Submitted to *Operations Research*
manuscript draft

New Reformulations for the Conflict Resolution Problem in the Scheduling of Television Commercials

Giovanni Giallombardo
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy, giallo@dimes.unical.it

Houyuan Jiang
Judge Business School, University of Cambridge, Trumpington Street, Cambridge CB2 1AG, UK, h.jiang@jbs.cam.ac.uk

Giovanna Miglionico
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy, gmiglionico@dimes.unical.it

We consider the *conflict-resolution* problem arising in the allocation of commercial advertisements to television program breaks. Due to the competition-avoidance requirements issued by advertisers, broadcasters aim to allocate any pairs of commercials promoting highly conflicting products to different breaks. Hence, the problem consists of assigning commercials to breaks, subject to time capacity constraints, with the aim of maximizing a total measure of the conflicts among commercials assigned to different breaks.

Since the existing reformulation can hardly be solved via exact methods, we introduce three new and efficient (mixed-)integer programming reformulations of the problem. Our computational study is based on two sets of test problems, one from the literature and another that we generate. Numerical results show the excellent performance of the proposed reformulations in terms of solution quality and computation times, when compared against an existing reformulation and an effective heuristic approach. We also provide theoretical evidences to demonstrate why some of our new reformulations should outperform the existing reformulation.

Key words: television advertising; conflict resolution problem; integer programming

1. Introduction

Selling commercial air time to clients is a critical task for television network companies, as advertising revenues represent a relevant part of their income. Several problems arise in this context, at different decisional levels, like negotiating advertising contracts, assigning commercials to breaks

in each TV program, and eventually scheduling commercials within each break. In general, the whole task can be seen as the problem of maximizing revenues for TV companies under several constraints, mostly representing the requirements issued by advertisers. Typical constraints are related to the budget available for the advertising campaign, to the desired audience demographics to be reached, to the number and length of commercials allowed per break, to the preferred positions of commercials in a break, and to possible conflicts among commercials of competing advertisers assigned to the same break.

Rather than focusing on the task as a whole, in the mathematical optimization literature the advertising problem has been studied by considering specific operational issues which arise at different times before airing each TV show. Such problems involve only a subset of the requirements described above with an appropriate objective function. For example, the problem of scheduling commercials into a specific program break aims to minimize deviations from the preferred positions of each commercial.

The focus of the paper is to study the Conflict Resolution Problem (CRP), first introduced in Gaur et al. (2009) along the guidelines of Bollapragada and Garbiras (2004). CRP arises in the allocation of commercials to TV breaks, as broadcasters have to account for competition-avoidance requirements issued by advertisers, whose aim is to have commercials promoting highly conflicting (i.e., competing) products assigned to different breaks. More formally, given a set of commercials and a set of TV program breaks, the problem consists of assigning each commercial to at most one break, so that a total measure of the conflicts among commercial pairs assigned to different breaks is maximized. As for the problem structure, common assumptions are that each program break contains three to five time slots, that the duration of every commercial is an integer multiple of a given minimum length (usually 15 seconds), and that all program breaks contain the same number of minimum-length commercials. As a consequence, the length of each slot is an integer multiple of the commercial minimum-length. Moreover, since the same commercial might need to be aired several times over the course of the show, it is useful to make a distinction between commercial

and insertion, the latter referring to a commercial assigned to a break. Hence, the problem will be formulated with respect to the set of insertions rather than the set of commercials, where the set of insertions will possibly contain multiple copies of the same commercial, and a conflict-weight is defined for each pair of insertions. Of course, insertions related to the same commercial (and ultimately to the same product) are expected to be the most conflicting, since advertisers do not want to place two (or more) advertisements of one product in the same program break.

Before going into further detail on CRP in the next section, we review three streams of literature related to the planning of TV commercial airings and the conflict resolution problem. The first stream of papers is related to scheduling commercials. In Bollapragada et al. (2002), the authors consider the problem of constructing a "sales plan" for the National Broadcasting Company. A sales plan defines, for each client, when (the week) and where (the show) each commercial of a given length will be aired during the broadcast year. The sales plan is generated trying to preserve as much as possible "premium inventory," i.e., shows that are the most preferred by clients, in order to attract further valuable advertisers. The entire plan must fulfil a set of constraints related to inventory and clients requirements, like budget availability, and the fraction of commercials to be aired on a specific show. The authors formulate the problem as an integer goal-programming problem that, due to the large dimension, cannot be solved to optimality in a reasonable amount of time. Thus, the linear relaxation of the problem is solved first, and its solution rounded, providing an initial integer solution which is later improved by a tabu search algorithm. In Bollapragada and Garbiras (2004) the problem of scheduling commercials is considered. Given that, in an earlier phase, each commercial has been already assigned to a certain program break, a goal programming problem is formulated with the aim of meeting both the product conflict constraints and the position percentage constraints (clients are ensured that given percentages of their commercials will be placed in the most preferred positions, the first and the last, of a break). The resulting integer programming formulation cannot be solved in a reasonable amount of time. Hence, the authors define a two stage heuristic where, in the first stage, they solve the product conflict problem by

swapping commercials of the same length between breaks, in order to decrease the total numbers of conflicts in the program; next, in the second stage, they solve the position percentage problem by a greedy algorithm. In Bollapragada et al. (2004) another interesting scheduling problem faced by television networks is introduced. In several circumstances, major advertisers buy several slots on a certain time horizon and only later send televisions the advertising videos to be aired. Hence, the broadcasting companies face the problem of scheduling the commercials so that the airings of the same commercial are as uniformly spread as possible. A mixed integer programming formulation of the problem is first introduced, together with a branch-and-bound algorithm for which a problem-related bounding scheme is presented. The authors also consider another formulation that can be solved efficiently for practical problems. Nevertheless, since a small increase of the size causes the problem to become computationally intractable, several heuristics are presented as well. In Brusco (2008) an improved version of the branch-and-bound of Bollapragada et al. (2004) is presented that returns optimal solutions for some of the problems previously unsolved. Moreover, a simulated-annealing method is presented that has the advantage of finding new best-known values for some of the problems considered in Bollapragada et al. (2004). In Zhang (2006) a slightly different model is considered. Indeed, the author proposes a two-step hierarchical approach to first select the advertisers and assign their commercials to shows (Winner Determination Problem), and to later schedule advertisements so that conflicts between competing commercials are avoided (Pod Assignment Problem). The Winner Determination Problem has the objective of maximizing the revenue from winning advertisers, while satisfying some constraints on demographic classes to be reached and on advertiser show preferences. This is a large scale integer programming problem that is solved by a Dantzig-Wolfe decomposition scheme which embeds an ad-hoc column generation algorithm. The Pod Assignment Problem is a decomposable quadratic integer program that is seen to be easily solvable. Finally, in Gaur et al. (2009) the above-mentioned Conflict Resolution Problem is introduced, as an extension of the commercial scheduling problem presented in Bollapragada and Garbiras (2004). In fact, while in Bollapragada and Garbiras (2004) each commercial is associated

to a 0-1 conflict-weight, i.e., each pair of commercials either does have a conflict or does not, in Gaur et al. (2009) a nonnegative conflict-weight is associated to each insertion. This choice allows programmers to avoid the presence in the same break of multiple insertions of the same commercial, and to express different degrees of conflicts between competing brands. The resulting problem, formulated as a binary program, is NP-hard. A local-search heuristic is then presented to solve the problem.

The second literature stream arises in the area of revenue management. In Bai and Xie (2006) and Kimms and Muller-Bungart (2006), an admission control and scheduling problem is considered. With a given timetable for commercial breaks, the task for the TV company is to accept or reject any incoming commercial advertisement request in order to maximize the total revenue, subject to a number of constraints including a resolution of conflicting advertisements. Exact and heuristic methods are proposed. An important problem for TV companies studied in Araman and Popescu (2010) and Bollapragada and Mallik (2008) is how to allocate the advertising capacity between upfront/forward contracts and the spot/scatter market in order to maximize profits and to meet contractual and operational constraints. The main focus in Araman and Popescu (2010) and Bollapragada and Mallik (2008) is to balance the trade-off between forward and spot markets. A prime-time TV programs allocation problem is considered in Reddy et al. (1998), where the company wants to maximize the profit, which is the difference between revenue and cost. It turns out that showing inexpensive programs with lower costs may be more profitable than showing expensive programs with higher costs.

