Submitted to *Operations Research*
manuscript OPRE-2015-03-179.R1

New Formulations 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 formulation can hardly be solved via exact methods, we introduce three new and efficient (mixed-)integer programming formulations of the problem. Our computational study is based on two sets of test problems, one from the literature and another more challenging data set that we generate. Numerical results show the excellent performance of the proposed formulations in terms of solution quality and computation times, when compared against an existing formulation and an effective heuristic approach. In particular, for the existing data set, all three formulations significantly outperform the existing formulation and heuristic, and moreover, for the new data set, our best formulation outperforms the heuristic on 76% of the test examples in terms of solution quality. We also provide theoretical evidence to demonstrate why some of our new formulations should outperform the existing formulation.

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

1. Introduction

The Conflict Resolution Problem (CRP), first introduced in Gaur et al. (2009) along the guidelines of Bollapragada and Garbiras (2004), 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. The same requirement somehow holds for those commercials that need to be aired several times over a given time horizon, and indeed it is generally preferred to make a distinction between a commercial and an insertion, the latter referring to an instance of a commercial assigned to a break. Hence, given a set of insertions (some insertions may represent the same commercial) and a set of TV program breaks, CRP consists of assigning each insertion to at most one break, so that a total measure of the conflicts among insertion-pairs assigned to different breaks is maximized.

We briefly review four 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. Bollapragada et al. (2002) consider the problem of constructing a "sales plan" for the National Broadcasting Company. They formulate the problem as an integer goal-programming problem, and solve it by a tabu search algorithm. Bollapragada and Garbiras (2004) study the problem of scheduling commercials and formulate it as a goal programming problem, but solved by a two-stage heuristic method. Bollapragada et al. (2004) investigate another interesting scheduling problem faced by television networks, where they aim to have the airings of the same commercial as uniformly spread as possible. They introduce a mixed integer programming formulation of the problem and a branch-and-bound algorithm. 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. The author proposes a two-step hierarchical approach involving a Dantzig-Wolfe decomposition scheme and a column generation algorithm.

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). While in Bollapragada and Garbiras (2004) each insertion-pair is associated to a 0-1 conflict-weight, i.e., each pair of insertions either does have a conflict or does not, in Gaur et al. (2009) a non-negative conflict-weight is associated to each insertion-pair, and 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 (accept/reject incoming advertisement requests to maximize revenues) and scheduling problem is considered. In Araman and Popescu (2010) and Bollapragada and Mallik (2008) the allocation of advertising capacity between upfront/forward contracts and the spot/scatter market is considered, with the aim of maximizing profits subject to contractual and operational constraints. In Reddy et al. (1998) a prime-time TV programs allocation problem is considered, where the company wants to maximize the difference between revenue and cost.

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 a branch-and-bound method based on Lagrangian relaxations. 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 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.

The fourth literature stream is about graph partitioning problems. Gaur et al. (2009) point out that CRP amounts to partitioning the nodes of a graph into k subsets of given sizes (not necessarily equal), so that the sum of edge weights across all pairs of subsets is as large as possible.

From this viewpoint, CRP can be seen as a generalization of the graph partitioning problem (see Wolkowicz and Zhao (1999)), and particularly of the capacitated max k-cut problem (see Gaur et al. (2008)). Graph partitioning problems have gained a lot of interest in the last 20 years, following the development of the semidefinite programming (SDP) relaxation paradigm and its application to enhancing approximation algorithms. Such an approach dates back to Goemans and Williamson (1995), where the SDP relaxation is introduced for the MAXCUT (i.e., max 2-cut) problem, and to Frieze and Jerrum (1997), where an extension to the max k-cut problem is proposed. Recent enhancements for the max k-cut are proposed in Anjos et al. (2013), where the SDP relaxation is tackled by a Lagrangian dual approach and solved via bundle methods. Different versions of the graph partitioning problem also exist. When the goal is to minimize the total weight of the edges joining nodes belonging to the same partition, the problem is called minimum k-partition. For such a problem the SDP relaxation is proposed in Eisenblatter (2002), and next it is embedded into a branch-and-cut approach in Ghaddar et al. (2011). The capacitated version of the minimum k-partition typically requires that each partition contains the same number of nodes, this problem being referred to as the k-way equipartition problem. Unlike such well known versions of the problem, the capacitated generalization of CRP refers to associating different weights (i.e., length) to the nodes (i.e., insertions) of the graph, and to considering subset-sizes that are expressed in terms of node-weights. For this problem, to the best of our knowledge, there is no SDP relaxation approach available in the literature. In this paper, we choose an approach based on linear programming reformulations.

