Smart TV Scheduling in Public Venues: A New Combinatorial Problem and Optimization Approaches

Shefket Bylygbashi, Fisnik Hazrolli, Endrita Vllasaliu, Eljon Shala, Yll Berisha, Kaltrina Krasniqi, Kadri Sylejmani, Labeat Arbneshi, and Uran Lajci

Faculty of Electrical and Computer Engineering,
University of Prishtina, Prishtina, Kosovo
{shefket.bylygbashi, fisnik.hazrolli, endrita.vllasaliu, eljon.shala, yll.berisha3, kaltrina.krasniqi31}@student.uni-pr.edu
{kadri.sylejmani, labeat.arbneshi, uran.lajqi}@uni-pr.edu

Abstract. We introduce the *Smart TV Scheduling in Public Venues* problem, a new combinatorial optimization problem in which an intelligent system must select and schedule content from multiple TV channels on a single screen in a public venue to maximize aggregate viewer satisfaction, allowing gaps in the schedule, subject to operational constraints including genre diversity, minimum continuity, priority time blocks, and channel-switch and partial-broadcast penalties. We propose a hybrid pipeline combining a Genetic Algorithm (GA) seeded from the better of a Beam Search and a greedy segment-based initial solution, a Local Search refinement phase (GA+LS), and a CP-SAT exact solver warm-started from the best GA+LS solution. Experiments on 16 benchmark instances covering traditional TV broadcast, IPTV, provider-based, and YouTube scenarios demonstrate that GA+LS consistently produces high-quality incumbents that enable CP-SAT to confirm optimality on all synthetic instances and to achieve competitive scores on the larger real-life ones within a 3,600-second time limit. The results highlight that the strength of the pipeline lies in the complementarity of its components: GA+LS provides fast heuristic exploration, while CP-SAT leverages the resulting incumbent to deliver the refinement and optimality guarantees that a pure heuristic cannot offer.

Keywords: TV Scheduling · Genetic Algorithm · Local Search · Constraint Programming · Metaheuristics

1 Problem Definition

The *Smart TV Scheduling in Public Venues* problem tasks an intelligent system with selecting and displaying content from multiple available TV channels on a single screen inside a public venue. The objective is to construct a broadcast

schedule over the venue's operating hours that maximizes aggregate viewer satisfaction while navigating complex operational constraints and penalties. Gaps between scheduled programs are permitted but earn no score.

Inputs and Outputs. A problem instance specifies the venue's opening and closing times and a set of available channels, each broadcasting a fixed sequence of non-overlapping programs. Every program is characterized by a start time, an end time, a genre, and a popularity score. The system must output a valid schedule as an ordered list of tuples $(\text{channel ID}, \text{program ID}, s, e)$, where s and e denote the actual displayed start and end times of the selected program, respectively. This representation explicitly captures the broadcast sequence as well as any partial broadcasts or gaps between consecutive programs. A channel may appear multiple times, but each program is scheduled at most once. Since program times are fixed in the input, any necessary partial broadcasts are implicitly derived from the sequencing of the output list.

Operational Constraints. To be considered valid, the schedule must strictly adhere to the following rules:

- **Temporal Feasibility:** All scheduled programs must start no earlier than the opening time and end no later than the closing time.
- **Non-overlap:** Scheduled programs cannot overlap in their actual displayed timelines.
- **Genre Diversity:** The sequence cannot contain more than R consecutive programs sharing the same genre.
- **Minimum Continuity:** Every scheduled program must be broadcast continuously for at least D minutes. If a program's total duration is shorter than D, it must be scheduled for its full original duration.
- **Priority Compliance:** During designated priority time blocks, only programs from a specified subset of allowed channels may be scheduled.

Objective Function. The system aims to maximize a net score defined as:

$$\begin{aligned}\text{Total Score} = &\sum \text{program scores} + \sum \text{time-block bonuses} \\ &- S \times (\text{channel switches}) - T \times (\text{partial broadcasts})\end{aligned}$$

where S is the per-switch penalty and T is the per-partial-broadcast penalty. A time-block bonus is awarded when a scheduled program's genre matches a preferred genre interval and the program overlaps that interval by at least D minutes. The penalty T applies both when a program is terminated before its official end time and when it is joined after its official start time.