The third literature stream is more related to the quadratic semi-assignment problem (Burkard et al. 2009). It is straightforward to show that both task allocation problems and CRP are special cases of the quadratic semi-assignment problem. An uncapacitated task allocation problem is studied in Billionnet et al. (1992) and is solved via some exact methods. A heuristic method is proposed in Hadj-Alouane et al. (1999) for solving a capacitated task allocation problem. In Ernst et al. (2006), several exact approaches including a column generation method are developed for solving

both uncapacitated and capacitated task allocation problems. It is known (Stone 1977) that when there are only two machines (or breaks in the notation of this paper), the uncapacitated task allocation problem is equivalent to a min-cut problem, which is polynomially solvable whereas the capacitated task allocation problem is NP-hard. It turns out that CRP is harder than the task allocation problem because CRP is equivalent to a max-cut problem, which is NP-hard, when there are only two breaks.

Our contribution to the CRP literature is to propose three new and efficient mathematical programming reformulations of CRP, which is formally introduced in Gaur et al. (2009), but early studied in Bollapragada and Garbiras (2004). In both Bollapragada and Garbiras (2004) and Gaur et al. (2009), heuristic methods are proposed as solution approaches. We also provide theoretical evidences to demonstrate why some of our new reformulations should outperform the existing reformulation.

The rest of this paper is organized as follows. In Section 2, the conflict resolution problem is formally defined in the language of mathematical programming. In Sections 3, we propose three (mixed-)integer linear programming reformulations for CRP based on different ways of aggregating conflict-weights between insertions. In Section 4, we provide and compare the upper bounds for the linear programming relaxations of the existing reformulation and some of our reformulations. We present computational results obtained on two sets of instances, aimed at comparing the performance of our reformulations, solved via an exact method, against an existing reformulation and an effective heuristic approach drawn from the literature. Some concluding remarks are presented in Section 5.

2. Integer Programming Formulations of CRP

Consider the conflict resolution problem introduced in Gaur et al. (2009), which is to allocate a given number of insertions to a given number of breaks. A *break* is a time window of a few minutes placed inside a TV program, or between two consecutive TV programmes, that is used for showing a sequence of advertisements. An *insertion* is a commercial advertisement with a fixed duration, that is scheduled in a break. We report in Table 1 the notation for problem data.

Table 1 Problem notation.

$\mathcal{I}$	set of all available TV insertions, with $I = \|\mathcal{I}\|$
$\mathcal{M}$	set of all program breaks, with $M = \|\mathcal{M}\|$
A_m	time capacity of each program break $m \in \mathcal{M}$
a_i	duration of each TV insertion $i \in \mathcal{I}$
f_{ij}	conflict-weight from TV insertion i to insertion j, $i, j \in \mathcal{I}$

In the remainder of the paper, for notational simplicity, we will always use indices i, j, and k to denote insertions, i.e., i, j, $k \in \mathcal{I}$, and indices m and n to denote program breaks, i.e., m, $n \in \mathcal{M}$. Note that we do not assume that $f_{ij} = f_{ji}$. This possible asymmetric property of conflict-weights reflects different views of conflicts by the owners of insertions i and j. For example, the owner of insertion i may not aim to have insertions i and j aired in different breaks, whereas the owner of insertion j, due to either a lower quality or a higher price of products for insertion j, may prefer to have i and j assigned to different breaks.

With the definitions of insertions and breaks and the notation in Table 1, the conflict resolution problem (CRP) is to assign each insertion to at most one break so that the sum of the conflict weights across all pairs of program breaks is maximized.

CRP can be formulated as the following extension of the generalized quadratic assignment problem:

$$\begin{aligned}
\max_{x} \quad & \sum_{i,j \in \mathcal{I}, i \neq j} \sum_{m,n \in \mathcal{M}, m \neq n} f_{ij} x_{im} x_{jn} \\
\text{s.t.} \quad & \sum_{m \in \mathcal{M}} x_{im} \leq 1, \ \ \forall i \in \mathcal{I} \\
& \sum_{i \in \mathcal{I}} a_i x_{im} \leq A_m, \ \ \forall m \in \mathcal{M} \\
& x_{im} \in \{0,1\}, \ \ \forall i \in \mathcal{I}, \forall m \in \mathcal{M}
\end{aligned}$$

where x_{im} is set to 1 if and only if insertion i is assigned to break m, while the main difference with the generalized quadratic assignment problem is that each insertion is not necessarily assigned to a program break.

REMARK 1. Since the following equality holds

$$\begin{aligned} & \textstyle\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}\mathbf{1}(i \text{ and } j \text{ are assigned to different breaks}) \\ = \quad & \textstyle\sum_{i,j\in\mathcal{I},i\neq j} f_{ij} - \sum_{i,j\in\mathcal{I},i\neq j} f_{ij}\mathbf{1}(i \text{ and } j \text{ are assigned to the same break}) \\ & \textstyle - \sum_{i,j\in\mathcal{I},i\neq j} f_{ij}\mathbf{1}(i \text{ or } j \text{ is not assigned to a break}), \end{aligned} \tag{1}$$

the CRP problem cannot trivially recast into the minimization of conflicts arising between insertions assigned to the same break. Only in the case where all the insertions are required to be assigned to some break the third term of the right-hand side disappears, and the minimization of intra-break conflicts becomes equivalent to the maximization of inter-break conflicts.

In Gaur et al. (2009), CRP is recast into an integer linear program, and reformulated as the following capacitated generalization of the max k-cut problem

$$\max_{x,y} \sum_{i,j\in\mathcal{I},i\neq j} \sum_{m,n\in\mathcal{M},n\neq m} f_{ij}y_{imjn} \tag{2}$$

$$\text{s.t.} \quad y_{imjn} \leq \frac{1}{2}(x_{im} + x_{jn}) \;\; \forall i \neq j \in \mathcal{I}, \forall m \neq n \in \mathcal{M} \tag{3}$$

$$\sum_{m\in\mathcal{M}} x_{im} \leq 1, \;\; \forall i \in \mathcal{I} \tag{4}$$

$$\sum_{i\in\mathcal{I}} a_i x_{im} \leq A_m, \;\; \forall m \in \mathcal{M} \tag{5}$$

$$x_{im} \in \{0,1\}, \;\; \forall i \in \mathcal{I}, \forall m \in \mathcal{M} \tag{6}$$

$$y_{imjn} \in \{0,1\}, \;\; \forall i,j \in \mathcal{I}, \forall m,n \in \mathcal{M}. \tag{7}$$

where y_{imjn} is set to 1 if and only if insertion i is assigned to break m, and insertion $j \neq i$ is assigned to break $n \neq m$.

We observe that constraint (3) restricts the conflict variable y_{imjn} to be zero unless insertions i and j are assigned to different program breaks, hence the objective function (2) calculates the total conflict-weight among the pairs that are assigned to different breaks. If insertions i and j are allocated to different breaks, then both conflict-weights f_{ij} and f_{ji} are included in the objective;

if insertions i and j are allocated to the same break, then neither f_{ij} nor f_{ji} is included in the objective; and if either insertion i or insertion j is not assigned to a break, then neither f_{ij} nor f_{ji} is included in the objective. Constraint (4) states that each insertion is assigned to at most one program break, while constraint (5) specifies the capacity restriction for each program break. We note that the assignment of an insertion to a break having enough capacity does not reduce the conflict-weight between the insertions that have already been assigned.

The inequality used in the semi-assignment constraint (4) allows the model to deal with problem instances where the total capacity is not enough to assign all the insertions. A natural question arising from this remark is whether an optimal solution of the problem might leave some insertions unassigned even though enough residual capacity is available. Of course, no feasible solution may be optimal if there exists one break $\tilde{m}$ having enough residual capacity to allocate at least one unassigned cross-conflicting insertion $\tilde{\imath}$; indeed, unless the insertion $\tilde{\imath}$ has zero conflict-weight with all the insertions assigned to breaks other than $\tilde{m}$, the assignment $(\tilde{\imath}, \tilde{m})$ would increase the total conflict-weight between the insertions that have been already assigned, as previously observed. Hence, an optimal solution cannot leave a cross-conflicting insertion unassigned if there is one break having enough residual capacity. The interesting case to explore is therefore when the residual capacity is fragmented among several breaks, as shown in the following example.