Our contribution to the CRP literature is to propose three new and efficient integer linear programming formulations of CRP. Heuristic methods are proposed as solution approaches both in Bollapragada and Garbiras (2004), where the problem was first studied, and in Gaur et al. (2009), where the problem was formally stated. We provide theoretical evidence to demonstrate why some of our new formulations should outperform the one introduced in Gaur et al. (2009). Furthermore, we present a computational study to evaluate the efficiency of the new formulations,

solved via an exact method, when compared against the local-search heuristics introduced in Gaur et al. (2009). We have tested the computational behaviour of the new formulations on two sets of test examples. On the first set, obtained in the literature, we have obtained quite surprising results, as the three formulations can solve all the examples at optimality in a negligible amount of time, outperforming the available heuristics in terms of both solution quality and computation time. For a more challenging test set that we generated, computational results show that with a time budget of 60 seconds, our best formulation is never worse than the existing heuristic on all examples in terms of the objective function value, although the heuristics has uniformly better results in terms of computation time. In order to evaluate the role played by the computational time limit, we have evaluated the performance of our best formulation with different time budgets. The encouraging results show that our best formulation outperforms the heuristics on 76% of the examples in terms of the objective function value.

The rest of this paper is organized as follows. In Section 2, the conflict resolution problem is formally defined. In Section 3 we propose three (mixed-)integer linear programming formulations for CRP. In Section 4, we provide a theoretical justification for the merit of our formulations and we present computational results. 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.

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.

Table 1 Problem notation.

$\mathcal{I}$	set of all available TV insertions, indexed by i, j, and k, with $I = \|\mathcal{I}\|$
$\mathcal{M}$	set of all program breaks, indexed by m and n, 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, $i, j \in \mathcal{I}$

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. Let x_{im} denote a binary decision-variable that is set to 1 if and only if insertion i is assigned to break m. We observe that any feasible assignment of insertions to break must fulfil the following constraints

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

$$\sum_{i \in \mathcal{I}} a_i x_{im} \leq A_m, \ \forall m \in \mathcal{M} \tag{2}$$

$$x_{im} \in \{0,1\}, \ \forall i \in \mathcal{I}, \forall m \in \mathcal{M}, \tag{3}$$

where constraint (1) states that each insertion is assigned to at most one program break, while constraint (2) specifies the capacity restriction for each program break. Hence, CRP can be formulated as an extension of the generalized quadratic assignment problem:

$$\max_{x \in \mathcal{X}} \sum_{i,j \in \mathcal{I}, i \neq j} \sum_{m,n \in \mathcal{M}, m \neq n} f_{ij} x_{im} x_{jn}$$

where $\mathcal{X} \triangleq \{x : (1), (2), (3) \text{ hold}\}$, while the main difference with the generalized quadratic assignment problem is that each insertion is not necessarily assigned to a program break. We note that the objective function calculates the total conflict-weight among the pairs that are assigned to different breaks, and 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 (1) allows the model to deal with problem examples where the total capacity is not enough to assign all the insertions. Such constraint structure has some significant consequences that are highlighted in the following remark.