1.1 Mathematical Model

We now give a precise formulation of the problem. This same model is solved exactly by the constraint-programming (CP-SAT) baseline used in our experiments (Section 3). We state it in the native form used by the solver: variables may take integer or Boolean values, and a constraint written "if A then B" (denoted $A \Rightarrow B$) is enforced directly through reification rather than through big-M linearization.

Sets and parameters. Let P be the set of candidate programs (those whose original window overlaps the operating day), C the channels, G the genres, B the priority blocks, and W the time preferences. The day runs over $[O, E]$ in minutes. The constants D, R, S, and T are the minimum airing duration, the maximum allowed same-genre streak, the per-switch penalty, and the per-partial-broadcast penalty, respectively. Each program $i \in P$ has channel $c_i \in C$, genre $g_i \in G$, popularity score v_i, original window $[a_i, b_i)$, and original duration $\delta_i = b_i - a_i$. Each priority block $b \in B$ has a window $[\alpha_b, \beta_b)$ and an allowed-channel set $A_b \subseteq C$. Each preference $w \in W$ has a window $[\alpha_w, \beta_w)$, a target genre $\hat{g}_w$, and a bonus ρ_w; we write W_i for the preferences compatible with program i, i.e. those with $\hat{g}_w = g_i$ whose window can overlap i by at least D minutes.

Decision variables. For every program $i \in P$:

- $x_i \in \{0, 1\}$: whether program i is included in the schedule;
- $s_i, e_i \in [\max(O, a_i), \min(E, b_i)]$: the actual displayed start and end time, with chosen duration $d_i = e_i - s_i \in [0, \delta_i]$;
- $\sigma_i^{\text{late}}, \sigma_i^{\text{early}} \in \{0, 1\}$: indicators for a late join and an early termination;
- $z_{iw} \in \{0, 1\}$ for each $w \in W_i$: whether i earns preference bonus w;
- $k_i \in \{1, \dots, R\}$: the length of the same-genre streak ending at i.

For sequencing we also use arc variables $\text{succ}_{ij} \in \{0, 1\}$ over the node set $P \cup \{\text{src}, \text{snk}\}$, where $\text{succ}_{ij} = 1$ means program j airs immediately after program i.

Minimum continuity. A scheduled program must air for at least D minutes, or for its whole length if it is shorter than D; an unscheduled program has zero duration:

$$d_i \geq \min(D, \delta_i) \text{ if } x_i = 1, \qquad d_i = 0 \text{ if } x_i = 0, \qquad \forall i \in P. \tag{1}$$

Non-overlap. The displayed intervals of any two scheduled programs may not overlap. This is posted directly as a global non-overlap constraint over the optional intervals $[s_i, e_i)$ of the present programs:

$$x_i = 1 \ \wedge \ x_j = 1 \ \Rightarrow \ e_i \leq s_j \ \vee \ e_j \leq s_i, \qquad \forall\, i \neq j. \tag{2}$$

This is implied by the sequencing constraints below, but is stated explicitly to strengthen propagation.

Priority compliance. Inside a priority block, a program whose channel is not allowed must either finish before the block opens or start after it closes:

$$x_i = 1 \ \Rightarrow \ \big(e_i \leq \alpha_b\big) \ \vee \ \big(s_i \geq \beta_b\big), \qquad \forall b \in B, \ \ \forall i \in P : c_i \notin A_b. \tag{3}$$

Partial-broadcast indicators. A late-join indicator is on exactly when a present program starts after its official start, and an early-termination indicator is on exactly when a present program ends before its official end:

$$\sigma_i^{\text{late}} = 1 \;\Leftrightarrow\; \big(x_i = 1 \;\wedge\; s_i > a_i\big), \qquad \forall i \in P, \tag{4}$$

$$\sigma_i^{\text{early}} = 1 \;\Leftrightarrow\; \big(x_i = 1 \;\wedge\; e_i < b_i\big), \qquad \forall i \in P. \tag{5}$$

Each program can therefore incur up to two partial-broadcast penalties, one at each end.