EXAMPLE 1. Let $M = 2$, with capacities $A_1 = A_2 = 3$, and $I = 5$ with lengths $a_1 = \ldots = a_4 = 1$ and $a_5 = 2$, where break capacities and insertion lengths are measured in number of time slots. The conflict-weights between insertions are $f_{13} = f_{24} = 4$, $f_{12} = f_{14} = f_{23} = f_{34} = 0$, and $f_{15} = f_{25} = f_{35} = f_{45} = 1$. We assume that $f_{ij} = f_{ji}$ for all i and j. Clearly, in an optimal solution, insertions 1 and 3 are assigned to different breaks and insertions 2 and 4 are assigned to different breaks, with a conflict value of 16 and residual capacity of one slot for both breaks, as shown in Fig. 1. Hence, on one hand, insertion 5 cannot be assigned to any regular break due to fragmentation of residual capacity; on the other hand, any feasible solution where insertion 5 is assigned cannot have a conflict value greater than 14, even though its assignment would allow allocation of the whole available capacity, as shown in Fig. 2.

Figure 1 Optimal solution for Example 1. Residual capacity cannot be used but conflicts are maximized.

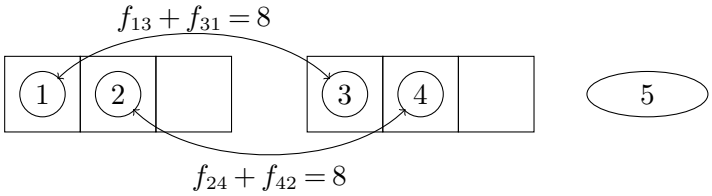
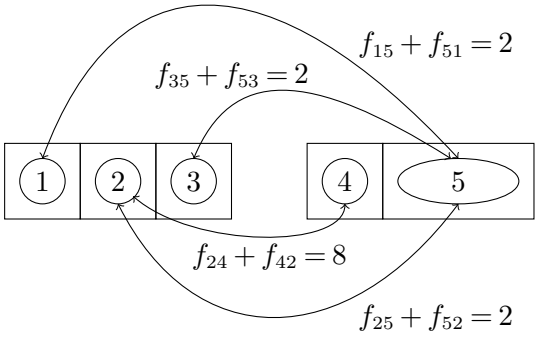


Figure 2 Feasible solution for Example 1. Every insertion is assigned but conflicts are not maximized.



In order to deal with unassigned insertions and to reformulate constraints (4)-(6), we introduce a new program break, indexed by 0, to which all unassigned insertions are allocated. We refer to such a break as a *null* break, while any scheduled break is called a *regular* break. Assume that the capacity for the null break is $A_0 = \infty$, and let $\overline{\mathcal{M}} = \mathcal{M} \cup \{0\}$.

Now, we can restate the definition of the x-variables as $x_{im} \in \{0,1\}$, $\forall i \in \mathcal{I}$, $\forall m \in \overline{\mathcal{M}}$, where x_{im} is set to 1 if and only if insertion i is assigned to break m, and $x_{i0} = 1$ has the obvious meaning that insertion i is not assigned to any regular break. We observe that any feasible assignment of insertions to break should fulfil the following constraints

$$\sum_{m \in \overline{\mathcal{M}}} x_{im} = 1, \ \forall i \in \mathcal{I}, \tag{8}$$

$$\sum_{i \in \mathcal{I}} a_i x_{im} \leq A_m, \ \forall m \in \overline{\mathcal{M}} \tag{9}$$

$$x_{im} \in \{0,1\}, \ \forall i \in \mathcal{I}, \forall m \in \overline{\mathcal{M}} \tag{10}$$

where (8) ensures that each insertion is assigned to exactly one (regular, or possibly null) break, while (9) represents the capacity constraint already introduced before. Hence, for later notational

Table 2 Insertion index subsets.

$\mathcal{J}_{im}(x) = \{j \in \mathcal{I} \mid j \neq i,\, x_{jm} = 1\}$	set of insertions other than i that are assigned to the regular break m
$\overline{\mathcal{J}}_{im}(x) = \{j \in \mathcal{I} \mid j \neq i,\, x_{jm} = x_{j0} = 0\}$	set of insertions other than i that are assigned to any regular break other than m
$\mathcal{J}_{i0}(x) = \{j \in \mathcal{I} \mid j \neq i,\, x_{j0} = 1\}$	set of insertions other than i that are assigned to the null break

convenience we define $\overline{\mathcal{X}}$ as the set of feasible insertion-to-break assignments, i.e.,

$$\overline{\mathcal{X}} = \{x : (8), (9), (10) \text{ hold}\}. \tag{11}$$

Moreover, letting $i \in \mathcal{I}$ and $m \in \mathcal{M}$ be fixed, we introduce in Table 2 three subsets of $\mathcal{I}$ dependent on any $x \in \overline{\mathcal{X}}$.

We remark that the CRP reformulation (2)-(7) contains $O(M^2 \times I^2)$ variables and $O(M^2 \times I^2)$ constraints, hence it does not look well-suited to be solved by exact methods as soon as the problem scale gets slightly larger. Therefore, heuristic methods are proposed in Gaur et al. (2009), as well as earlier in Bollapragada and Garbiras (2004), to solve the problem.

In the following section we propose efficient reformulations of CRP, trying to significantly reduce the order of magnitude of the number of variables and constraints and, more importantly, to improve the computational results obtained by adopting the CRP reformulation (2)-(7).

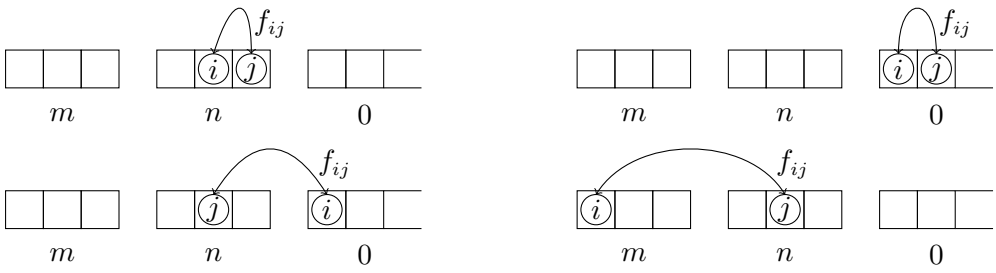
3. Integer Linear Programming Reformulations

In this section we introduce three reformulations of CRP. These reformulations are based on different ways of aggregating conflict-weights between insertions, while the introduction of the *null* break will ensure feasibility of those solutions where not all insertions are assigned to some break.

3.1. An Integer Linear Programming Reformulation Based on Inter-break Conflicts

Our first reformulation of CRP is a pure zero-one mathematical program whose structure is based on implicitly aggregating conflict-weights between insertions assigned to different breaks. This reformulation derives the total conflicts based on the right-hand side of (1). To this end, we introduce variables y_{ij} and z_{ij} for calculating the right-hand side of (1), and linear constraints that

Figure 3 First panel: $y_{ij}=0$ and $z_{ij}=1$. Second panel: $y_{ij}=1$ and $z_{ij}=0$.
Third panel: $y_{ij}=1$ and $z_{ij}=0$. Fourth panel: $y_{ij}=0$ and $z_{ij}=0$.



link variables y_{ij} and z_{ij} and assignment variables x_{im} such that y_{ij} and z_{ij} are precisely defined mathematically. In detail, we have

- $y_{ij} \in \{0,1\}$, $\forall i,j \in \mathcal{I}$, set to 1 if and only if at least one of insertions i and j is assigned to the null break;
- $z_{ij} \in \{0,1\}$, $\forall i,j \in \mathcal{I}$, set to 1 if and only if insertions i and j are both assigned to the same regular break.

A pictorial interpretation of y and z is given in Figure 3.

We then have the following two-index reformulation CRP1:

$$\max_{x,y,z} \sum_{i,j\in\mathcal{I}} f_{ij}(1-z_{ij}-y_{ij}) \tag{12}$$

$$\text{s.t. } x \in \overline{\mathcal{X}} \tag{13}$$

$$y_{ij} \le x_{i0} + x_{j0}, \ \forall i \neq j \in \mathcal{I} \tag{14}$$

$$y_{ij} \ge x_{i0}, \ \forall i \neq j \in \mathcal{I} \tag{15}$$

$$y_{ij} \ge x_{j0}, \ \forall i \neq j \in \mathcal{I} \tag{16}$$

$$z_{ij} \ge x_{im} + x_{jm} - 1, \ \forall i \neq j \in \mathcal{I}, \forall m \in \mathcal{M} \tag{17}$$

$$y_{ij}, z_{ij} \in \{0,1\}, \ \forall i,j \in \mathcal{I} \tag{18}$$

The objective function in (12) represents the total amount of conflict-weights between the insertions that are assigned to different regular breaks. In fact, it is obtained by subtracting from the

total conflict-weight, the intra-(regular)break conflicts and the null-break-related conflicts. Linking constraints (14), (15), and (16) show that $y_{ij} = 1$ if and only if at least one of insertions i and j must be assigned to the null break. Constraint (17) states that z_{ij} must be equal to one if both insertions i and j are assigned to the same regular break m. Notice that the maximization goal forces us to choose correct values for z_{ij} even though the corresponding constraints do not necessarily completely characterize the definition for z_{ij}.