REMARK 1. CRP cannot be trivially recast into the minimization of conflicts arising between insertions assigned to the same break, although such an approach would seem more natural. In fact, since the following equality holds:

$$\begin{aligned} & \textstyle\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}\mathbf{1}(i \text{ and } j \text{ are assigned to different breaks}) \\ = \quad & \textstyle\sum_{i,j\in\mathcal{I},i\neq j} f_{ij} - \sum_{i,j\in\mathcal{I},i\neq j} f_{ij}\mathbf{1}(i \text{ and } j \text{ are assigned to the same break}) \\ & \textstyle - \sum_{i,j\in\mathcal{I},i\neq j} f_{ij}\mathbf{1}(i \text{ or } j \text{ is not assigned to a break}), \end{aligned} \tag{4}$$

an equivalent minimization version of CRP should also account for conflicts involving unassigned insertions. In other words, the minimization version must explicitly prevent the model from achieving the goal by simply leaving insertions unassigned.

A natural question arising at this stage is whether an optimal solution of the problem might leave some insertions unassigned even though enough residual capacity is available. It is easy to see that an optimal solution cannot leave a cross-conflicting insertion unassigned if there is one break having enough residual capacity. In fact, take a feasible solution such that one break $\tilde{m}$ has enough residual capacity to allocate at least one unassigned cross-conflicting insertion $\tilde{\imath}$; then, 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. The interesting case to explore is when the residual capacity is fragmented among several breaks. The following example shows that under particular circumstances an optimal solution may leave few insertions unassigned.

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. 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.

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

$$\max_{x,y} \sum_{i,j\in\mathcal{I},i\neq j} \sum_{m,n\in\mathcal{M},n\neq m} f_{ij} y_{imjn} \tag{5}$$

$$\text{s.t.} \quad x \in \mathcal{X} \tag{6}$$

$$y_{imjn} \leq \frac{1}{2}(x_{im} + x_{jn}) \;\; \forall i \neq j \in \mathcal{I}, \forall m \neq n \in \mathcal{M} \tag{7}$$

$$y_{imjn} \in \{0,1\}, \;\; \forall i,j \in \mathcal{I}, \forall m,n \in \mathcal{M}. \tag{8}$$

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 $n \neq m$. In fact, constraint (7) restricts the conflict variable y_{imjn} to be zero unless insertions i and j are assigned to different program breaks. We observe that CRP can also be formulated on an undirected graph, where each edge $e = (i,j)$ is assigned a conflict weight $F_e = f_{ij} + f_{ji}$, since both f_{ij} and f_{ji} contribute the objective value whenever i and j are assigned to different partitions (breaks). Nonetheless, since some of our formulations, next denoted as CRP2 and CRP3, make explicit use of the directed graph structure, we will keep adopting the structure of having both f_{ij} and f_{ji}.

We remark that the CRP formulation (5)-(8) 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 next section we introduce three new formulations of CRP, which not only reduce the order of magnitude of the number of variables and constraints, but also improve computational efficiency.

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

3. Three New (Mixed-)Integer Linear Programming Formulations of CRP

We reformulate CRP by adopting different ways of conflict-weight aggregation between insertions. All formulations share a common modeling feature, that we adopt in order to explicitly deal with unassigned insertions. In fact, 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 *regular*. Assuming that the capacity for the null break is $A_0 = \infty$, and letting $\overline{\mathcal{M}} = \mathcal{M} \cup \{0\}$, we next reformulate constraints (1)-(3). First, we update the definition of the x-variables as $x_{im} \in \{0,1\}$, $\forall i \in \mathcal{I}$, $\forall m \in \overline{\mathcal{M}}$, where $x_{i0} = 1$ has the obvious meaning that insertion i is not assigned to any regular break. Then, we define the set of feasible insertion-to-break assignments as

$$\overline{\mathcal{X}} \triangleq \left\{ x : \sum_{m \in \overline{\mathcal{M}}} x_{im} = 1 \ \forall i \in \mathcal{I},\, \sum_{i \in \mathcal{I}} a_i x_{im} \leq A_m \ \forall m \in \overline{\mathcal{M}},\, x_{im} \in \{0,1\} \ \forall i \in \mathcal{I} \ \forall m \in \overline{\mathcal{M}} \right\}. \tag{9}$$