Time-preference bonuses. A program earns a compatible bonus only if it is scheduled and overlaps the preference window by at least D minutes:

$$z_{iw} = 1 \;\Rightarrow\; x_i = 1 \;\wedge\; s_i \le \beta_w - D \;\wedge\; e_i \ge \alpha_w + D, \qquad \forall i \in P,\;\; \forall w \in W_i. \tag{6}$$

Sequencing. The scheduled programs must form a single ordered broadcast sequence. A directed circuit is built over $P \cup \{\text{src}, \text{snk}\}$ with the closing arc $\text{snk} \to \text{src}$ forced on; a self-loop $i \to i$ is taken exactly when i is *not* scheduled, so that the circuit visits precisely the chosen programs. Whenever j follows i, program i must finish no later than j starts:

$$\text{succ}_{ij} = 1 \;\Rightarrow\; e_i \le s_j, \qquad \forall\, i, j \in P,\; i \ne j. \tag{7}$$

Because every scheduled program has positive duration, this ordering forces start times to strictly increase along the chain and rules out subtours.

Genre diversity. A streak counter runs along the sequence: it grows by one when consecutive programs share a genre and resets to one otherwise. The first program in the sequence starts a streak of length one, and no counter may exceed R:

$$\text{succ}_{ij} = 1 \;\Rightarrow\; k_j = \begin{cases} k_i + 1, & \text{if } g_i = g_j, \\ 1, & \text{otherwise,} \end{cases} \qquad 1 \le k_j \le R. \tag{8}$$

Objective. Let a *switch* be an active arc joining two different channels. The model maximizes the total program score plus earned bonuses, minus the switch and partial-broadcast penalties:

$$\max \;\sum_{i \in P} v_i\, x_i \;+\; \sum_{i \in P} \sum_{w \in W_i} \rho_w\, z_{iw} \;-\; S \sum_{\substack{i,j \in P \\ c_i \ne c_j}} \text{succ}_{ij} \;-\; T \sum_{i \in P} \big(\sigma_i^{\text{late}} + \sigma_i^{\text{early}}\big). \tag{9}$$

This is exactly the net score of Figure 1: program scores and time-block bonuses are rewarded, while each channel switch costs S and each clipped end costs T.

Illustrative Example. Figure 1 illustrates a toy instance with 4 channels over a 90-minute window (D=10, R=2, S=1, T=2). Panel (a) shows all available programs across channels; selected programs are highlighted while unselected ones are faded. A priority block from minute 20 to 60 restricts broadcasting to