3.2. A Mixed-integer Linear Programming Reformulation Based on Inter-break Conflicts

Now we present the second reformulation of CRP, a mixed-integer linear program that makes an explicit evaluation via continuous variables of inter-break conflicts. This reformulation derives the total conflicts based on the left-hand side of (1). For that purpose, we introduce continuous variables u_{im}, v_{im}, w_{im} for calculating the left-hand side of (1), and linear constraints that link variables u_{im}, v_{im} and w_{im} and assignment variables x_{im} such that u_{im}, v_{im} and w_{im} are precisely defined mathematically. In detail, we introduce for each pair insertion/regular-break (i, m) three auxiliary continuous variables u_{im}, v_{im}, and w_{im} representing different conflict-weights:

- u_{im} is the total amount of conflict-weight from insertion i to all other insertions that are assigned to regular breaks other than m, if insertion i is assigned to m, hence

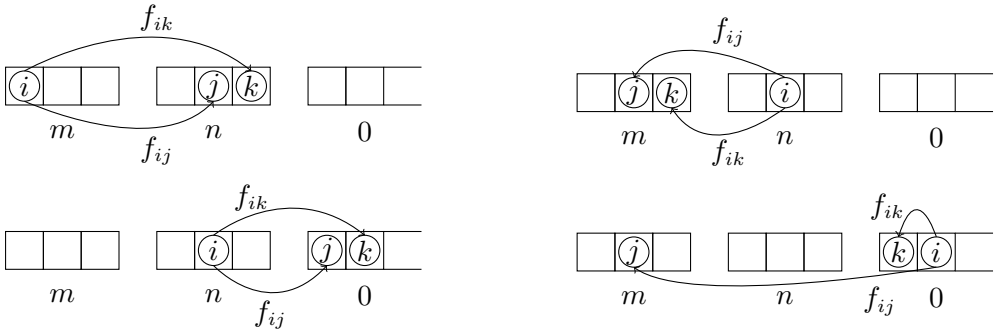
$$u_{im} = \begin{cases} \displaystyle\sum_{j \in \overline{\mathcal{J}}_{im}(x)} f_{ij} \text{ if } x_{im} = 1, \\ \\ 0 \qquad \text{otherwise;} \end{cases}$$

- v_{im} is the total amount of conflict-weight from insertion i to all other insertions that are assigned to regular break m, if insertion i is assigned to a regular break other than m, hence

$$v_{im} = \begin{cases} \displaystyle\sum_{j \in \mathcal{J}_{im}(x)} f_{ij} \text{ if } x_{im} = x_{i0} = 0, \\ \\ 0 \qquad \text{otherwise;} \end{cases}$$

- w_{im} is the total amount of conflict-weight from insertion i to the insertions that are assigned to the null break, if insertion i is assigned to a regular break other than m, or is the total amount

Figure 4 First panel: $x_{im}=1$, $u_{im}=\ldots+f_{ij}+f_{ik}+\ldots$. Second panel: $x_{im}=x_{i0}=0$, $v_{im}=\ldots+f_{ij}+f_{ik}+\ldots$. Third panel: $x_{i0}=x_{im}=0$, $w_{im}=\ldots+f_{ij}+f_{ik}+\ldots$. Fourth panel: $x_{i0}=1$, $w_{im}=\ldots+f_{ij}+f_{ik}+\ldots$.



of conflict-weight from insertion i to the insertions that are assigned to either the regular break m or the null break, if insertion i is assigned to the null break, hence

$$w_{im}=\begin{cases}\displaystyle\sum_{j\in\mathcal{J}_{i0}(x)} f_{ij} & \text{if } x_{im}=x_{i0}=0,\\ \displaystyle\sum_{j\in\mathcal{J}_{im}(x)\cup\mathcal{J}_{i0}(x)} f_{ij} & \text{if } x_{i0}=1,\\ 0 & \text{otherwise.}\end{cases}$$

We notice that u_{im} never contains the conflict-weight from insertion i to any insertion that is assigned to the null break, and that the sum of all u_{im}, or of all v_{im}, gives the total conflict-weight between the insertions that are assigned to regular breaks. Moreover, we also notice that w_{im} is mainly associated with the conflict-weights between two insertions, at least one of which is assigned to the null break. We introduce variable w in order to establish a conflict-weight balance equation.

The above definitions can be illustrated graphically as shown in Figure 4.

In Table 3, we summarize explicit formulas for u_{im}, v_{im} and w_{im} when insertion i is assigned to break m, a break other than m, and the null break, respectively.

We then have the following two-index mixed-integer linear reformulation CRP2, where δ_i is a sufficiently large scalar for each $i\in\mathcal{I}$ (e.g., $\delta_i=\sum_{j\neq i} f_{ij}$).

$$\max_{x,u,v,w} \sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} \frac{1}{2}(u_{im}+v_{im}) \tag{19}$$

Table 3 Explicit formulas for u_{im}, v_{im}, and w_{im}.

	u_{im}	v_{im}	w_{im}
$x_{im}=1$	$\sum_{j\in\overline{\mathcal{J}}_{im}(x)} f_{ij}$	0	0
$x_{im}=0,\ x_{i0}=0$	0	$\sum_{j\in\mathcal{J}_{im}(x)} f_{ij}$	$\sum_{j\in\mathcal{J}_i^0(x)} f_{ij}$
$x_{i0}=1$	0	0	$\sum_{j\in\mathcal{J}_{im}(x)\cup\mathcal{J}_{i0}(x)} f_{ij}$

$$\text{s.t.}\quad x\in\overline{\mathcal{X}} \tag{20}$$

$$u_{im}-v_{im}-w_{im}=\sum_{j\in\mathcal{I},j\neq i} f_{ij}(x_{im}-x_{jm}-x_{j0}),\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{21}$$

$$u_{im}\leq\delta_i x_{im},\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{22}$$

$$v_{im}\leq\delta_i(1-x_{im}),\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{23}$$

$$v_{im}\leq\sum_{j\neq i,j\in\mathcal{I}} f_{ij}x_{jm},\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{24}$$

$$\sum_{m\in\mathcal{M}} u_{im}\leq\delta_i(1-x_{i0}),\ \ \forall i\in\mathcal{I} \tag{25}$$

$$\sum_{m\in\mathcal{M}} v_{im}\leq\delta_i(1-x_{i0}),\ \ \forall i\in\mathcal{I} \tag{26}$$

$$w_{im}\leq\delta_i(1-x_{im}),\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{27}$$

$$u_{im},v_{im},w_{im}\geq 0,\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M}. \tag{28}$$

The objective function in (19) calculates the total conflict-weight between all the pairs of insertions that are assigned to different regular breaks. Constraint (21) is a conflict-weight balance equation which links variables u, v and w together. Constraint (22) implies that $u_{im}=0$ when insertion i is not assigned to the regular break m. Constraint (23) says that $v_{im}=0$ when insertion i is assigned to the regular break m. Constraint (24) gives an upper bound for v_{im}, which is equal to the total amount of conflict-weights between i and the insertions that are assigned to the regular break m. Constraints (25) and (26) ensure that u_{im} and v_{im} are equal to zero for all regular breaks when insertion i is assigned to the null break. Constraint (27) shows that $w_{im}=0$ when insertion i is assigned to the regular break m.