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

3.1. An Integer Linear Programming Formulation Based on Inter-Break Conflicts

Our first formulation 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 formulation derives the total conflicts based on the right-hand side of (4). To this end, we introduce variables y_{ij} and z_{ij} for calculating the right-hand side of (4), and linear constraints that 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.

We then have the following two-index formulation CRP1:

$$\max_{x,y,z} \sum_{i,j\in\mathcal{I}} f_{ij}(1 - z_{ij} - y_{ij}) \tag{10}$$

$$\text{s.t. } x \in \overline{\mathcal{X}} \tag{11}$$

$$y_{ij} \leq x_{i0} + x_{j0}, \ \forall i \neq j \in \mathcal{I} \tag{12}$$

$$y_{ij} \geq x_{i0}, \ \forall i \neq j \in \mathcal{I} \tag{13}$$

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

$$z_{ij} \geq x_{im} + x_{jm} - 1, \ \forall i \neq j \in \mathcal{I}, \forall m \in \mathcal{M} \tag{15}$$

$$y_{ij}, z_{ij} \in \{0,1\}, \ \forall i,j \in \mathcal{I} \tag{16}$$

The objective function in (10) 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 (12), (13), and (14) 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 (15) 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 Formulation Based on Inter-Break Conflicts

Now we present the second formulation of CRP, a mixed-integer linear program that makes an explicit evaluation of inter-break conflicts by means of continuous variables. This formulation derives the total conflicts based on the left-hand side of (4). For that purpose, we introduce for

each insertion/regular-break pair (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

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

- 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

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

- 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 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

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

We notice that w_{im} is associated with conflict-weights between two insertions at least one of which is assigned to the null break. As a consequence, such variables do not contribute the objective function, their role being only limited to establish a conflict-weight balance equation.

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 formulation CRP2, where δ_i is a sufficiently large scalar for each $i \in \mathcal{I}$ (e.g., $\delta_i = \sum_{j \neq i} f_{ij}$).

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	$\sum_{j\in\mathcal{J}_{im}(x)} f_{ij}$	$\sum_{j\in\mathcal{J}_{i0}(x)} f_{ij}$
$x_{i0}=1$	0	0	$\sum_{j\in\mathcal{J}_{im}(x)\cup\mathcal{J}_{i0}(x)} f_{ij}$

$$\max_{x,u,v,w} \sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} \frac{1}{2}(u_{im}+v_{im}) \tag{17}$$

$$\text{s.t. } x\in\overline{\mathcal{X}} \tag{18}$$

$$u_{im}\le\delta_i x_{im},\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{19}$$

$$v_{im}\le\delta_i(1-x_{im}),\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{20}$$

$$w_{im}\le\delta_i(1-x_{im}),\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{21}$$

$$v_{im}\le\sum_{j\neq i,j\in\mathcal{I}} f_{ij}x_{jm},\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{22}$$

$$\sum_{m\in\mathcal{M}} u_{im}\le\delta_i(1-x_{i0}),\ \ \forall i\in\mathcal{I} \tag{23}$$

$$\sum_{m\in\mathcal{M}} v_{im}\le\delta_i(1-x_{i0}),\ \ \forall i\in\mathcal{I} \tag{24}$$

$$u_{im}-v_{im}-w_{im}=\sum_{j\in\mathcal{I},j\neq i} f_{ij}(x_{im}-x_{jm}-x_{j0}),\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M} \tag{25}$$

$$u_{im},v_{im},w_{im}\ge 0,\ \ \forall i\in\mathcal{I},\forall m\in\mathcal{M}. \tag{26}$$