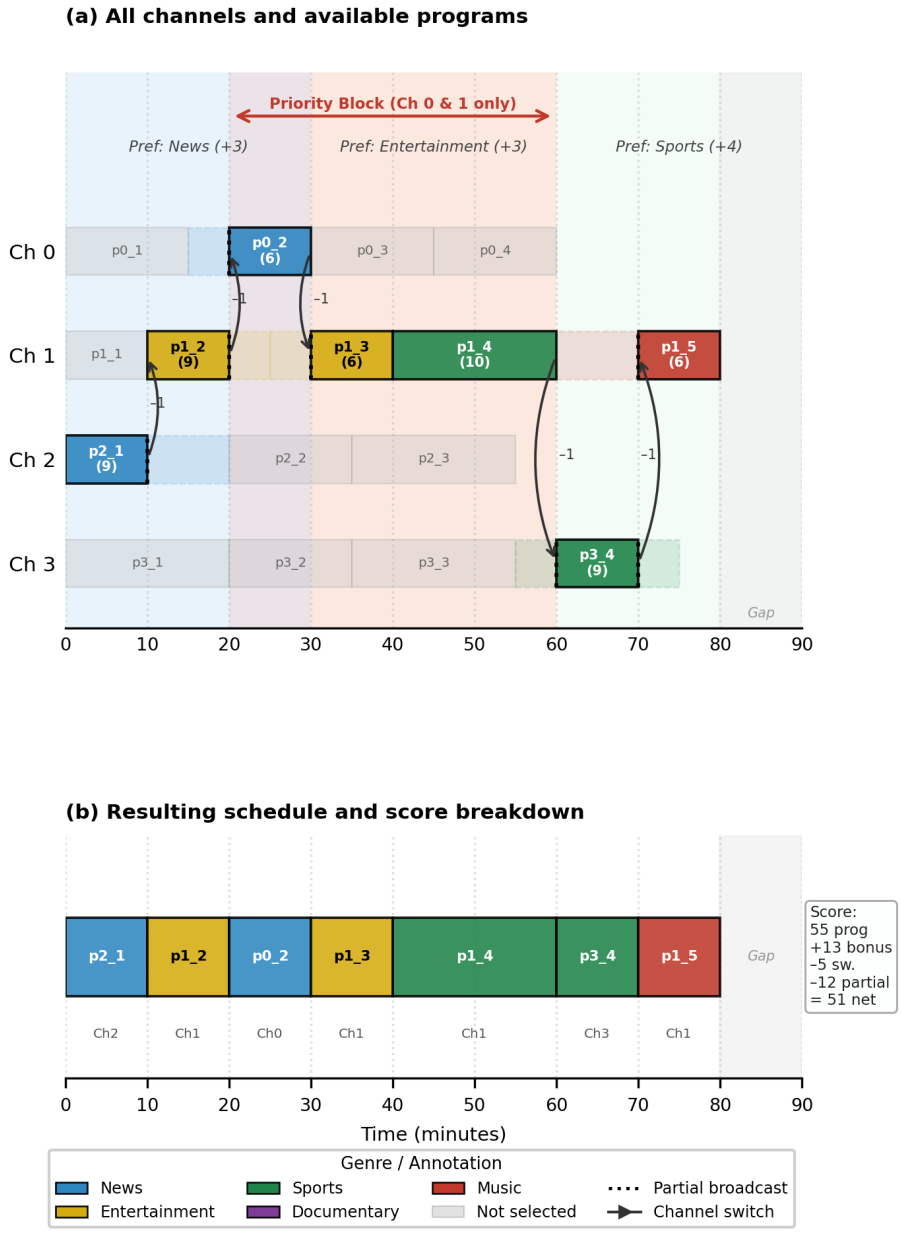

Fig. 1. Illustrative example of the *Smart TV Scheduling in Public Venues* problem on a toy instance (D=10, R=2, S=1, T=2).

Channels 0 and 1, and three time-preference intervals reward news, entertainment, and sports content in successive windows. The resulting schedule shown in panel (b) selects seven programs with a total program score of 55, earning 13 bonus points from time-preference matches. Five channel switches incur a penalty of -5 and six partial broadcasts (programs joined late or terminated early) incur a further penalty of -12, yielding a net score of 51.

2 Solution Approach

We propose a two-phase heuristic pipeline: a *Genetic Algorithm* (GA) that searches for high-quality feasible schedules, followed by a *Local Search* refinement phase (GA+LS) that improves the best solution found. The best GA+LS solution is then passed as a warm-start to the CP-SAT exact solver. Source code for all components is publicly available on GitHub (see Section 3).

Genetic Algorithm (GA). The GA begins by constructing a deterministic initial solution by running both a Beam Search and a segment-based deterministic approach [9,3] and seeding the population with the better of the two solutions.[1] A population of candidate schedules is then created as variants of this seed and evolved over multiple generations. In each generation, the population is ranked by score; the best individuals are carried forward unchanged through elitism; parents are selected via tournament selection; offspring are produced by crossover and may be further perturbed by mutation; and every offspring is repaired to restore feasibility before being evaluated.

Two crossover operators are used. *Time-cut crossover* selects a cut point in time and combines the portion before that point from one parent with the portion after it from the other, exploiting the temporal structure of the schedule. *Block-mix crossover* selects a time interval and takes a block of segments from one parent while the remainder comes from the other, preserving high-quality sub-schedules across recombination.

Five mutation operators are applied with varying disruption: *Replace segment* substitutes an existing segment with a nearby alternative from the candidate pool; *Delete weak* removes the segment with the lowest marginal contribution to the score; *Insert gap* finds an empty interval and attempts to fill it with a new program; *Expand boundary* widens a partial broadcast to a longer valid version of the same program, reducing partial-broadcast penalties; and *Block rebuild* removes a contiguous portion of the schedule and reconstructs it from scratch, enabling stronger exploration. If no improvement is observed for several generations, a more disruptive mutation is applied to escape local optima.