3.3. A Mixed-integer Linear Programming Reformulation Based on Intra-break Conflicts

In this section we change our reformulation viewpoint by focusing on conflicts between insertions assigned to the same regular break. In this reformulation, we derives the total conflicts based on the right-hand side of (1), which is similar to CRP1, and we employ continuous variables to represent conflict-weights, which is similar to CRP2. To this end, we introduce variables q_{im} to calculate the right-hand side of (1), and linear constraints that link variables q_{im} and assignment variables x_{im} such that q_{im} are precisely defined mathematically. For each pair insertion/regular-break (i, m) a set of auxiliary continuous variables q_{im} representing intra-break conflict-weights as follows:

- q_{im} is the total amount of conflict-weight from insertion i to all other insertions assigned either to regular break m or to the null break, if insertion i is assigned to m, hence

$$q_{im} = \begin{cases} \displaystyle\sum_{j \in \mathcal{J}_{im}(x) \cup \mathcal{J}_{i0}(x)} f_{ij} \text{ if } x_{im} = 1, \\ \\ 0 \qquad\qquad \text{otherwise.} \end{cases}$$

  The above definition can be illustrated graphically as shown in Figure 5.

  We now have the third two-index reformulation CRP3:

$$\max_{x,q} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}, j \neq i} (1 - x_{i0}) f_{ij} - \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} q_{im} \tag{29}$$

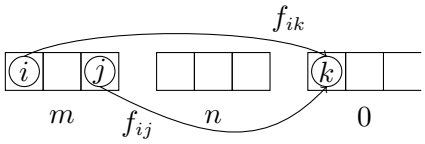
$$\text{s.t. } x \in \overline{\mathcal{X}} \tag{30}$$

$$q_{im} \geq \sum_{j \in \mathcal{I}, j \neq i} f_{ij}(x_{jm} + x_{j0} + x_{im} - 1), \ \forall i \in \mathcal{I}, \forall m \in \mathcal{M} \tag{31}$$

$$q_{im} \geq 0, \ \forall i \in \mathcal{I}, \forall m \in \mathcal{M}. \tag{32}$$

We observe that the objective function (29) still represents the total conflict-weight between all the pairs of insertions that are assigned to different regular breaks. In fact, the first term is the total conflict-weight between every insertion assigned to a regular break with all other insertions. The second term in the objective function is the total amount of conflict-weights between every insertion assigned to a regular break and all the insertions assigned either to the same regular break or the null break. Constraint (31) defines a lower bound for q_{im} when insertion i is indeed assigned to a regular break m or to the null break.

Figure 5 $x_{im} = 1$, $q_{im} = \ldots + f_{ij} + f_{ik} + \ldots$.



4. Computational Analysis

In this section we report on the computational performance of our three reformulations CRP1, CRP2, and CRP3. In particular, we focus on evaluating the behavior of our reformulations when solved via an exact (possibly truncated) method, and on comparing such performance against the subset-swapping heuristics (next referred to as GKK–H) and the CRP reformulation (2)-(7) (next referred to as GKK), both presented in Gaur et al. (2009). With this aim, we have first analyzed the experimental plan proposed in Gaur et al. (2009), which includes 1800 test problems (next referred to as TS-0) where the number of program breaks ranges from 2 to 10, and the insertion lengths belong to two classes, all equal lengths (e.g., 15 seconds) or short and long insertions (e.g., 15 and 30 seconds). There are two configurations of program breaks, depending on whether the break may allocate five short insertions, or four short insertions plus one long insertion. A test problem with M program breaks has $5M$ insertions and no excess capacity. As for the generation of conflict-weights, insertions can be assumed as partitioned into M subsets $I_1, \ldots, I_M$ (corresponding to the M breaks) of appropriate size. The conflict-weights are then set to zero for each insertion-pair belonging to the same subset. The remaining conflict-weights are instead generated by sampling from a uniform distribution in $[0, 1]$, changing the result to 100 with probability $p = 0.05$, in order to generate insertions that must appear in different program breaks. Such a procedure allows to obtain instances whose obvious optimal value is known, being the sum of all the conflict-weights. For every choice of M, and each of the two break configurations, the random generation process is repeated 100 times.

As for the execution of the swapping-subset heuristics, in Gaur et al. (2009) the authors propose to randomly shuffle the optimal insertion-to-break assignments in order to generate a starting feasible solution that is significantly different from the optimal one. Then, the computational

analysis presented in Gaur et al. (2009) refers to the best results obtained over 50 runs of GKK–H on each test instance, where each run adopts a different starting solution.

We have implemented the subset-swapping algorithm GKK–H following the guidelines given in Gaur et al. (2009). Letting t denote the number of insertions that are exchanged between pairs of breaks at each step of the algorithm, at this stage of our testing we have adopted $t = 1$, also allowing feasible swaps between a long insertion in one subset and a pair of short insertions in another subset. We wrote the code in JAVA and executed tests on an Intel Core I7 CPU at 3.50GHz with 12GB RAM. The results of our experiments confirm the excellent performance of GKK-H on the test set TS-0, as described in Gaur et al. (2009). In fact, the so-called performance ratio, i.e., the ratio between the objective value returned by the algorithm upon termination and the optimal value $\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}$, is never lower than 93% (Gaur et al. 2009, Figure 3), while the average running time across the 50 random restarts is 0.3 seconds per instance, which is much shorter than the average running time reported in Gaur et al. (2009) due to the different computing facilities adopted.

Our computational experiments involving reformulations CRP1, CRP2, CRP3, and GKK have been carried out by means of the MIP solver of IBM ILOG CPLEX 12.6 on the same machine. In the following, we briefly report on the obtained results in terms of solution quality and computation time (for simplicity of presentation we do not report the related tables). We measure the solution quality in terms of performance ratio and of the number of resolved maximal-weight conflicts (i.e., the number of insertion-pairs that must be strictly separated that are assigned to different breaks). Formulations CRP1, CRP2, and CRP3 outperform GKK-H in terms of solution quality as they can solve all the 1800 instances of TS-0 at optimality, returning 100% performance ratio (against a performance ratio of GKK-H between 93% and 100%), and obviously resolving all maximal-weight conflicts. On the contrary, reformulation GKK has poor performance compared against GKK-H since GKK can solve all the 1800 instances of TS-0 at optimality only in case $M = 2$, with an average running time of 0.02 seconds per instance, while as soon as the size increases, i.e., $M \geq 3$,

the CPLEX MIP solver can never find the optimal solution for GKK in less than 10 minutes. Formulations CRP1, CRP2, and CRP3 perform very well also in term of computation time, as the worst-case performance are obtained by adopting CRP2, that returns the optimal solution within an average running time of 0.15 seconds per instance, while the best-case performance are obtained by adopting CRP3, whose average running time is 0.02 seconds per instance.

All the above remarks have motivated us to understand the role played by the structure of TS-0 instances, as we clarify in the following proposition.

PROPOSITION 1. *Assume that an optimal assignment x^* of insertions to breaks exists for a given instance, such that*

(A1) every insertion is assigned, and

(A2) the optimal value equals the sum of all conflict-weights, i.e.,

$$\sum_{i,j\in\mathcal{I},i\neq j}\sum_{m,n\in\mathcal{M},m\neq n} f_{ij}x^*_{im}x^*_{jn} = \sum_{i,j\in\mathcal{I},i\neq j} f_{ij}.$$

Then

(a) the linear relaxation GKK-LP of problem GKK has an optimal value equal to

$$(M-1)\sum_{i,j\in\mathcal{I},i\neq j} f_{ij},$$

(b) the linear relaxations CRP1-LP and CRP3-LP of problems CRP1 and CRP3, respectively, have optimal values equal to

$$\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}.$$

Proof. The assumptions imply that the insertions can be partitioned into M subsets $I_1,\ldots,I_M$ such that conflict-weights between insertion-pairs belonging to the same subset are zero. Hence the optimal assignment x^* is such that for every $m\in\mathcal{M}$

$$x^*_{im}=1 \iff i\in I_m.$$

(a) Observe that the objective function of GKK (and GKK-LP) can be written as

$$\sum_{i,j\in\mathcal{I},i\neq j} f_{ij} \sum_{m,n\in\mathcal{M},m\neq n} y_{imjn}.$$

Summing up constraints (3) of GKK-LP we obtain for every $i,j\in\mathcal{I}$, $i\neq j$, that

$$\sum_{m,n\in\mathcal{M},m\neq n} y_{imjn} \leq \frac{1}{2} \sum_{m,n\in\mathcal{M},m\neq n} (x_{im}+x_{jn}) = \frac{1}{2}(M-1)\left(\sum_{m\in\mathcal{M}} x_{im} + \sum_{n\in\mathcal{M}} x_{jn}\right) \leq M-1$$

where the latter inequality follows from (4). As a consequence, an upper bound for the objective function of GKK-LP is given by

$$(M-1)\sum_{i,j\in\mathcal{I},i\neq j} f_{ij},$$

hence it remains to prove the existence of a feasible solution whose objective value equals such bound for instances fulfilling assumptions (A1) and (A2).