The objective function in (17) calculates the total conflict-weight between all the pairs of insertions that are assigned to different regular breaks. Indeed, we observe that u_{im} and v_{im} never contain 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. Constraint (19) implies that $u_{im}=0$ when insertion i is not assigned to the regular break m. Constraint (20) and (21) force v_{im} and w_{im} to zero when insertion i is assigned to the regular break m. Constraint (22) 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 (23) and (24) 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 (25) is a conflict-weight balance equation

which links variables u, v and w together. A deeper insight into the latter constraint can be gained by comparing its structure with formulas listed in Table 3. For an example, assume that $x_{im} = 1$ for some insertion/regular-break pair (i, m), then it is easy to see from (25) that $u_{im} = \sum_{j \in \overline{\mathcal{J}}_{im}(x)} f_{ij}$, since $v_{im} = w_{im} = 0$ due to (20)-(21).

3.3. A Mixed-Integer Linear Programming Formulation Based on Intra-Break Conflicts

Our third formulation is based on conflicts between insertions assigned to the same regular break. In particular, we derive the total conflicts based on the right-hand side of (4), 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 (4), and linear constraints that link variables q_{im} and assignment variables x_{im}. For each insertion/regular-break pair (i, m) we introduce a set of auxiliary continuous variables q_{im} representing intra-break conflict-weights:

- 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

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

We now have the third two-index formulation CRP3:

$$\max_{x,q} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}, j \neq i} (1 - x_{i0}) f_{ij} - \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} q_{im} \tag{27}$$

$$\text{s.t. } x \in \overline{\mathcal{X}} \tag{28}$$

$$q_{im} \geq \sum_{j \in \mathcal{I}, j \neq i} f_{ij}(x_{jm} + x_{j0} + x_{im} - 1), \ \forall i \in \mathcal{I}, \forall m \in \mathcal{M} \tag{29}$$

$$q_{im} \geq 0, \ \forall i \in \mathcal{I}, \forall m \in \mathcal{M}. \tag{30}$$

We observe that the objective function (27) 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 (29) defines a lower bound for q_{im} when insertion i is indeed assigned to a regular break m or to the null break.

4. Computational Analysis

In this section we report on the computational performance of our three formulations CRP1, CRP2, and CRP3. In particular, we focus on evaluating the behavior of our formulations 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 formulation (5)-(8) (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 (next referred to as *strong conflict*) with probability $p = 0.05$, in order to generate insertions that must appear in different program breaks. Such a procedure allows to obtain examples 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 formulations 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 our presentation we do not report the related tables). We measure the solution quality in terms of performance ratio and of the number of resolved *strong conflicts* (i.e., the number of insertion-pairs, having $f_{ij} = 100$, that result assigned to different breaks). Formulations CRP1, CRP2, and CRP3 outperform GKK-H in terms of solution quality as they can solve all the 1800 examples of TS-0 at optimality, returning 100% performance ratio (against a performance ratio of GKK-H between 93% and 100%), and obviously resolving all *strong conflicts*. On the contrary, formulation GKK has poor performance compared against GKK-H since GKK can solve all the 1800 examples 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 examples, 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 to some regular break (i.e., no insertion is assigned to the null break), and (A2) the optimal value equals the sum of all conflict-weights (i.e., if i and j are assigned to the same break then $f_{ij} = 0$). 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 $x^*_{im} = 1$ if and only if $i \in I_m$, for every $m \in \mathcal{M}$.

(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 (7) 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 (1). As a consequence, an upper bound for the objective function of GKK-LP is given by $(M-1)\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}$. 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:

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

Now, focusing on constraint (7) 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 (27) 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 $x^*_{i0} = 0$ (see assumption (A1)) and $q^*_{im} = 0$ (see assumption(A2)), 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 $\sum_{i,j\in\mathcal{I},i\neq j} f_{ij}$ 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 formulations, 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 examples.