After crossover or mutation, a repair procedure removes invalid segments, resolves overlaps, eliminates duplicate program uses, corrects genre-streak violations, and attempts to fill resulting gaps, returning the schedule to a fully feasible

[1] Earlier stages of this project also explored a Beam Search combined with Simulated Annealing pipeline [4] as a standalone solver. Those results served as a proof of concept and are superseded by the GA+LS pipeline presented here.

state. Each individual is then scored according to the objective function: program scores and time-preference bonuses are rewarded, while channel switches and partial broadcasts are penalised. The key tuned parameters are population size, elite count, and top-K (the number of candidate segments retained per program).

Local Search (GA+LS). After the GA phase, a multi-neighbourhood greedy local search is applied to the best solution found. Rather than restarting from scratch, it explores the neighbourhood of the incumbent through several move types: replacing segments with time-adjacent alternatives, inserting programs into free slots, expanding partial broadcasts, removing low-contribution segments, rebuilding small blocks, and extending runs on the same channel to reduce switch penalties. Each candidate move is repaired and re-evaluated; only improving moves are accepted. The GA thus handles broad exploration of the solution space, while the local search provides fine-grained intensification around the most promising solution found.

3 Computational Results

All experiments were conducted on a standard desktop machine. The 16 benchmark instances span four scenario types – traditional TV broadcast (TV), internet-based (IPTV), provider-based (PW), and YouTube – covering a range of channel counts, program densities, and constraint configurations, and represent a subset of the full benchmark dataset introduced in [8].

Table 1 summarizes the 16 benchmark instances and their key structural parameters. They span more than three orders of magnitude in program count, from the compact *germany_tv* instance with 52 programs to the very large *us_iptv* instance with 115,446 programs. The synthetic TV instances use shorter horizons (900–1,080 minutes) and few time-preference intervals (5–6), whereas the real-life-based IPTV, PW, and YouTube instances run over a full day or longer and include up to 26 such intervals. Although the average number of programs per channel stays within a relatively narrow band (about 10–20), the instances differ sharply in constraint structure: the smaller instances are more tightly constrained – with priority blocks admitting only a handful of channels – while the larger instances are looser but combinatorially far more complex. Per-instance values of the constants D, R, S, and T are provided with the dataset [8].

Experimental setup. The GA was run 10 times per instance with a budget of approximately 5 minutes per run (roughly 50 minutes of total computation per instance), varying key parameters – including population size, elite count, and top-K selection – across runs; the configuration yielding the highest score was then fixed and used as the reported result. For GA+LS, the best parameter configuration identified from the GA runs was adopted and fixed; each run then executes the GA phase for approximately 60 seconds, followed immediately by 60 seconds of local search, for a total of roughly 2 minutes per run. GA+LS was also repeated 10 times per instance, with the best solution across all runs used as the warm-start for CP-SAT. Each instance was then solved with a 3,600-second

Table 1. Characteristics of the 16 benchmark instances. "Type" marks whether the instance is synthetic or derived from real-life data; "Hor." is the operating horizon in minutes; "Prog./Ch." the average programs per channel; "Gen.", "Pref.", and "Blocks" the numbers of genres, time-preference intervals, and priority blocks. The leading # matches the numbering of Table 2.

#	Instance	Type	Hor.	Chan.	Prog.	Prog./Ch.	Gen.	Pref.	Blocks
1	croatia_tv	Synth.	1,020	15	205	13.67	10	5	2
2	germany_tv	Synth.	900	5	52	10.40	15	5	2
3	kosovo_tv	Synth.	960	13	175	13.46	8	5	2
4	netherlands_tv	Synth.	1,080	12	180	15.00	10	6	2
5	uk_tv	Synth.	900	120	1,621	13.51	10	5	2
6	usa_tv	Synth.	1,080	1,100	16,277	14.80	10	5	2
7	australia_iptv	Real	1,440	499	8,151	16.33	13	26	3
8	canada_pw	Real	1,440	634	12,162	19.18	13	26	3
9	china_pw	Real	1,440	1,254	20,429	16.29	8	26	3
10	france_iptv	Real	1,440	397	6,291	15.85	13	26	2
11	singapore_pw	Real	1,440	211	4,158	19.71	13	26	2
12	spain_iptv	Real	1,440	269	4,448	16.54	13	26	2
13	uk_iptv	Real	1,440	903	12,441	13.78	13	26	2
14	us_iptv	Real	1,440	11,457	115,446	10.08	13	26	3
15	youtube_gold	Real	20,160	1,422	25,198	17.72	12	12	6
16	youtube_premium	Real	21,600	1,677	29,590	17.64	7	12	6

