# Comparing Optimization Models for Radiotherapy Scheduling

Chiara Camilla Rambaldi Migliore*, David Stanicel*, Nysret Musliu‡, Giovanni Iacca* and Marco Roveri*
*University of Trento, Italy †University of Pisa, Italy ‡Technische Universität Wien, Austria

***Abstract*—The Radiotherapy Scheduling Problem (RTSP) involves determining an optimal schedule for patients undergoing radiation treatments, a task that has a massive impact on clinical outcomes given the central role of radiotherapy in cancer care. The daily batch approach–which consists of scheduling all the newly arrived patients together at the end of each day–modelled with Integer Linear Programming, is currently one of the most effective methods for the RTSP. However, this kind of formulation requires substantial computational resources in terms of time and memory. Here, we address these limitations by developing two novel greedy heuristics (named RTSP First Fit and RTSP Best Fit) and use them as constructive heuristics for a Simulated Annealing (SA) approach to optimize the scheduling. The proposed methods–the heuristics alone and their combination with SA–are evaluated on a publicly available dataset against an integer linear program formulation solved with two different state-of-the-art exact solvers. Evaluation metrics include six scheduling objectives capturing patient waiting times, preference satisfaction, and changes in linear accelerator assignment (aggregated in four different weight configurations), solving time, and memory consumption. The results show that the novel heuristics achieve solutions close to those of exact methods, while dramatically reducing runtime and memory usage; furthermore, when combined with SA, they further improve the solution quality while maintaining low runtime and memory usage.**

***Index Terms*—Radiotherapy Scheduling, Heuristic, Simulated Annealing.**

## I. Introduction

Radiotherapy is currently one of the most effective cancer treatments [1]. It aims to deliver precise doses of radiation to tumours while preserving healthy tissue. In addition, early initiation of treatment is crucial for achieving optimal results. Therefore, effective scheduling of therapies is essential to balance patient needs, resource constraints, and treatment efficacy.

The research community addressed the Radiotherapy Scheduling Problem (RTSP) [2] by introducing various formulations and algorithms, ranging from exact Operational Research (OR) methods [3], [4], [5] to metaheuristics [6], [7], [8], [9]. Existing solutions can be roughly divided into three main categories, namely: (i) *offline scheduling* approaches, which search for optimal appointments for all patients at the same time; (ii) *batch scheduling* approaches, where newly arrived patients are grouped (daily or weekly) and then scheduled, providing the scheduling procedure information about only a small group of patients; and (iii) *online scheduling* approaches, where the goal is to schedule one patient at a time, without knowledge of future patients. Besides these formulations, only a few previous works have attempted to map the RTSP to existing combinatorial problems, such as the Bin Packing Problem (BPP). Furthermore, we are not aware of any previous attempt leveraging heuristics to rapidly generate good solutions for the RTSP, and then feeding them to a metaheuristic, such as Simulated Annealing (SA), to achieve good-quality results with limited resources. Here, we address these gaps by making the following contributions:

- We adopt the 1D BPP RTSP formulation from [10] to develop two novel heuristics, inspired by the First Fit (FF) and Best Fit (BF) heuristics [11].
- We develop an SA method to further improve the heuristic solutions and evaluate three distinct variants.
- We perform a thorough experimental evaluation of the proposed methods on a public dataset simulating patient arrivals, evaluating performance across (i) four different combinations of scheduling objectives (capturing patient waiting times, preference satisfaction, and changes in machine assignments), (ii) solving time, and (iii) memory consumption. We compare heuristics alone, and their combination with SA, against the Integer Linear Programming (ILP) formulation proposed in [4] solved using both the CPLEX commercial solver and the OR-Tools CP-SAT open-source solver with different problem configurations. This comparison provides practical indications about the most suitable approach to choose depending on the available resources.

The results show that the proposed heuristics can provide the SA with good solutions from which to start the search toward the optimal solution. Moreover, both the heuristics alone and the SA implementations perform remarkably better in terms of runtime and memory usage compared to the ILP baseline formulation. This improvement is achieved while preserving a good solution quality in comparison to the exact solvers. This is an important finding, for a number of reasons, as: (1) physicians may not have dedicated servers on which they can run exact resource-intensive methods; (2) OR solvers are not always open-source, and (3) hospitals may not have enough funds for software licences.

## II. Related Works

Scheduling in healthcare is a broad topic that encompasses several specific problems. For the purpose of this paper, we focus on scheduling radiotherapy appointments, categorising approaches into (1) offline approaches, (2) batch approaches, and (3) online approaches. For each category, we first analyse the exact operations research methods, followed by the approaches based on metaheuristics.

*a) Appointment scheduling with offline approaches:* Several studies have addressed offline radiotherapy appointment scheduling, often using OR methods and metaheuristics while bypassing pre-treatment phases. Early work by Conforti et al. [12] formulated the RTSP as an ILP problem first to maximise the number of patients starting treatment and then to minimise patient waiting times [13] and patient availability [14], testing their approaches on both toy and real-world data. Vogl et al. [8], [7] studied metaheuristics—Genetic Algorithm (GA), Iterated Local Search (ILS), and a hybrid of the two—for an ion beam facility, modelling the task as a modified job shop scheduling problem with custom constraints to minimise bottleneck resource time and violations of daily *time window* constraints (e.g., early/late morning, early/late afternoon).

*b) Appointment scheduling with batch approaches:* Numerous researchers have employed batch scheduling strategies, where patient appointments are arranged over a multi-day or weekly time frame. Previous studies on batch scheduling suggest the utilisation of ILP, Mixed ILP (MILP), Column Generation (CG) methods, or combinations of these, along with metaheuristics.