To address all such issues, we have prepared a slightly different test-set to gain an insight into the computational performance of our formulations 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 examples, we have structured the test problems such that there exist feasible solutions where every insertion is assigned.

The main difference between the new set of examples 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 insertions that must be placed in different breaks. Aiming to evaluate performance on larger examples, we have pushed the instance-size of TS-1 up to $M = 20$. For simplicity of our presentation we report here on the 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 examples 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 *strong conflict* (# strong 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.

Table 4 Test sets TS-1.

TS group	M	I	$\sum f_{ij}$	# strong 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]

We report in Table 5 a summary of the results for the test-set TS-1 obtained by solving the four formulations CRP1, CRP2, CRP3, and GKK, adopting a CPU time limit of 60 seconds (we keep such limit quite low aiming to avoid unfair comparison against the heuristic method). In particular, the results are grouped in 5 rows, one for each group of examples, whose structure is the following. For each formulation there are four columns containing statistical results, expressed in terms of minimum-, average-, and maximum-value, returned by CPLEX at the expiration of time limit (or possibly when optimality is reached): the first column contains the incumbent value ρ, expressed as a percentage of the best incumbent value returned by CPLEX over CRP1, CRP2, CRP3, and

Table 5 CRP1, CRP2, CRP3, and GKK results on test set TS-1.

TS-1.		CRP1				CRP2				CRP3				GKK			
		ρ (% *best*)	gap (%)	cpu (s)	cfs res (%)	ρ (% *best*)	gap (%)	cpu (s)	cfs res (%)	ρ (% *best*)	gap (%)	cpu (s)	cfs res (%)	ρ (% *best*)	gap (%)	cpu (s)	cfs res (%)
1	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
2	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
3	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	-	-	-	-
4	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	-	-	-	-
5	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	-	-	-	-

GKK; the second column contains the percentage optimality gap returned by CPLEX; the third column contains the computation time in seconds; the fourth column contains the percentage of resolved strong conflicts (i.e., a value of 100 means that in the final solution returned by CPLEX there are no breaks containing insertion-pairs with strong conflict).

Table 6 GKK-H results on test set TS-1, with $\mathbf{t = 1, \ldots, 4}$.

TS-1.		GKK-H ($t=1$)			GKK-H ($t=2$)			GKK-H ($t=3$)			GKK-H ($t=4$)		
		ρ (% *best*)	cpu (s)	cfs res (%)	ρ (% *best*)	cpu (s)	cfs res (%)	ρ (% *best*)	cpu (s)	cfs res (%)	ρ (% *best*)	cpu (s)	cfs res (%)
1	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
2	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
3	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
4	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
5	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

Next, in Table 6, we report computational results obtained by running, without any time restriction, the GKK–H algorithm using different values for the number of insertions, represented by t, that are exchanged between pairs of breaks at each step. We adopt as a starting solution for GKK-H with $t = k > 1$ the solution returned by GKK-H with $t = k - 1$. The structure of the table is similar

to Table 5, the first column containing the (minimum-, average-, maximum-) objective value ρ returned by GKK–H as a percentage of the best incumbent value returned by our formulations.

As before, we analyze the results reported in Tables 5 and 6 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 strong 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. Although for small-size problems CRP1 has similar performance as GKK-H, it behaves worse than GKK-H for medium- and large-size examples. We finally observe that the GKK formulation does not perform well compared with other formulations CRP1, CRP2, CRP3 and the heuristics GKK-H. In fact, only for the small-size group TS-1.1 the solution quality returned by the GKK formulation is somehow comparable with CRP1, while for TS-1.2 results start to get significantly worse. Then, for higher-size examples 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 examples for TS-1.4 and TS-1.5.