CP-SAT time limit. If the solver proved optimality within this limit, the result is reported as *Opt.*; if the time limit was reached before optimality could be proven but a feasible solution was found, the result is reported as *Feas.*, corresponding to the best incumbent found within the computational budget without a global optimality guarantee.

All solvers and tools developed in this work are publicly available on GitHub, including the Beam Search and greedy initial solution implementations [9], the Segment-Based Deterministic Scheduler [3], the Genetic Algorithm and Local Search pipeline [2], the CP-SAT exact solver [1], and two web-based schedule validation tools [7,5].

Table 2 reports scores for GA, GA+LS, and CP-SAT across all 16 instances. The central motivation for warm-starting CP-SAT from GA+LS is that CP-SAT, like all exact solvers, explores the solution space by progressively tightening bounds – a process that on large instances can consume the entire time budget before reaching a good feasible solution. By injecting a high-quality GA+LS solution at the start, the solver immediately has a strong lower bound to prune against, which drastically reduces the effective search space and allows CP-SAT to spend its time budget on refinement rather than on finding a first feasible solution from scratch. This is particularly important on instances such as *us_iptv* (115,446 programs) or *youtube_premium* (29,590 programs), where the combinatorial space is so large that a cold-started CP-SAT would be unlikely to find any competitive solution within the 3,600-second limit.

On the six synthetic TV instances, CP-SAT reaches optimality in all cases; in five of these six the warm-start value provided by GA+LS already equals the final CP-SAT score, meaning the solver confirms optimality without any further improvement – a direct consequence of GA+LS delivering near-optimal

Table 2. Results across 16 instances. CP-SAT is warm-started from the best GA+LS solution across 10 runs; Impr. shows the percentage gain of CP-SAT Final over the warm-start value. St.: solver status (Opt. = proven optimal within the time limit, Feas. = best incumbent within the time limit).

			GA		GA+LS		CP-SAT		
#	Instance	St.	Avg	Max	Avg	Max	Warm	Final	Impr. (%)
1	croatia_tv	Opt.	2,220.0	2,220	2,220.0	2,220	2,220	2,220	0.00
2	germany_tv	Opt.	1,604.0	1,608	1,604.0	1,608	1,608	1,608	0.00
3	kosovo_tv	Opt.	2,588.4	2,591	2,590.6	2,591	2,591	2,591	0.00
4	netherlands_tv	Opt.	2,634.6	2,635	2,633.8	2,635	2,635	2,635	0.00
5	uk_tv	Opt.	2,266.0	2,266	2,266.0	2,266	2,266	2,266	0.00
6	usa_tv	Opt.	3,598.1	3,601	3,598.0	3,598	3,598	3,601	0.08
7	austr._iptv	Feas.	4,886.7	4,897	4,888.2	4,899	4,899	4,903	0.08
8	canada_pw	Feas.	5,668.6	5,671	5,670.2	5,671	5,671	5,671	0.00
9	china_pw	Feas.	3,056.1	3,116	3,054.3	3,083	3,083	3,136	1.72
10	france_iptv	Feas.	10,983.0	10,983	10,983.0	10,983	10,983	10,998	0.14
11	singap._pw	Feas.	6,986.0	6,986	6,986.0	6,986	6,986	6,987	0.01
12	spain_iptv	Feas.	6,655.0	6,655	6,655.0	6,655	6,655	6,655	0.00
13	uk_iptv	Feas.	9,948.0	9,948	9,948.2	9,950	9,950	9,950	0.00
14	us_iptv	Feas.	5,560.0	5,560	5,560.0	5,560	5,560	5,560	0.00
15	yt._gold	Feas.	107,436.2	107,439	107,441.1	107,452	107,452	107,452	0.00
16	yt._prem.	Feas.	67,862.0	67,862	67,862.0	67,862	67,862	67,862	0.00