Jacquemin et al. [15] introduced an ILP model to schedule patients weekly (over a 15-week period), using a small dataset with two identical machines. They considered patient availability and continuity of care,treatment patterns, optimising the use of the radiotherapy room, treating more patients, and reducing waiting times [16]. [17] used an ILP model daily with various optimisation criteria, determining that scheduling within 7 days post-release was optimal. [10], [18] examined an ILP model cast as a BPP on small synthetic instances and compared it with a GA using real data.

Vieira et al. [19] introduced a MILP formulation with window preferences, employing heuristics for machine pre-assignment, and then using MILP for weekly scheduling. [20] improved this by incorporating patient time preferences, reducing machine switches, and aligning more closely with patient preferences than manual scheduling. [21] introduced constraints involving doctor availability and treatment coordination, proving feasibility with data from the Thai Cancer Center. [22] refined the model by adding surgical recovery times and optimising capacity constraints, improving computational efficiency. Hybrid approaches that combine ILP/MILP and Constraint Programming (CP) were explored, such as the two-phase method proposed in [3], including (i) session assignment using ILP and (ii) scheduling with MILP and CP.

Saure et al. [23] initially explored the CG approach by modelling the RTSP as a discounted infinite-horizon Markov Decision Process, converting it into an LP, and implementing CG daily to manage appointments and overtime. Frimodig et al. [4] evaluated ILP, CG-ILP, and CP to reduce waiting times and align with patient preferences. Later, [5] improved the CG model to deal with machine unavailability, establishing the most comprehensive public radiotherapy dataset to date. Their ILP serves as the baseline for our paper.

A metaheuristic-based approach was introduced in [6], proposing an effective Local Search (LS) method for the RTSP. Maschler et al. [24] addressed the particle therapy scheduling problem, initially using MILP but finding it intractable. They then developed an algorithm with a forward-looking mechanism and proposed a Greedy Randomised Adaptive Search Procedure (GRASP) along with an iterated greedy metaheuristic for daily patient scheduling. Braune et al. [9] employed a GA-based approach and Monte Carlo simulations to manage appointment durations with variable treatment times, with a focus on uncertainty in patient preparation and exit times for daily scheduling.

*c) Appointment scheduling with online approaches:* To our knowledge, the only work that proposed an online scheduling approach for RTSP, in which patients are individually scheduled, was done in [25], incorporating patient arrival uncertainty into a hybrid stochastic online model to enhance scheduling.

*d) Novelty of this work:* To the best of our knowledge, this is the first work to tackle the 1D BPP formulation of the RTSP using newly designed greedy heuristics inspired by online 1D BPP methods, and the first to explore how these heuristics can be further sharpened via SA. Moreover, it is the first study to conduct a systematic, head-to-head comparison of these heuristic and metaheuristic approaches against exact methods on a common dataset.

## III. Problem formulation

The RTSP consists of scheduling appointments for all patients' **fractions** (single radiotherapy sessions delivering part of the total dose), i.e., assigning each fraction a **day** and a **time window** (a predefined part of the day), on each specific **machine**, subject to some constraints. The formulation and solution quality metric presented here are taken from [4].

*1) Problem Input.:* The problem has three main inputs: a list of **patients**, a **time horizon**, and a list of **machines**. Each patient has a given number of fractions to schedule and an associated treatment protocol, which defines the priority of the treatment and the preferred and allowed machines on which the fraction can be performed. Each patient also has a minimum start day–which from now on will be called the "arrival day"–and a target day, i.e., the maximum day after which it is considered late to start the treatment. Patients may or may not have a preference for the time window of the day in which to be scheduled (for example, early/late morning

or afternoon). The time horizon is defined by the number of days considered for the scheduling. Each machine has a residual time capacity for each time window of each day of the time horizon. Some machines may be partially or fully beam-matched, meaning that the same radiotherapy plan can be safely administered by any of them. Machines are considered fully matched when they are located in the same building; if they are in different buildings, they are partially matched.

*2) Solution.:* A solution to the RTSP consists of the complete appointment calendar for each patient. A feasible solution satisfies the following criteria:

1) **Singularity**: There cannot be more than one fraction per day; since a day is divided into several time windows, each fraction must be scheduled only in one time window of a machine for each day.
2) **Consecutiveness**: For each patient, fractions must be scheduled consecutively on weekdays.
3) **Availability**: A fraction can be scheduled on a machine in a time window only if there is enough time remaining for that machine in that time window.
4) **Starting**: A patient can only start treatment on protocol-allowed days, not before their arrival, and no later than the last day of the time horizon minus the total number of fractions to be scheduled.
5) **Specificity**: The fractions of each patient can be performed only on the allowed machines specific to their treatments.
6) **Protocol precedence**: For two patients sharing the same treatment protocol, the one with the earlier target day must start no later than the other.

*3) Evaluation Metrics.:* The comparative analysis uses three evaluation metrics: solution quality, solving time, and memory usage. Solution quality is computed as a weighted sum of six objectives, plus an offset of 1, as in [4]:

$$\text{Obj. Sum} = 1 + \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3 + \alpha_4 f_4 + \alpha_5 f_5 + \alpha_6 f_6 \quad (1)$$

Each objective is described as follows:

- $f_1$: the weighted sum of days patients are waiting before starting the treatment (the weights are 10, 3, 1, respectively, for priority 1, 2, and 3, where the lower value for priority means higher precedence);
- $f_2$: the weighted sum of days beyond the deadline for patients to start their treatment (the weights are the same as for $f_1$);
- $f_3$: the number of times patients switch time windows between two fractions;
- $f_4$: the sum of the deviations from the time window preferred by the patients over all fractions;
- $f_5$: the number of patients' fractions scheduled on a non-preferred machine as per the treatment protocol;
- $f_6$: the sum of switches between machines that are partially beam-matched, i.e., machines whose treatment plans are interchangeable (they can be thought of as clones), but located in different buildings.

