cut prunes', decreasing the computational time. In order to do combine heuristics and exact methods investigating hybrid methods that combine exact methods, meta-heuristics and data mining seems to be a suited path to pursue [14, 16, 30].

Also recalling that ASLOP generalizes classical combinatorial optimization problems like Subset Sum and Knapsack, and as shown in Theorem 2.1, the problem remains NP-hard even when both the screen and the input media have some of the dimensions equal to one. This implies that combinatorial explosions are inherent in solving the problem, even for instances that intuitively should be "easy", for example, instances with simplified shapes (*i.e.*, having both media and media items with just one dimension). To try to unravel characteristics that make instances of the problem difficult, we should go into deeper computational complexity issues. More precisely, there is a branch of Computing Theory called Parameterized Complexity Theory that aims to map the parameters that are sources of the intractability of a problem. An analysis of the Parameterized Complexity of ASLOP is also a potential future work.

5.2. Some generalizations of the problem

As one can see, the ASLOP problem is a simple version of the real problem having three structural restrictions: fitting in the screen dimension, overlap avoidance, positioning items in candidate central points of the given grid. Having said that, several variants could be proposed. The first one, instead of considering the area coverage, one must decide that it should be interesting to weigh the media and maximize the summation of the weight of the used items instead of maximizing the covered area. Note that ASLOP is a particular case of this problem where the weight of each item corresponds to its area. This variant can be explained when different enterprises have different marketing budgets and are willing to pay different amounts for their advertising. Recall that one can handle this variant using the methods proposed in this paper.

One possible second variant would deal with conflict sets of items, where items from such sets are not allowed together in the solution. This can be easily seen whenever two competing brands decide to try to advertise on the same platform. In other words, in addition to the screen R, the set of media items M, and the grid G_C, we are given sets $L_1, L_2, \dots L_\ell$ of items and asked to find a conflict-free solution S for the problem where $|S \cap L_i| \leq 1$ for each $1 \leq i \leq \ell$, representing that a solution S contains at most one item by a group of conflicting elements. To deal with this variant, both presented mathematical formulations will need one more constraint for each conflict set. These constraints will ensure that at most one item in the set is selected. From the ILS perspective, to adapt it to this variant, one should keep an auxiliary data structure to remove/add candidates whenever one item of a conflict set is selected or removed from the current solution.

Another path to pursue consists of considering multiple media with different shapes, for example, rectangular and circular media altogether, which leads to the necessity of adjustments in equations (3.3) and (3.4). Besides these adjustments, the proposed methods are also able to deal with this variant.

Another point that may be investigated as future work is the choice of the centers as well as considering that the candidate centers are not positioned according to a grid.

Acknowledgements. This work was partially supported by the National Council for Scientific and Technological Development - CNPq, under grants #307835/2017-0, #303726/2017-2 and #309832/2020-9 and by the Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ, grants #233926 and E-26/203.272/2017.

References

[1] G.F. Amorim, J.A.F. dos Santos and D.C. Muchaluat-Saade, Providing adjustable and dynamic spatial layouts for multimedia applications with style. *Multimed. Tools. Appl.* **79** (2020) 25989–26021.

[2] A. Atamtürk, G.L. Nemhauser and M.W.P. Savelsbergh, Conflict graphs in solving integer programming problems. *Eur. J. Oper. Res.* **121** (2000) 40–55.

[3] R.G.A. Azevedo, E.C. Araújo, B. Lima, L.F.G. Soares and M.F. Moreno, Composer: meeting non-functional aspects of hypermedia authoring environment. *Multimed. Tools Appl.* **70** (2014) 1199–1228.

[4] J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure. *Oper. Res.* **33** (1985) 49–64.

[5] D. Bulterman, P. Cesar and R.L. Guimaraes, Socially-aware multimedia authoring: Past, present, and future. *ACM Trans. Multimedia Comput. Commun. Appl. (TOMCCAP)* **9** (2013) 35.

[6] A. Caprara and M. Monaci, On the two-dimensional knapsack problem. *Oper. Res. Lett.* **32** (2004) 5–14.

[7] N. Clube, A liberdade de desenvolver e compartilhar conteúdo interativo (2011). http://clube.ncl.org.br/

[8] M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk and S. Saurabh, Parameterized algorithms, Vol. 4. Springer (2015).

[9] R.S. de Abreu, D. Mattos, Jd. Santos, G. Ghinea and D.C. Muchaluat-Saade, Toward content-driven intelligent authoring of mulsemedia applications. *IEEE MultiMed.* **28** (2021) 7–16.

[10] D.P. de Mattos and D.C. Muchaluat-Saade, Steve: Spatial-temporal view editor for authoring hypermedia documents. In: Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web, ACM, New York, NY, USA, Webmedia '16 (2016) 63–70. DOI: 10.1145/2976796.2976865

[11] A.M. Del Valle, T.A. de Queiroz, F.K. Miyazawa and E.C. Xavier, Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape. *Expert Syst. Appl.* **39** (2012) 12589–12598.

[12] R.G. Downey and M.R. Fellows, Fundamentals of parameterized complexity, vol 4. Springer (2013).

[13] K.A. Dowsland and W.B. Dowsland, Packing problems. *Eur. J. Oper. Res.* **56** (1992) 2–14.

[14] P.H. Gonzalez and J. Brandão, A biased random key genetic algorithm to solve the transmission expansion planning problem with re-design. In: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE (2018) 1–7.

[15] P.H. Gonzalez, L. Simonetti, P. Michelon, C. Martinhon and E. Santos, A variable fixing heuristic with local branching for the fixed charge uncapacitated network design problem with user-optimal flow. *Comput. Oper. Res.* **76** (2016) 134–146.