solutions on these smaller, well-constrained instances. On the ten larger real-life instances (IPTV, PW, and YouTube), CP-SAT remains feasible within the time limit, and the benefit of warm-starting is most visible: the solver begins from a strong incumbent and uses its remaining budget to close the gap further, yielding additional gains of 1.72% on *china_pw* and 0.14% on *france_iptv*. GA and GA+LS produce near-identical results across most instances, suggesting that the local search phase reliably reaches high-quality solutions that serve as effective starting points. Overall, the pipeline is designed so that each component compensates for the weakness of the other: GA+LS explores the space quickly and heuristically, providing CP-SAT with a strong incumbent, while CP-SAT leverages that incumbent to provide final refinement and, when optimality is proven within the time limit, global optimality guarantees; otherwise, it returns the best feasible incumbent found within the available computational budget.

4 Conclusion and Future Work

We introduced the *Smart TV Scheduling in Public Venues* problem and proposed a hybrid pipeline combining a Genetic Algorithm (GA) seeded from the better of a Beam Search and a greedy segment-based initial solution, a Local Search refinement phase (GA+LS), and a CP-SAT exact solver warm-started from the best GA+LS incumbent. Experiments on 16 benchmark instances spanning synthetic TV and real-life IPTV, provider-based, and YouTube scenarios demonstrate that GA+LS consistently produces high-quality solutions that enable CP-SAT to confirm optimality on all six synthetic instances and to achieve competitive scores on the ten larger real-life instances within a 3,600-second time

limit. The results highlight that neither a pure heuristic nor a standalone exact solver is sufficient at this scale: the strength of the pipeline lies in the complementarity of its components, with GA+LS providing fast heuristic exploration and CP-SAT leveraging the resulting incumbent to deliver the refinement and optimality guarantees that a heuristic alone cannot offer.

Future work will explore alternative metaheuristic strategies – such as ant colony optimization and adaptive large neighborhood search with learned operator weights [6] – as warm-start providers for CP-SAT, to investigate whether stronger incumbents can further reduce the gap on the hardest instances such as *china_pw*, where CP-SAT still improved upon the GA+LS warm-start by 1.72%. Extensive experimentation on the full benchmark dataset [8] beyond the 16 instances studied here also remains an important direction, as does the investigation of decomposition strategies that could make the exact CP-SAT formulation more tractable on very large instances such as *us_iptv* (115,446 programs).

References

1. Berisha, Y.: Smart TV scheduling – CP-SAT solver with OR-Tools. `https://github.com/yllberisha/tv-scheduling-ortools` (2025), accessed: 2025
2. Bylygbashi, S.: Smart TV scheduling – genetic algorithm solver. `https://github.com/shefketbylykbashi/smart-tv-scheduling-ga` (2025), accessed: 2025
3. Bylygbashi, S.: Smart tv scheduling: Segment-based deterministic scheduler (sbds) (2025), `https://github.com/shefketbylykbashi/smart-tv-scheduling-sbds`. University of Prishtina, Faculty of Electrical and Computer Engineering
4. Hazrolli, F., Vllasaliu, E., et al.: Metaheuristic tv scheduling: Implementation of beam search and simulated annealing for smart tv scheduling in public venues (2025), `https://github.com/fisnikhz/metaheuristic-tv-scheduling`. University of Prishtina, Faculty of Electrical and Computer Engineering
5. Krasniqi, K.: Smart tv scheduling validator (2025), `https://smarttvschedulingvalidator.netlify.app/`. Accessed: March 2026
6. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4), 455–472 (2006)
7. Shala, E.: Tv scheduling validator: An online validation tool for the smart tv scheduling optimization problem (2025), `https://tv-programms-validator.netlify.app/`. Accessed: March 2026
8. Sylejmani, K., Bylygbashi, S., Lajçi, U., Kastrati, Z.: A curated dataset of multichannel tv program schedules for optimization and benchmarking. Data in Brief p. 112568 (2026)
9. Vllasaliu, E., Bylygbashi, S., et al.: AA_25-26: Implementation of smart tv scheduling approaches (2025), `https://github.com/endritavllasaliu16/AA_25-26`. Accessed: 2025