We denote by $\#\text{n} : (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ a combination of weights. As in [4], we consider, in total, the following four combinations:

$\#1 : (50, 100, 1, 0, 10, 10)$ $\#2 : (50, 100, 1, 1, 0, 0)$
$\#3 : (100, 0, 1, 0, 10, 0)$ $\#4 : (100, 0, 1, 5, 10, 10)$.

Combination #1 suits hospitals that do not consider patients' preferred treatment times. Combination #2 suits cancer centres without multiple buildings and without preferred machines. Combination #3 suits hospitals with no target days, no patient preferences, and no preferred machines. Combination #4 considers all objectives except deadlines and thus fits cancer centres that consider everything but deadlines.

## IV. Methodology

This section presents the proposed methodology used to tackle the RTSP. In particular, we briefly recall the ILP model, then describe the two proposed heuristics and the SA, along with its different implementations.

### A. Integer Linear Programming Model

In order to compare our approach with the state-of-the-art formulation from [4], we used their ILP formulation and implemented the model with both the Google OR-Tools framework and the CPLEX Python API. The problem is formulated with three principal pseudo-Boolean variables for each patient $p$, namely: (1) $q_{p,m,d,f}$, which takes value 1 if the fraction $f$ of the patient is scheduled on day $d$ and machine $m$ and 0 otherwise; (2) $x_{p,m,d,w}$, which takes value 1 if the patient has a fraction scheduled for day $d$ on machine $m$ in the time window $w$ and 0 otherwise; (3) $t_{p,m,d,w}$, which can be described as $x_{p,m,d,w}$ but only for the first fraction. Those variables are used to model the constraints and objective function presented in the previous section, along with additional helper variables. For further details on the formulations, see [4].

### B. RTSP Heuristics

Extending the work from [10], we reinterpret the RTSP as a modified 1D BPP by incorporating time windows. We provide a visual representation of this mapping in Figure 1. According to this modified 1D BPP, items correspond to fractions and are grouped by patient, while bins correspond to machines operating during a specific time window, grouped by day.

Based on this analogy, we design two new greedy heuristics drawing inspiration from those developed for the 1D BPP [26]. Both heuristics deal with one group of items at a time, store the items in groups of bins, and have a mechanism to keep track of the consecutiveness in placing the items. Patients (group of items) are ordered by arrival day, treatment priority, and target day (the maximum number of days they can wait before starting treatment).

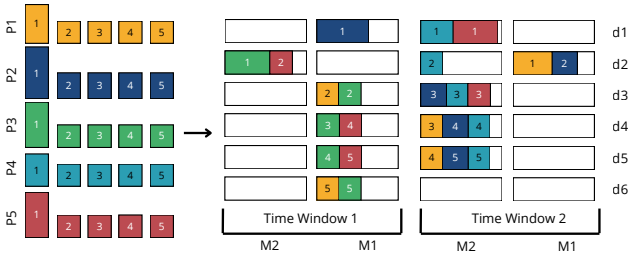


Fig. 1: A visual example of the modified 1D BPP. Each colour indicates a patient. Each patient's fraction must appear no more than once in each bin's group.

*a) RTSP First Fit (FF):* The algorithm starts by ordering the patients by arrival day, treatment priority, and target days, so that they can then be selected in an order that resembles the real one used by the physicians (for each day, the patients with higher priority and a closer target day are scheduled first). Once a patient (corresponding to a group of items) is selected, the heuristic tries to find the first $n$ consecutive groups of bins, where $n$ is the number of items in the selected group (patient), so that in each $i$-th group of bins, at least one allowed bin has enough capacity to store the $i$-th item. In the general FF, if $n$ consecutive groups of bins cannot be found, new groups of bins should be added; however, in our case, the algorithm starts with enough groups to store all the items (this number is taken from the dataset instances), so this part is unnecessary.

*b) RTSP Best Fit (BF):* As in RTSP FF, this algorithm first orders patients by arrival day, treatment priority, and target days. Then, it attempts to store the selected group of items (i.e. patient's fractions) in the $n$ consecutive groups of bins, so that in each $i$-th group, the chosen bin is the one with the least remaining capacity and that can accommodate the $i$-th item. Similar to the RTSP FF heuristic, this algorithm starts with a sufficient number of groups (this fixed number is taken from the dataset instances) to store all items by design. This entails that adding new groups of bins is not necessary.

*c) Complexity analysis:* Indicating with $p$ the number of patients, $d$ the number of days, and $f_{max}$ the maximum number of fractions prescribed to patients among all patients, both FF and BF have a complexity of $\mathcal{O}(\max(p \cdot log(p), p \cdot d \cdot f_{max}))$. The $p \cdot log(p)$ term corresponds to the complexity of the sorting of *PatientsList*, while $p \cdot d \cdot f_{max}$ is the complexity of the *for* loops. From a wide analysis of the RTSP instances, it results that $d \cdot f_{max}$ is always greater than $p \cdot log(p)$. We can therefore conclude that, for both algorithms, the complexity is $\mathcal{O}(p \cdot d \cdot f_{max})$.

### C. RTSP Simulated Annealing (SA)

The two previously introduced heuristics are further improved by SA, which uses the solution found by either of the two heuristics as a starting point for optimisation. As discussed in [27], SA can benefit from starting from solutions constructed with heuristics to avoid a wasteful initial search that would otherwise be needed starting from random solutions. SA was chosen because it can escape local minima by occasionally accepting worsening moves and because it is easy to start its search from a solution found by heuristics. Furthermore, it is widely used and studied in combinatorial problems, where one can include different kinds of domain-specific moves, such as the case of the RTSP. Three different implementations of the algorithm have been developed: **plain**, **daily**, and **reheat**. Each algorithm takes as input the initial temperature $t_{start}$, the cooling factor $\alpha$, four parameters serving as the size of the neighbourhood of possible moves, and five parameters for the probability of each move being chosen; the maximum number of iterations $k_{max}$ is fixed. All shift operations' resulting indexes are computed modulo the corresponding domain size. The moves are the same for each implementation:

- **Shift Time Window (m0)**: the move shifts by $k$ the time window assigned to a randomly chosen patient on $j$ days. The shift amount $k$ is decided as $k \leftarrow randomInt(1, tsm)$, where $tsm$ is the *time window shift* parameter. The number of days $j$ is decided as $j \leftarrow randomInt(1, tdm)$, where $tdm$ is the *time window days* parameter.
- **Shift Machine (m1)**: this move shifts by $m$ the machine assigned to a randomly chosen patient on $j$ days. The shift amount $m$ is decided as $m \leftarrow randomInt(1, msm)$, where $msm$ is the *machine shift* parameter. The number of days $j$ is decided as $j \leftarrow randomInt(1, mdm)$, where $mdm$ is the *machine days* parameter;
- **Swap Time Windows (m2)**: two random patients are chosen for swapping their assigned time windows on the same–randomly picked–day;
- **Swap Machines (m3)**: two random patients are chosen for swapping their assigned machines on the same–randomly picked–day;
- **Shift Start Day (m4)**: a random patient is chosen for shifting–backward if possible, forward otherwise–their first day of treatment; all the consequent fractions are then shifted accordingly.

The acceptance criterion is always the standard one: accept a new solution if it is better than the current one, and accept it with probability $e^{-(obj_{curr} - obj_{last})/t}$ if it is worse. The SA *plain* implementation corresponds to the standard SA algorithm, as typically presented in foundational literature [28]. The SA *daily* algorithm follows a day-by-day batching logic: the heuristic (fixed at the beginning of the algorithm–either RTSP FF or RTSP BF) generates a schedule for the patients assigned to a day $d$. This schedule is then optimised through SA, machine occupancies are updated, and the algorithm proceeds to the next day. This process is repeated sequentially until all daily batches of patients have been scheduled. The SA *reheat* algorithm follows the same logic as the plain one, but the temperature is restored after $k_{max}/10$ iterations, hence 10 times, during the course of the run.

## V. Evaluation

### A. Dataset and experimental setup

*a) Dataset:* We use a public synthetic dataset generated by [4] based on historical real-world data collected from Iridium Netwerk, Belgium's largest cancer centre. The dataset follows a Poisson distribution and is composed of 80 instances, divided into four setups, which are organised as follows (with $\lambda$ being the average arrival rates and $W$ the time windows per day): 20 instances are generated with $\lambda = 16$ and $W = 2$; 20 with $\lambda = 16$ and $W = 4$; 20 with $\lambda = 18$ and $W = 2$; and 20 with $\lambda = 18$ and $W = 4$.

*b) Experimental Setup:* We executed all the experiments on a Linux workstation with 256 GB of RAM, two Intel(R) Xeon(R) Gold 6238R CPUs, each featuring 28 physical cores with a base clock speed of 2.20 GHz.

We implemented the proposed heuristics and SA using Python v3.11.11. The baselines chosen for comparison were the ILP formulation solved with the CPLEX solver (hereafter referred to as ILP CPLEX) and with the Google OR-Tools CP-SAT (hereafter referred to as ILP CP-SAT). We implemented these encodings to the best of our knowledge, as the original code is not publicly available[1]. For ILP, we used the CPLEX v22.1.1 Python API with Python v3.10.16, while we used Google OR-Tools CP-SAT with Python v3.11.11. For CPLEX and Google OR-Tools, we used the default settings.

The ILP algorithms operate on two modalities: a) daily schedule: scheduling is performed at the end of each day, taking into account all patients who arrived that day. Then, the machines' capacity is updated, and the next day is scheduled until all patients have been treated. b) all patients' schedules at a time: we take all the patients from the dataset and schedule them all at once to find what the best schedule would be if we knew in advance all the patients' arrivals.

Due to time constraints, the CPLEX and CP-SAT experiments were run only once, with the first seed of the series used for the SA (see later). We considered two different time limits for these exact methods: 1) 1 hour (the same time limit used by [4]) for both modalities; and 2) 200 seconds (a value closer to the average time needed by the SA algorithm–see later) for the first modality only, since the second would not have led to feasible solutions.

The heuristic algorithms are deterministic: given the same daily set of patients, they always apply the same priority-based ordering and the same scheduling rules. Therefore, running them multiple times is unnecessary. These heuristics operate in an online modality: not only do they process patients day-by-day, but within each day, they schedule patients sequentially, making irrevocable decisions one patient at a time. In contrast, each of the SA implementations (*plain*, *daily*, and *reheat*) has been run five times with five different seeds–i.e., 198743, 3947394, 50343784, 93790244, 234720309. The parameters of each SA configuration have been tuned with iRace 4.3 [29] on a subset of instances.

If any of the algorithms cannot find a solution within the considered time limit, the Obj. Sum for the unfeasible instance is set equal to the maximum Obj. Sum found over all instances, to allow for a consistent scale of the Obj. Sum values for each algorithm.

For all formulations, the number of days (bins for the heuristics) considered in the time horizon for patient scheduling is determined by each problem instance.

*c) Statistical evaluation:* Following [30] guidelines, to statistically compare the different methodologies, we used the Friedman test to reject the null hypothesis (that the methodologies compared are not statistically different), and we performed a pairwise analysis using the Wilcoxon signed-rank test. For overall ranking, we used Critical Difference (CD) diagrams [31], which show the rank from the worst to the best–left to right–of each methodology compared, and draw a horizontal tick line between methodologies that are not significantly different in terms of the metric compared.

### B. Results

*a) ILP:* Figure 2 presents the performance comparison of the ILP implementations. *CPLEX* refers to the all-patients schedule modality with a 1-hour time limit, while all the others will refer to the daily batch modality; the time limit is indicated in the legend, i.e., *CPLEX 1h* uses a time limit of 1 hour. We remark that ILP CP-SAT, for all patients' schedules with a 1-hour time limit, and for the daily schedule with a 200-second time limit, produced poor-quality solutions for the former and no solution at all for the latter; therefore, we decided not to report them in the plots or discuss them further.