Summarizing, when allotted a time budget of 60 seconds, in terms of objective function value our best model (CRP3) outperforms the existing heuristic (GKK-H) on 100% of the examples by an amount ranging from 0.17% to 2.73%, while in terms of strong conflicts CRP3 resolves them all on 100% of the examples, outperforming GKK-H on 70% of the examples by an amount ranging from 0.51% to 2.82%.

Now, focusing on the computation time, we remark that GKK-H with $t = 1$ has uniformly better performance than all formulations 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 examples in TS-1.2 the computation time is reasonably small, while for higher-size examples the computation time gets larger than the time limit of 60 seconds which was used for solving CRP1, CRP2 and CRP3. Furthermore, for those examples the solution quality does not improve.

In order to evaluate the role played by the computational time limit for the performance of our formulations, we have first repeated the numerical experiments by reducing the time limit from 60 seconds down to 30 seconds and then 15 seconds, and we have obtained quite the same results in terms of solution quality (i.e., objective function value and resolved strong conflicts). Second, to further evaluate the performance of our best formulation, we have adopted as a computational limit for CRP3 a different value for each group of examples in TS1.x, that is equal to the shortest execution time of GKK-H over the 20 execution times of TS1.x. The results show that CRP3 outperforms GKK-H with $t = 1$ on 76% of the examples by an amount ranging from 0.09% to 2.72% with respect to the objective function value. Moreover, CRP3 resolves 100% of strong conflicts on 45% of the examples, while GKK-H does the same on 30% of the examples.

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 formulations 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 evidence to show why some of our formulations should outperform one existing formulation, and we have validated the improved efficiency via an experimental plan based on two sets of problem examples.

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. 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 formulations to a more complex task like the integrated assignment and scheduling of TV commercials. Another interesting research direction is to extend our formulations for solving general max-k-cut problems and to explore semidefinite programming relaxations.

Acknowledgments

The authors are grateful to Daya Gaur, for providing insights into generation of problem examples presented in Gaur et al. (2009), and to an anonymous reviewer, for carefully reading the manuscript and providing a number of helpful and constructive comments which significantly improved the presentation.

References

Anjos MF, Ghaddar B, Hupp L, Liers F, and Wiegele A (2013) Solving k-way Graph Partitioning Problems to Optimality: The Impact of Semidefinite Relaxations and the Bundle Method. In *Facets of Combinatorial Optimization*, edited by Junger M, Reinelt G, Springer, Heidelberg, 355–386.

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).

Eisenblatter A (2002) The semidefinite relaxation of the k-partition polytope is strong. In *Integer programming and combinatorial optimization,* Cook WJ and Schulz AS editors, Lecture Notes in Computer Science 2337:273–290.

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

Frieze A, Jerrum M (2006) Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica* 18:67–81.

Ghaddar B, Anjos MF, Liers F (2011) A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. *Annals of Operations Research* 188(1):155–174.

Gaur DR, Krishnamurti R, Kohli R (2008) The capacitated max k-cut problem. *Mathematical Programming* 115:65–72.

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

Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM* 42:1115–1145.

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.

Wolkowicz H, Zhao Q (1999) Semidefinite programming relaxations for the graph partitioning problem. *Discrete Applied Mathematics* 96-97:461–479.

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

Giovanni Giallombardo is an Assistant Professor of operations research at University of Calabria, Italy. His research interests include nonlinear programming, nonsmooth optimization, mathematical programs with equilibrium constraints, and optimization models for decision making in logistics.

Houyuan Jiang is University Reader (Associate Professor) in Management Science in Judge Business School of University of Cambridge. His research interests include healthcare operations management, robust optimization, operations management, revenue management, complementarity problems, mathematical programs with equilibrium constraints, hub location problems, and personnel scheduling and rostering.

Giovanna Miglionico is a Research Assistant at the Logistics Laboratory of University of Calabria, Italy. Her research interests include nonlinear and nonsmooth optimization, revenue management, and logistics.