Recall that $I_1,\ldots,I_M$ are given such that $f_{ij}=0$ for every insertion-pair (i,j) for which there is an index $m\in\{1,\ldots,M\}$ with $i,j\in I_m$. As a consequence, it will suffice to focus on those insertion-pairs (i,j) such that there exist break indexes $\tilde{m}\neq\tilde{n}$, with $i\in I_{\tilde{m}}$ and $j\in I_{\tilde{n}}$, i.e., $x^*_{i\tilde{m}}=x^*_{j\tilde{n}}=1$. Let (x^*,y^*) denote the optimal solution of GKK, and observe that $y^*_{i\tilde{m}j\tilde{n}}=1$ since $x^*_{i\tilde{m}}=1$ and $x^*_{j\tilde{n}}=1$. Observe, next, that the (i,j)-term in the summation can be written as follows:

$$f_{ij}\sum_{m,n\in\mathcal{M},m\neq n} y_{imjn} = f_{ij}\left(y_{i\tilde{m}j\tilde{n}} + \sum_{(m,n)\neq(\tilde{m},\tilde{n}),m\neq n} y_{imjn} + \sum_{m\neq\tilde{m},\tilde{n}} y_{imj\tilde{n}} + \sum_{n\neq\tilde{m},\tilde{n}} y_{i\tilde{m}jn}\right).$$

Now, focusing on constraint (3) of GKK, we construct a feasible solution $(\bar{x},\bar{y})$ of GKK-LP whose value is $(M-1)\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}$. In fact, let $\bar{x}=x^*$ and, focusing for simplicity only on the insertion-pair (i,j), let the (i,j)-terms of $\bar{y}$ be such that $\bar{y}_{i\tilde{m}j\tilde{n}}=1$, and

$$\begin{cases} \bar{y}_{imj\tilde{n}} = \frac{1}{2}(\bar{x}_{im}+\bar{x}_{j\tilde{n}}) = \frac{1}{2}(0+1) = \frac{1}{2},\ \forall m\neq\tilde{m},\tilde{n} \\ \bar{y}_{i\tilde{m}jn} = \frac{1}{2}(\bar{x}_{i\tilde{m}}+\bar{x}_{jn}) = \frac{1}{2}(1+0) = \frac{1}{2},\ \forall n\neq\tilde{m},\tilde{n} \\ \bar{y}_{imjn} = \frac{1}{2}(\bar{x}_{im}+\bar{x}_{jn}) = \frac{1}{2}(0+0) = 0,\ \forall (m,n)\neq(\tilde{m},\tilde{n}),\ m\neq n. \end{cases}$$

Then, the corresponding (i,j)-term in the objective function of GKK-LP can be expressed as

$$f_{ij}\sum_{m,n\in\mathcal{M},m\neq n} \bar{y}_{imjn} = f_{ij}\left(1+0+\frac{1}{2}(M-2)+\frac{1}{2}(M-2)\right) = (M-1)f_{ij},$$

from which the thesis easily follows.

(b) We focus only on problem CRP3-LP, as a similar proof holds for CRP1-LP. We observe that the objective function (29) of CRP3-LP can be written as

$$\sum_{i,j\in\mathcal{I},j\neq i} f_{ij} - \sum_{i\in\mathcal{I}} x_{i0} \sum_{j\in\mathcal{I},j\neq i} f_{ij} - \sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{M}} q_{im}$$

hence the upper bound for its value is given by $\sum_{i,j\in\mathcal{I},j\neq i} f_{ij}$. Since the optimal assignment x^* is such that $x^*_{i0} = 0$ and $q^*_{im} = 0$, for every $i \in \mathcal{I}$ and $m \in \mathcal{M}$, x^* is also optimal for CRP3-LP, as its objective value attains the upper bound. □

Remark 2. From Proposition 1 it follows that, for instances satisfying assumptions (A1) and (A2), on one hand a nonzero optimality gap exists between GKK and GKK-LP if $M > 2$, while GKK-LP is a tight relaxation of GKK if $M = 2$. On the other hand, there is no optimality gap between CRP1/CRP3 and their linear relaxations (independent of M), while it still remains an open issue to find a related result about CRP2.

Remark 3. From the proof of Proposition 1 it can be seen that $(M-1)\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}$ and $\sum_{i,j\in\mathcal{I},j\neq i} f_{ij}$ are upper bounds for GKK and CRP1/CRP3, respectively, independent of assumptions (A1) and (A2).

We observe that all the test problems TS-0 satisfy the assumptions of Proposition 1. In view of the theoretical results of Proposition 1, then, it is not a surprise that high-level computational performance is obtained by solving CRP1 and CRP3, as every instance is actually solved at the root node of the branch-and-bound tree by simply solving its linear relaxation. Furthermore, it can be easily understood that GKK has the same performance as CRP3 whenever $M = 2$, and that performance can only worsen as soon as the number of breaks gets larger.

Summarizing, the test set TS-0 looks not challenging at all if tackled by means of our reformulations, as both the theoretical and computational results show very clearly. Moreover, a natural question arises about the extent to which the good performance of GKK–H depend on the structure of the test instances.

To address all such issues, we have prepared a slightly different test-set to gain an insight into the computational performance of our reformulations compared against the subset-swapping heuristics GKK–H. In particular, in preparing the new test problems we have got rid of assumption (A2) of Proposition 1. Moreover, in order to allow the execution of GKK–H on the new instances, we have structured the test problems such that there exist feasible solutions where every insertion is assigned, although all reformulations can be used to solve instances where not all insertions are assigned.

The main difference between the new set of instances TS-1 and TS-0 is related to the conflict-weights generation. In fact, the conflict-weights are obtained by randomly sampling every f_{ij} from a uniform distribution between 0 and 1, thus preventing the assumption (A2) of Proposition 1 from being satisfied. Similar to TS-0, f_{ij} is then changed to 100 with probability 5%, in order to represent strong conflicts arising between commercials that must be placed in different breaks. Aiming to evaluate performance on larger instances, we have pushed the instance-size of TS-1 up to $M = 20$. For simplicity of presentation we report here on results obtained by setting $M = 4, 8, 12, 16, 20$, since the computational behavior looks proportionately similar for other intermediate values of M as well. Once the number of breaks M is given, each instance is generated by considering $4M$ short insertions and M long insertions, i.e., $I = 5M$. Then, the random generation of conflict-weights involves every pair of insertions, and is repeated 20 times for each value of M, returning 100 instances partitioned into 5 groups. A summary of the main features of the new test set can be found in Table 4, where for each instance group we report the number of breaks (M), the number of insertions (I), the range of the sum of all conflict-weights ($\sum f_{ij}$), and the number of insertion pairs that have maximal conflict-weight (# max cfts). Of course, unlike TS-0, $\sum f_{ij}$ is no longer the optimal value of the problem, i.e., the optimal value of each instance is not known in advance.

Before explaining and commenting on the statistical results summarized in Tables 6 and 7, in order to make understanding of those tables easier, we introduce in Table 5 a detailed report of results obtained on a sample group (TS-1.2) of medium-size instances belonging to TS-1 by solving the four reformulations CRP1, CRP2, CRP3, and GKK, adopting a CPU time limit of 60 seconds.

Table 4 Test sets TS-1.

TS group	M	I	$\sum f_{ij}$	# max cfts
TS-1.1	4	20	[1286.74, 2782.30]	[11, 26]
TS-1.2	8	40	[6831.00, 9447.94]	[61, 87]
TS-1.3	12	60	[17988.90, 21159.60]	[163, 195]
TS-1.4	16	80	[31876.30, 37177.40]	[289, 342]
TS-1.5	20	100	[49190.30, 57004.90]	[445, 523]

In particular, the upper part of Table 5 contains in the first two columns relevant data for each of the 20 instances of TS-1.2 (cf. Table 4), i.e., the sum of all conflict-weights ($\sum f_{ij}$), and the number of insertion-pairs having maximal conflict-weight (# max cfts). The third column contains the *best* of all the four incumbent values returned by CPLEX at the expiration of time limit, for each of the four reformulations CRP1, CRP2, CRP3, and GKK. The remaining four groups of four columns summarize results returned by CPLEX at the expiration of time limit (or possibly when optimality is reached) for each reformulation: the first column contains the incumbent value f^* as a percentage of the *best* value; the second column contains the percentage optimality gap; the third column contains the computation time in seconds; the fourth column contains the percentage of resolved maximal-weight conflicts (i.e., a value of 100 means that in the final solution returned by CPLEX there are no breaks containing insertion-pairs having maximal conflict, while a value of zero would mean that no maximal-weight conflict has been solved). A statistical summary for the test group TS-1.2, in terms of minimum-, average-, and maximum-value, is then reported in the lower part of Table 5, so that the structure of the tables reported next can be easily understood.
In fact, in Table 6 we report the results for the test set TS-1 grouped in 5 rows whose structure is the same as the lower part of Table 5. Next, in Table 7, we report computational results obtained by running, without any time restriction, the GKK–H algorithm using different values for the parameter t (where the solution obtained with $t = k$ is adopted as the starting solution for the algorithm with $t = k+1$). The structure of the table is similar to Table 6, the first column containing the (minimum-, average-, maximum-) objective value f^* returned by GKK–H as a percentage of the *best* value returned by our reformulations.