From the figure, it can be seen that the *CPLEX 1h* configuration is the one that can find better results in terms of solution quality and memory usage metrics, while *CPLEX 200s* is obviously–by construction–the best in terms of solving time, but with really poor results in terms of solution quality. The other two configurations use both more solving time and memory, but yield solutions of better quality. Overall, across all the weighted objective combinations and instances–encompassing a total of 320 experiments–*CPLEX* found a solution for 250 experiments, *CP-SAT 1h* for 293 experiments, *CPLEX 200s* for 112, while *CPLEX 1h* solved every instance for every weighted objective combination. For subsequent comparisons, only *CPLEX 1h* and *CPLEX 200s* have been chosen. The first is the best one on almost all the evaluation metrics, while the latter is the one with a solving time comparable to the SA.

*b) Heuristics:* The heuristics are the algorithms with better solving time and memory usage. RTSP FF can solve the instances over all weighted Obj. Sum combinations in 0.5 seconds, using 0.1 GB of memory on average, while RTSP BF takes 3.3 seconds and uses 0.1 GB of memory on average. Since they do not actively search for

[1]We had several exchanges of emails with the authors of [4] for clarifications; however, the code was neither released nor made available to us for comparison.

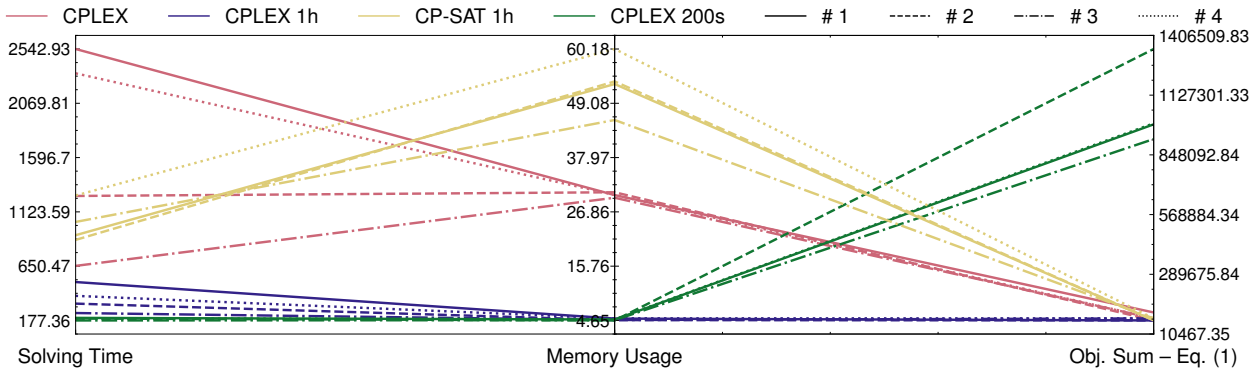


Fig. 2: Parallel coordinate plot of solving time (seconds), memory usage (GB), and best Obj. Sum value (see Equation (1)), for each weighted objective combination (#1–#4), averaged over all instances. Colours represent solvers; line styles, weighted combinations.

an optimal solution, they can reach sub-optimal values, as shown in Table I, where the Obj. Sum value, averaged across all instances, is presented on the first row with each combination on each sub-row.

*c) Simulated Annealing:* The SA *plain*, *daily*, and *reheat* implementations have been tested on all different instances and combinations of weighted Obj. Sum. The results of these experiments are reported in Figure 3 (a), where it is clear that the *plain* implementation is not only statistically different from the other for both starting solutions, but it is also the best one in terms of solution quality. Therefore, for the subsequent comparisons, we focus only on the *plain* SA.

### C. Critical discussion of the results

Figure 3 (b) shows that the *plain* implementation of the SA effectively improves the quality of the solution obtained from the heuristic algorithms. Moreover, as RTSP FF yields better solutions than RTSP BF, so does the SA starting from the former (SA_FF) compared to the SA starting from the latter (SA_BF).

The CD diagram in Figure 3 (c) compares the *plain* SA and the two ILP methods chosen for comparison, i.e., *CPLEX 1h* and *CPLEX 200s*. This diagram clearly shows that both the SA *plain* algorithms, i.e., the one starting from RTSP FF solutions and the one starting from RTSP BF solutions, can outperform the CPLEX solver when using a comparable amount of time, i.e., the *CPLEX 200s* setting. On the other hand, when CPLEX is given more time to find the solution, i.e., *CPLEX 1h*, it can outperform SA algorithms in terms of solution quality. Table I presents a more detailed comparison of the evaluation metrics results for *CPLEX 1h*, *CPLEX 200s*, *SA_FF*, and *SA_BF*.

*a) Ablation:* To better evaluate SA performance, an ablation study has been performed. The study is conducted by analysing how the SA performs when using all neighbourhoods except one [32]. To exclude a move, the probability associated with it is zeroed, while the remaining are scaled to sum to 1, maintaining their in-between ratio. A Friedman test with a significance level $\alpha = 0.05$ was performed against the plain SA, and all ablation experiments were performed with the starting solution obtained by both RTSP FF and RTSP BF, to determine if they perform equally. For the SA_FF experiments, the *p-value* is $1.60e^{-194}$, while for the SA_BF the *p-value* is $4.33e^{-222}$, rejecting in both cases the null hypothesis and therefore stating that the two sets are statistically different.

Following the Friedman test, we further analyse the configurations using the CD diagram in Figure 3 (d) and (e), comparing the ablations of SA starting from the RTSP FF (*SA_FF*) and RTSP BF (*SA_BF*) solutions. For each ablation, the SA variant in which the probability of performing a given move is set to zero is denoted as *no_m#*, where # refers to the move index defined in the description of the SA methodology.