[16] P.H. Gonzalez, A.F.U.S. Macambira, R.V. Pinto, L. Simonetti, M. Maculan and P. Michelon, New proposals for modelling and solving the problem of covering solids using spheres of different radii. *RAIRO-Oper. Res.* **54** (2020) 873–882.

[17] HbbTV Association (2018) HbbTV 2.0.2 Specification. https://www.hbbtv.org/resource\discretionary{-}{}{}library/#specifications Accessed 20 July (2018).

[18] K. He, Y. Jin and W. Huang, Heuristics for two-dimensional strip packing problem with 90 rotations. *Expert Syst. Appl.* **40** (2013) 5542–5550.

[19] E. Huang and R.E. Korf, Optimal rectangle packing: An absolute placement approach. *J. Artif. Intell. Res.* **46** (2013) 47–87.

[20] ITU, Nested Context Language (NCL) and Ginga-NCL for IPTV services. http://www.itu.int/rec/T-REC-H.761-200904-S, iTU-T Recommendation H.761 Accessed Mar. 13, 2018 (2009).

[21] ITU, Integrated broadcast-broadband systems. https://www.itu.int/pub/R-REP-BT.2267-8-2018, Report ITU-R BT.2267-8 Accessed 15 December, 2018 (2018).

[22] R.M. Karp, Reducibility among combinatorial problems. In: Complexity of computer computations, Springer (1972) 85–103.

[23] S.C.H. Leung and D. Zhang, A fast layer-based heuristic for non-guillotine strip packing. *Expert Syst. Appl.* **38** (2011) 13032–13042.

[24] J. Li, T. Röggla, M. Glancy , J. Jansen and P. Cesar, A new production platform for authoring object-based multiscreen tv viewing experiences. In: Proceedings of the 2018 ACM International Conference on Interactive Experiences for TV and Online Video (2018) 115–126.

[25] J. Li, Z. Zheng, B. Meixner, T. Röggla, M. Glancy and P. Cesar, Designing an object-based preproduction tool for multiscreen tv viewing. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (2018) 1–6.

[26] H. Lourenco, O. Martin and T. Stutzle, Iterated local search. In "handbook of metaheuristics", edited by F. glover and G. kochenberger. isorms 57 (2002) 321–353.

[27] D.P.D. Mattos, D.C. Muchaluat-Saade and G. Ghinea, Beyond multimedia authoring: On the need for mulsemedia authoring tools. *IEEE MultiMed.* **54** (2021) 1–31.

[28] R. Meirelles, C. Júlio and Á.M. Dias, Economia e Consumo na Era da Pandemia. Tech. rep., Locomotiva - Pesquisa e Estratégia (2020). DOI: 10.1017/CBO9781107415324.004

[29] E. Santos, L.S. Ochi, L. Simonetti and P.H. González, A hybrid heuristic based on iterated local search for multivehicle inventory routing problem. *Electron. Notes Discrete Math.* **52** (2016) 197–204.

[30] G. Souto, I. Morais, G.R. Mauri, G.M. Ribeiro and P.H. González, A hybrid matheuristic for the two-stage capacitated facility location problem. *Expert Syst. Appl.* **185** (2021) 115501.

[31] R.E. Tarjan and A.E. Trojanowski, Finding a maximum independent set. *SIAM J. Comput.* **6** (1977) 537–546.

[32] W3C, Synchronized Multimedia Integration Language - SMIL 3.0 Specification. http://www.w3c.org/TR/SMIL3, world-Wide Web Consortium Recommendation Accessed Fev. 15, 2018 (2008).

[33] W3C, HTML5: A vocabulary and associated APIs for HTML and XHTML. https://www.w3.org/TR/2010/WD-html5-20100624/, world-Wide Web Consortium Recommendation Accessed Mar. 12, 2018 (2014).

[34] W3C, CSS Flexible Box Layout Module Level 1. https://www.w3.org/TR/css-flexbox-1/, w3C Candidate Recommendation Accessed Apr. 05, 2018 (2017).

[35] W3C, CSS Grid Layout Module Level 1. https://www.w3.org/TR/css-grid/, w3C Candidate Recommendation Accessed Apr. 05, 2018 (2017).

[36] Y. Wang and L. Chen, Two-dimensional residual-space-maximized packing. *Expert Syst. Appl.* **42** (2015) 3297–3305.
[37] L. Wolsey, Integer Programming. Wiley Series in Discrete Mathematics and Optimization, Wiley (1998).
[38] G. Wäscher, H. Haußner and H. Schumann, An improved typology of cutting and packing problems. *Eur. J. Oper. Res.* **183** (2007) 1109–1130.
[39] Z.Z. Zeng, X.G. Yu , K. He and Z.H. Fu, Adaptive tabu search and variable neighborhood descent for packing unequal circles into a square. *Appl. Soft Comput. J.* **65** (2018) 196–213.
[40] B. Zhang, H.F. Teng, Y.J. Shi, Layout optimization of satellite module using soft computing techniques. *Appl. Soft Comput.* **8** (2008) 507–521.

Subscribe to Open (S2O)
A fair and sustainable open access model

This journal is currently published in open access under a Subscribe-to-Open model (S2O). S2O is a transformative model that aims to move subscription journals to open access. Open access is the free, immediate, online availability of research articles combined with the rights to use these articles fully in the digital environment. We are thankful to our subscribers and sponsors for making it possible to publish this journal in open access, free of charge for authors.

Please help to maintain this journal in open access!

Check that your library subscribes to the journal, or make a personal donation to the S2O programme, by contacting subscribers@edpsciences.org

More information, including a list of sponsors and a financial transparency report, available at: https://www.edpsciences.org/en/maths-s2o-programme