Table 5 Results on a group (TS-1.2, $M = 8$ and $I = 40$) of medium-size instances of $\mathbf{TS_{GJM}}$.

TS-1.2			CRP1				CRP2				CRP3				GKK			
$\sum f_{ij}$	# max cfts	*best*	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)
9265.09	85	9207.44	99.90	0.89	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	**100.00**	0.81	60.00	**100.00**	90.64	6.77	60.00	92.94
8334.26	76	8281.23	99.93	0.87	60.00	**100.00**	99.98	0.94	60.00	**100.00**	**100.00**	0.89	60.00	**100.00**	88.80	6.93	60.00	88.16
9035.59	83	8980.29	99.88	0.87	60.00	**100.00**	99.97	0.95	60.00	**100.00**	**100.00**	0.95	60.00	**100.00**	88.61	6.95	60.00	87.95
8433.32	77	8377.35	99.92	0.87	60.00	**100.00**	**100.00**	0.95	60.00	**100.00**	99.98	0.93	60.00	**100.00**	89.00	6.92	60.00	88.31
7440.19	67	7384.70	99.90	0.88	60.00	**100.00**	99.99	0.96	60.00	**100.00**	**100.00**	0.96	60.00	**100.00**	91.52	6.71	60.00	91.04
7365.38	66	7310.17	99.89	0.86	60.00	**100.00**	99.98	0.99	60.00	**100.00**	**100.00**	0.91	60.00	**100.00**	92.70	6.61	60.00	92.42
9447.94	87	9395.67	99.88	0.87	60.00	**100.00**	99.97	0.95	60.00	**100.00**	**100.00**	0.95	60.00	**100.00**	85.91	7.19	60.00	85.06
9350.50	86	9296.79	99.91	0.83	60.00	**100.00**	99.99	0.97	60.00	**100.00**	**100.00**	0.89	60.00	**100.00**	91.13	6.73	60.00	90.70
7121.55	64	7069.52	99.91	0.85	60.00	**100.00**	99.97	0.96	60.00	**100.00**	**100.00**	0.86	60.00	**100.00**	90.62	6.78	60.00	90.63
9152.55	84	9093.54	99.95	0.86	60.00	**100.00**	**100.00**	0.96	60.00	**100.00**	**100.00**	0.91	60.00	**100.00**	90.96	6.75	60.00	90.48
7869.11	71	7811.20	99.96	0.82	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	99.99	0.91	60.00	**100.00**	85.61	7.24	60.00	84.51
8544.68	78	8489.94	99.94	0.83	60.00	**100.00**	**100.00**	0.95	60.00	**100.00**	**100.00**	0.88	60.00	**100.00**	86.85	7.11	60.00	85.90
7155.02	64	7102.73	99.83	0.87	60.00	**100.00**	99.99	0.99	60.00	**100.00**	**100.00**	0.80	60.00	**100.00**	89.70	6.86	60.00	89.06
6831.00	61	6780.50	99.89	0.87	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	99.99	0.86	60.00	**100.00**	87.32	7.08	60.00	88.52
8756.72	80	8700.84	99.87	0.89	60.00	**100.00**	99.98	0.99	60.00	**100.00**	**100.00**	0.90	60.00	**100.00**	93.51	6.53	60.00	95.00
9041.12	83	8987.66	99.89	0.87	60.00	**100.00**	99.97	0.98	60.00	**100.00**	**100.00**	0.83	60.00	**100.00**	89.36	6.72	60.00	90.36
8062.78	73	8008.61	99.87	0.87	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	99.97	0.91	60.00	**100.00**	88.51	6.96	60.00	87.67
8939.67	82	8884.75	99.93	0.86	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	99.99	0.95	60.00	**100.00**	94.83	6.43	60.00	96.34
8039.97	73	7987.77	99.92	0.81	60.00	**100.00**	99.94	0.99	60.00	**100.00**	**100.00**	0.89	60.00	**100.00**	85.89	7.20	60.00	84.93
9317.47	86	9264.29	99.91	0.86	60.00	**100.00**	99.98	0.99	60.00	**100.00**	**100.00**	0.95	60.00	**100.00**	88.95	6.91	60.00	88.37
Statistical results		min	99.83	0.81	60.00	**100.00**	99.94	0.94	60.00	**100.00**	99.97	0.80	60.00	**100.00**	85.61	6.43	60.00	84.51
		avg	99.90	0.86	60.00	**100.00**	99.99	0.97	60.00	**100.00**	**100.00**	0.90	60.00	**100.00**	89.52	6.87	60.00	89.42
		max	99.96	0.89	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	**100.00**	0.96	60.00	**100.00**	94.83	7.24	60.00	96.34

Table 6 CRP1, CRP2, CRP3, and GKK results on test set TS-1.

TS-1	M	I		CRP1				CRP2				CRP3				GKK			
				f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)	f^* (% *best*)	gap (%)	cpu (s)	cfts res (%)
TS-1.1	4	20	min	**100.00**	0.00	2.20	**100.00**	99.99	0.09	60.00	**100.00**	**100.00**	0.00	8.22	**100.00**	99.46	0.71	60.00	**100.00**
			avg	**100.00**	0.00	10.62	**100.00**	**100.00**	0.37	60.00	**100.00**	**100.00**	0.03	28.77	**100.00**	99.81	0.86	60.00	**100.00**
			max	**100.00**	0.00	38.83	**100.00**	**100.00**	0.56	60.00	**100.00**	**100.00**	0.22	60.00	**100.00**	99.93	0.99	60.00	**100.00**
TS-1.2	8	40	min	99.83	0.81	60.00	**100.00**	99.94	0.94	60.00	**100.00**	99.97	0.80	60.00	**100.00**	85.61	6.43	60.00	84.51
			avg	99.90	0.86	60.00	**100.00**	99.99	0.97	60.00	**100.00**	**100.00**	0.90	60.00	**100.00**	89.52	6.87	60.00	89.42
			max	99.96	0.89	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	**100.00**	0.96	60.00	**100.00**	94.83	7.24	60.00	96.34
TS-1.3	12	60	min	93.14	0.99	60.00	94.05	99.91	0.97	60.00	**100.00**	99.99	0.92	60.00	**100.00**	-	-	-	-
			avg	95.75	1.00	60.00	96.55	99.96	0.98	60.00	**100.00**	**100.00**	0.96	60.00	**100.00**	-	-	-	-
			max	97.95	1.00	60.00	98.86	**100.00**	0.99	60.00	**100.00**	**100.00**	0.98	60.00	**100.00**	-	-	-	-
TS-1.4	16	80	min	92.76	1.00	60.00	92.21	99.65	0.99	60.00	99.67	**100.00**	0.98	60.00	**100.00**	-	-	-	-
			avg	94.79	1.00	60.00	94.57	99.94	0.99	60.00	99.97	**100.00**	0.98	60.00	**100.00**	-	-	-	-
			max	96.84	1.00	60.00	96.72	**100.00**	0.99	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	-	-	-	-
TS-1.5	20	100	min	93.31	1.00	60.00	92.80	99.46	0.99	60.00	99.40	**100.00**	0.99	60.00	**100.00**	-	-	-	-
			avg	95.74	1.00	60.00	95.47	99.80	1.00	60.00	99.80	**100.00**	0.99	60.00	**100.00**	-	-	-	-
			max	97.42	1.00	60.00	97.30	**100.00**	1.00	60.00	**100.00**	**100.00**	0.99	60.00	**100.00**	-	-	-	-

Table 7 GKK-H results on test set TS-1, with $\mathbf{t} = \mathbf{1}, \ldots, \mathbf{4}$.