For *SA_FF*, the only move whose removal does not produce results statistically different from the full algorithm is *Swap Machines (m3)*. This is likely due to the fact that many patients can be treated on multiple beam-matched machines, making a large portion of machine swaps ineffective. When two patients exchange two beam-matched machines that are also among their preferred options, the weighted objective value remains unchanged (see the $f_5$ and $f_6$ objectives in the Problem Formulation section). As expected, disabling any of the other moves leads to statistically worse outcomes.

For *SA_BF*, the CD diagram shows that none of the ablations are statistically equivalent to the whole algorithm: removing a single move consistently leads to inferior performance.

*b) Statistics:* To understand how SA explores the neighbourhood, we analyse the percentages of accepted and improving moves for each weighted sum combination and SA variant, using First Fit as starting heuristic since it yields better performance. Accepted moves are those whose resulting solutions are retained by SA, while improving moves are those that actually produce better solutions.

As shown in Table II, *m0* is more relevant for combinations #2–only for SA *plain* implementation–and #4,

TABLE I: Comparison of SA and ILP performance across Obj. Sum, time, memory, and heuristic baselines. Values are presented as mean ± std. dev. across all instances and across seeds for the SA cases. The Obj. Sum. values must be multiplied by $10^3$.

| Comb. | | CPLEX 1h | CPLEX 200s | RTSP FF | RTSP BF | SA_FF | SA_BF |
|---|---|---|---|---|---|---|---|
| Obj. Sum | #1 | **10.53** ± 2.99 | 1019 ± 178 | 14.17 ± 4.37 | 25.04 ± 4.43 | 12.67 ± 3.93 | 13.93 ± 4.04 |
| | #2 | **10.56** ± 3.01 | 1407 ± 595 | 13.23 ± 4.41 | 14.87 ± 4.41 | 11.28 ± 3.08 | 11.38 ± 3.18 |
| | #3 | **20.96** ± 5.91 | 943 ± 384 | 23.20 ± 8.17 | 25.71 ± 7.90 | 21.94 ± 6.21 | 22.55 ± 7.34 |
| | #4 | **21.16** ± 5.99 | 1024 ± 189 | 32.06 ± 9.47 | 48.28 ± 10.47 | 26.58 ± 8.10 | 27.30 ± 7.66 |
| Time | #1 | 512.12 ± 482.08 | 199.76 ± 2.16 | **0.53** ± 0.12 | 3.34 ± 0.001 | 24.91 ± 0.98 | 40.21 ± 3.49 |
| | #2 | 324.05 ± 285.04 | 193.99 ± 9.80 | **0.53** ± 0.16 | 3.29 ± 0.001 | 32.49 ± 3.28 | 33.34 ± 2.06 |
| | #3 | 241.98 ± 97.73 | 177.36 ± 16.63 | **0.53** ± 0.13 | 3.30 ± 0.001 | 23.43 ± 2.40 | 29.20 ± 1.69 |
| | #4 | 391.73 ± 321.68 | 195.60 ± 9.79 | **0.54** ± 0.16 | 3.26 ± 0.001 | 26.78 ± 2.50 | 36.99 ± 2.94 |
| Memory | #1 | 5.03 ± 1.20 | 4.84 ± 1.12 | 0.099 ± 0.40 | 0.099 ± 0.001 | **0.04** ± 0.0004 | **0.04** ± 0.0004 |
| | #2 | 4.73 ± 1.17 | 4.65 ± 1.05 | 0.099 ± 0.40 | 0.099 ± 0.001 | **0.05** ± 0.0021 | **0.05** ± 0.0019 |
| | #3 | 4.74 ± 1.17 | 4.71 ± 1.15 | 0.099 ± 0.40 | 0.099 ± 0.001 | **0.05** ± 0.0028 | **0.05** ± 0.0011 |
| | #4 | 5.09 ± 1.41 | 4.89 ± 1.05 | 0.099 ± 0.40 | 0.099 ± 0.001 | **0.04** ± 0.0004 | **0.04** ± 0.0004 |

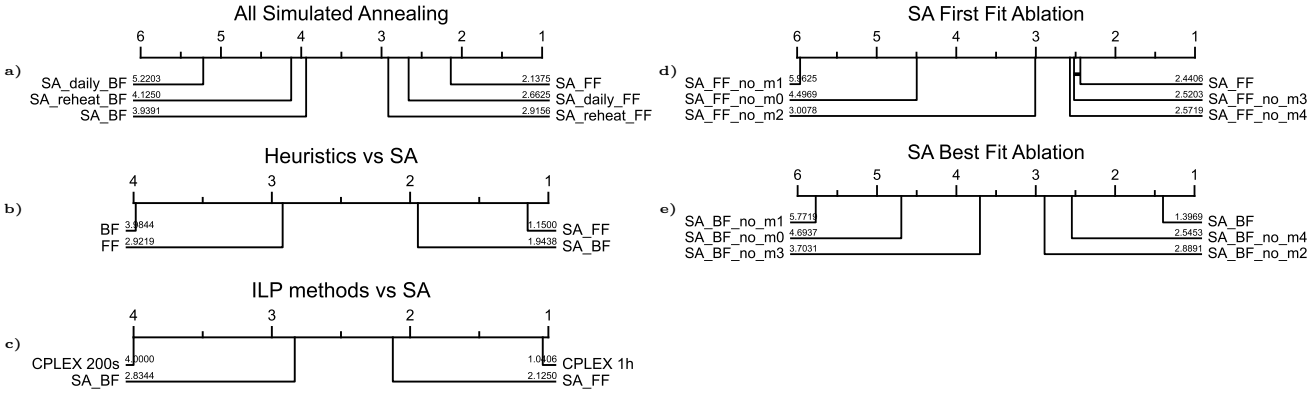


Fig. 3: CD diagrams for a) all SA implementations; b) heuristics vs. SA; c) SA vs. ILP methods; d) SA starting from FF heuristic and e) SA starting from BF heuristic.

TABLE II: Percentage of accepted and improving moves across all weighted Obj. Sum combinations and SA_FF implementations. P, D, and R stand for, respectively, Plain, Daily, and Reheat SA implementations.

| | | Comb. #1 | | | Comb. #2 | | | Comb. #3 | | | Comb. #4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P | D | R | P | D | R | P | D | R | P | D | R |
| Accepted | m0 | 6.81 | 7.17 | 6.20 | 27.33 | 1.96 | 2.92 | 4.42 | 3.75 | 4.81 | 27.27 | 28.15 | 27.23 |
| | m1 | 2.51 | 2.78 | 3.21 | 83.00 | 79.17 | 79.45 | 52.43 | 52.19 | 55.80 | 2.90 | 2.98 | 3.33 |
| | m2 | 2.12 | 5.12 | 1.73 | 1.69 | 3.91 | 1.71 | 0.98 | 3.08 | 1.18 | 2.96 | 6.34 | 2.83 |
| | m3 | 5.63 | 14.11 | 5.62 | 91.34 | 94.05 | 91.97 | 66.75 | 69.51 | 67.02 | 5.14 | 10.17 | 5.12 |
| | m4 | 0.03 | 1.03 | 0.05 | 0.02 | 1.05 | 0.01 | 0.02 | 1.06 | 0.03 | 0.04 | 1.06 | 0.05 |
| Improving | m0 | 1.03 | 0.99 | 0.86 | 0.90 | 0.49 | 1.09 | 0.40 | 0.40 | 0.47 | 1.08 | 1.63 | 1.06 |
| | m1 | 0.12 | 0.19 | 0.15 | 0.01 | 0.00 | 0.001 | 0.02 | 0.01 | 0.03 | 0.16 | 0.25 | 0.18 |
| | m2 | 0.37 | 0.46 | 0.27 | 0.47 | 0.19 | 0.49 | 0.14 | 0.12 | 0.16 | 1.35 | 1.64 | 1.28 |
| | m3 | 0.07 | 0.10 | 0.05 | 0.003 | 0.00 | 0.00 | 0.02 | 0.01 | 0.02 | 0.09 | 0.14 | 0.06 |
| | m4 | 0.03 | 0.002 | 0.05 | 0.02 | 0.003 | 0.01 | 0.02 | 0.004 | 0.03 | 0.04 | 0.009 | 0.05 |

with 27-28% accepted moves, suggesting that the move contributes more to search diversification. On the other hand, despite being the most improved move among all the others, the improved percentage is quite low–around 0.5-1.5%–suggesting that, in general, the First Fit heuristic is already finding near-to-optimal solutions and, in particular, that *m0* is the most effective one. *m1* is the predominant move for combinations #2 and #3, with very high accepted moves' percentages (up to 83%) but low improving moves' percentages, meaning that it supports exploration over direct improvement. *m2* is slightly higher for combinations #1 and #4 for the SA *daily* implementation, while it remains at lower values for the others. It's improving percentages are the second best, after *m0*, especially for combination #4, suggesting the move tends to improve more than the others when accepted. *m3* shows a large discrepancy between the percentages of accepted and improving moves, especially for combinations #2 and #3. This indicates that many machine swaps move patients among beam-matched machines without improving the quality of the solution. *m4* shows very low percentages in terms of both accepted and improving moves, indicating

that the move is rarely selected and provides little search value.

*c) Overall considerations:* The experimental evaluation highlights clear trade-offs between exact methods and heuristic/metaheuristic approaches for solving the RTSP. First, ILP can reach optimal solutions when enough runtime is available, although it requires more memory. However, its performance drops rapidly under strict time limits. Second, the proposed RTSP heuristics–despite not performing an active search–are able to produce good solutions with very low computational cost. Third, the SA algorithm can further improve the heuristic solutions while keeping runtime short and memory usage low. The analysis also shows the importance of the current neighbourhoods, and there is a potential for new neighbourhoods to be explored. In general, heuristics and SA achieve a good balance between solution quality and computational effort. Although ILP remains the most reliable choice for optimality when runtime is not restricted, SA offers a valid alternative, producing near-optimal schedules with a significant reduction in both runtime and memory usage.

## VI. Conclusion and Future Work

The RTSP has been widely tackled with exact methods and metaheuristics. However, the existing literature reveals a lack of work focused on simpler heuristics. In this paper, we developed two novel greedy heuristics for the RTSP, which are then combined with a custom SA algorithm. We compared the heuristics and three SA implementations with different exact methods. The experimental results on a public dataset simulating a real-world arrival flow to a large cancer centre demonstrate that simple heuristics such as RTSP FF and RTSP BF can drastically reduce both computational time and memory usage. Moreover, the SA starting from the solutions found by heuristics can further improve the solution quality while keeping the computational time and memory usage drastically low compared to the exact methods. This makes our approach attractive for hospitals where physicians may have access to limited computational resources and cannot allocate budget for the purchase of expensive solver licences. However, this computational efficiency comes at the cost of achieving suboptimal solutions for the four weight combinations investigated in comparison with the exact methods run for a longer time. To address this limitation, future work will focus on refining the SA moves to increase the improvement percentage, aiming at neighbourhoods that not only explore but also exploit the solution quality. Moreover, we will investigate new heuristics, extending the RTSP BF and FF to better take into account the objective function.

## References


[1] M. W. Roomi, "Radiation Therapy for Cancer Treatment," *Journal of Otolaryngology Research & Reports*, vol. 2, pp. 1–10, 2023.

[2] B. Vieira, E. W. Hans, C. van Vliet-Vroegindeweij, J. van de Kamer, and W. H. van Harten, "Operations research for resource planning and -use in radiotherapy: a literature review," *BMC Medical Informatics Decis. Mak.*, vol. 16, pp. 149:1–149:11, 2016.