TS-1	M	I		GKK-H ($t = 1$)			GKK-H ($t = 1, 2$)			GKK-H ($t = 1, 2, 3$)			GKK-H ($t = 1, 2, 3, 4$)		
				f^* (% *best*)	cpu (s)	cfts res (%)	f^* (% *best*)	cpu (s)	cfts res (%)	f^* (% *best*)	cpu (s)	cfts res (%)	f^* (% *best*)	cpu (s)	cfts res (%)
TS-1.1	4	20	min	99.51	0.00	**100.00**	99.57	0.14	**100.00**	99.57	0.30	**100.00**	99.57	0.64	**100.00**
			avg	99.69	0.05	**100.00**	99.76	0.22	**100.00**	99.77	0.43	**100.00**	99.77	0.74	**100.00**
			max	99.83	0.11	**100.00**	99.94	0.31	**100.00**	99.94	0.55	**100.00**	99.94	0.91	**100.00**
TS-1.2	8	40	min	97.51	0.39	97.56	98.37	2.33	98.51	98.37	8.34	98.51	98.37	32.14	98.51
			avg	99.04	0.52	99.29	99.36	2.81	99.62	99.41	9.34	99.68	99.41	32.80	99.68
			max	99.74	0.70	**100.00**	99.74	3.23	**100.00**	99.77	10.28	**100.00**	99.77	33.75	**100.00**
TS-1.3	12	60	min	97.27	2.34	97.18	97.77	12.58	97.74	97.77	60.33	97.74	97.77	355.67	97.74
			avg	98.56	2.73	98.65	98.71	14.64	98.82	98.72	63.38	98.82	98.72	362.35	98.82
			max	99.35	3.02	99.49	99.35	16.02	99.49	99.35	66.64	99.49	99.35	370.17	99.49
TS-1.4	16	80	min	98.07	7.58	98.05	98.07	43.06	98.05	98.07	229.20	98.05	98.07	1928.78	98.05
			avg	98.42	8.51	98.43	98.54	49.00	98.56	98.54	256.76	98.56	98.54	1974.79	98.56
			max	99.26	9.38	99.35	99.27	55.80	99.35	99.27	300.88	99.35	99.27	2044.03	99.35
TS-1.5	20	100	min	97.98	18.03	97.90	98.15	108.39	98.09	98.15	632.09	98.09	98.15	7166.69	98.09
			avg	98.47	19.96	98.43	98.55	121.25	98.53	98.55	716.64	98.53	98.55	7431.90	98.53
			max	99.15	21.98	99.20	99.15	132.80	99.20	99.15	780.09	99.20	99.15	7784.56	99.20

As before, we analyze the results reported in Tables 6 and 7 in terms of solution quality and computation time. Focusing first on the solution quality, we observe that CRP3 shows the best performance, since it can always resolve all maximal-weight conflicts independent of the instance size, returning the *best* objective values for almost every problem. Slightly worse performance than CRP3 is returned by CRP2, that is anyway always better than GKK-H, for every value of t. In fact, GKK-H has good performance only for small-size instances, while for medium- and large-size instances performance gets worse than CRP3 and CRP2 in terms of both resolved maximal-weight conflicts and objective values. Although for small-size problems CRP1 has similar performance as GKK-H, still it behaves worse than GKK-H for medium- and large-size instances. We finally observe that only for the small-size group TS-1.1 the solution quality returned by the GKK reformulation is somehow comparable with CRP1, while for TS-1.2 results start to get significantly worse. Then, for higher-size instances GKK returns no results due to either time expiration without obtaining any feasible integer solution for TS-1.3, or even insufficient memory to run instances for TS-1.4 and TS-1.5. In summary, in terms of solution quality CRP3 has the best performance, CRP2 is slightly worse than CRP3 but better than GKK-H, which in turn outperforms CRP1, while the worst performance is obtained by GKK.

Now, focusing on the computation time, we remark that GKK-H with $t = 1$ has uniformly better performance than all reformulations over the whole test set, as it could be expected, being GKK-H a heuristic method. The results also show that it is not fruitful to run GKK-H with higher values of t. In fact, only for instances TS-1.2 the computation time is kept reasonably small, while for higher-size instances the computation time gets larger than the time limit adopted for solving the reformulations, with no significant improvement, if any, in terms of solution quality. Summarizing, with only a small additional computational time our reformulations CRP2 and CRP3 significantly outperform GKK-H in terms of solution quality.

5. Concluding remarks

The Conflict Resolution Problem arises in the allocation of commercials to TV program breaks, and originates from the competition-avoidance requirements issued by advertisers, whose aim is to

have conflicting commercials assigned to different breaks. In fact, given a set of commercials and a set of TV program breaks, the problem consists of assigning each commercial to at most one break, so that a total measure of the conflicts among commercial pairs assigned to different breaks is maximized.

We have introduced three new and efficient reformulations of CRP, adopting a reduced number of variables and constraints, based on different ways of aggregating conflict-weights between insertions. As a new modeling feature we have adopted an artificial break, whose role is to allow representation of assignment of some insertion to none of the regular breaks, in case this is necessary due to time capacity limitation of breaks, or convenient in order to better exploit available time to reduce intra-break conflicts. We have provided theoretical evidences to show why some of our reformulations should outperform one existing reformulation, and we have validated the improved efficiency via an experimental plan based on two sets of problem instances.

Possible future research on CRP involves generating effective valid inequalities in order to reduce linear programming relaxation gaps, as well as computation times necessary to get closer to optimal solutions. Other research issues are related to the study of different versions of CRP, for example the one where all insertions are required to be assigned to some regular break. Moreover, recalling that CRP is only a specific operational issue in the management of TV commercial airing, it looks relevant to study how to extend our reformulations to a more complex task like the integrated assignment and scheduling of TV commercials.

Acknowledgments

The authors are grateful to Daya Gaur for providing insights into generation of problem instances presented in Gaur et al. (2009).

References

Araman VF, Popescu I (2010) Media revenue management with audience uncertainty: Balancing upfront and spot market sales. *Manufacturing & Service Operations Management* 12(2):190–212.

Bai R, Xie J (2006) Heuristic algorithms for simultaneously accepting and scheduling advertisements on broadcast television. *Journal of Information and Computer Science* 1(4):245–251.

Billionnet A, Costa MC, Sutter A (1992) An efficient algorithm for a task allocation problem. *Journal of Association on Computing and Machinery* 39:502–518.

Bollapragada S, Cheng H, Phillips M, Garbiras M, Scholes M, Gibbs T, Humphreville M (2002) NBC's optimization systems increase revenues and productivity. *Interfaces* 32(1):47–60.

Bollapragada S, Garbiras M (2004) Sheduling commercials on broadcast television. *Operations Research* 52(3):337–345.

Bollapragada S, Bussieck MR, Mallik S (2004) Scheduling commercial videotapes in broadcast television. *Operations Research* 52(5):679–689.

Bollapragada S, Mallik S (2008) Managing on-air ad inventory in broadcast television. *IIE Transactions* 40(12):1107–1123.

Brusco MJ (2008) Scheduling advertising slots for television. *Journal of the Operational Research Society* 59:1363–1372.

Burkard R, Dell'Amico M, Martello S (2009) *Assignment Problems* (SIAM, Philadelphia).

Ernst A, Jiang H, Krishnamoorthy M (2006) Exact solutions to task allocation problems. *Management science* 52(10):1634–1646.

Gaur DR, Krishnamurti R, Kohli R (2009) Conflict resolution in the scheduling of television commercials. *Operations Research* 57(5):1098–1105.

Hadj-Alouane AB, Bean JC, Murty KG (1999) A hybrid genetic/optimization algorithm for a task allocation problem. *Journal of Scheduling* 2:189–201.

Kimms A, Muller-Bungart M (2006) Revenue management for broadcasting commercials: The channel's problem of selecting and scheduling the advertisements to be aired. *International Journal of Revenue Management* 1(1):28–44.

Reddy SK, Aronson JE, Stam A (1998) SPOT: Scheduling programs optimally for television. *Management Science* 44(1):83–102.

Stone HS (1977) Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software and Energy*, SE-3:85–93.

Zhang X (2006) Mathematical models for the television advertising allocation problem. *International Journal of the Operational Research* 1(3):302–322.