[3] T.-S. Pham, L.-M. Rousseau, and P. De Causmaecker, "A two-phase approach for the Radiotherapy Scheduling Problem," *Health Care Management Science*, pp. 1–17, 2022.

[4] S. Frimodig, P. Enqvist, M. Carlsson, and C. Mercier, "Comparing optimization methods for radiation therapy patient scheduling using different objectives," in *Operations Research Forum*, vol. 4. Springer, 2023, p. 83.

[5] S. Frimodig, P. Enqvist, and J. Kronqvist, "A Column Generation Approach for Radiation Therapy Patient Scheduling with Planned Machine Unavailability and Uncertain Future Arrivals," 2023, arXiv:2303.10985.

[6] J. P. Cares, M. C. Riff, and I. Araya, "LS2R: A local search algorithm to solve scheduling radiotherapy problems," in *International Conference on Hybrid Intelligent Systems (HIS 2013)*. IEEE, 2013, pp. 256–261.

[7] P. Vogl, R. Braune, and K. F. Doerner, "Scheduling recurring radiotherapy appointments in an ion beam facility: Considering optional activities and time window constraints," *Journal of Scheduling*, vol. 22, no. 2, pp. 137–154, 2019.

[8] ——, "A multi-encoded genetic algorithm approach to scheduling recurring radiotherapy treatment activities with alternative resources, optional activities, and time window constraints," in *International Conference on Computer Aided Systems Theory*. Springer, 2017, pp. 373–382.

[9] R. Braune, W. J. Gutjahr, and P. Vogl, "Stochastic radiotherapy appointment scheduling," *Central European Journal of Operations Research*, pp. 1–39, 2022.

[10] C. C. Rambaldi Migliore, G. Iacca, and M. Roveri, "A Bin-Packing Formulation for Radiotherapy Treatment Scheduling," in *Intelligence for Climate Change, Italian Workshop on Planning and Scheduling*, 2024.

[11] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974.

[12] D. Conforti, F. Guerriero, and R. Guido, "Optimization models for radiotherapy patient scheduling," *4OR*, vol. 6, pp. 263–278, 2008.

[13] ——, "Non-block scheduling with priority for radiotherapy treatments," *European Journal of Operational Research*, vol. 201, no. 1, pp. 289–296, 2010.

[14] D. Conforti, F. Guerriero, R. Guido, and M. Veltri, "An optimal decision-making approach for the management of radiotherapy patients," *OR Spectrum*, vol. 33, pp. 123–148, 2011.

[15] Y. Jacquemin, E. Marcon, and P. Pommier, "Towards an improved resolution of radiotherapy, scheduling," in *2010 IEEE workshop on health care management (WHCM)*. IEEE, 2010, pp. 1–6.

[16] J. Yoan, M. Eric, and P. Pascal, "A pattern-based approach of radiotherapy scheduling," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6945–6950, 2011.

[17] E. K. Burke, P. Leite-Rocha, and S. Petrovic, "An integer linear programming model for the radiotherapy treatment scheduling problem," *arXiv preprint arXiv:1103.3391*, 2011.

[18] C. C. Rambaldi Migliore, D. Stanicel, M. Roveri, and G. Iacca, "Addressing Radiotherapy Scheduling with a Bin Packing Problem Formulation: A Comparative Study of Exact Solvers and Genetic Algorithms," in *Applications of Evolutionary Computation*. Cham, Switzerland: Springer Nature, 2025, pp. 475–491.

[19] B. Vieira, D. Demirtas, J. B. van de Kamer, E. W. Hans, L.-M. Rousseau, N. Lahrichi, and W. H. van Harten, "Radiotherapy treatment scheduling considering time window preferences," *Health Care Management Science*, vol. 23, pp. 520–534, 2020.

[20] B. Vieira, D. Demirtas, J. B. van de Kamer, E. W. Hans, W. Jongste, and W. van Harten, "Radiotherapy treatment scheduling: Implementing operations research into clinical practice," *PLOS ONE*, vol. 16, no. 2, p. e0247428, 2021.

[21] C. Boonmee, N. Pisutha-Arnond, W. Chattinnawat, P. Muangwong, W. Nobnop, and I. Chitapanarux, "Decision support system for radiotherapy patient scheduling: Thai cancer center case study," in *International Conference on Medical and Health Informatics*, 2021, pp. 168–175.

[22] N. Emsamrit and C. Boonmee, "Mixed-Integer Linear programming for scheduling of radiotherapy patients," *Engineering and Applied Science Research*, vol. 51, no. 1, pp. 106–116, 2024.
[23] A. Saure, J. Patrick, S. Tyldesley, and M. L. Puterman, "Dynamic multi-appointment patient scheduling for radiation therapy," *European Journal of Operational Research*, vol. 223, no. 2, pp. 573–584, 2012.
[24] J. Maschler, M. Riedler, M. Stock, and G. R. Raidl, "Particle therapy patient scheduling: First heuristic approaches," in *International Conference on the Practice and Theory of Automated Timetabling*, 2016, pp. 223–244.
[25] A. Legrain, M.-A. Fortin, N. Lahrichi, and L.-M. Rousseau, "Online stochastic optimization of radiotherapy patient scheduling," *Health Care Management Science*, vol. 18, no. 2, pp. 110–123, 2015.
[26] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin-packing—an updated survey," in *Algorithm design for computer system design*. Springer, 1984, pp. 49–106.
[27] M. Sarhani, S. Voß, and R. Jovanovic, "Initialization of metaheuristics: comprehensive review, critical analysis, and research directions," *International Transactions in Operational Research*, vol. 30, no. 6, pp. 3361–3397, 2023.
[28] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.
[29] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.
[30] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.
[31] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.
[32] C. Fawcett and H. H. Hoos, "Analysing differences between algorithm configurations through ablation," *Journal of Heuristics*, vol. 22, no. 4, pp. 431–458, 2016.