Telescope Scheduling

Georgios Christopoulos
Utrecht University
Student Number: 2789175

Abstract

The paper aims to study the Telescope Scheduling problem and describe theoretical results in combination with experimental ones to measure time complexity,hardness and how it works on an offline setting.An algorithm based on an MIP formulation was constructed for the offline scenario and for the online algorithm we will use $ALG_{SmallToBig}$ which transmits the small items first and then the larger ones. We will show that is a 2-competitive algorithm for a special case.Moreover, The algorithm seemed to achieve a lower bound of $\Omega(3/2)$ for the general instance.Finally, we show that the Telescope Scheduling problem is NP-complete.

2012 ACM Subject Classification Theory of computation Design and analysis of algorithms

Keywords and phrases Telescope Scheduling, Online algorithms

Supplementary Material https://github.com/GeorgiosChristopoulos96/AlgorithmsDecisionSupport

1 Introduction

In this paper it will be consider an MIP optimization offline algorithm which is pretty similar to the known problems of knapsack and bin-packing.On the Knapsack problem there are n items given, a knapsack volume of B, where each of n items $j_1, \ldots, j_n$ has its value a_j and weight C_j. Given the value and the weight of the items, we try to get the maximum total value in the knapsack. It is not possible to put a partial item in the knapsack.On the latter problem, given a set of bins with a common capacity, find the fewest that will hold all the items.The algorithm for Telescope Scheduling is of similar nature as we are trying to minimize the time intervals by maximizing the packing of images.

2 Formal problem definition

Recently a new telescope has been sent to space named James Webb. While Hubble operates just 535 km from Earth, James Webb does it being approximately 1.5 million km away. Knowing this fact, it makes the transmission of the images considerably harder. Being further from Earth makes the unavailabilites appear more frequently in the transmission channel thus causing image transmission failure.
The problem our algorithm tends to solve is exactly this optimization problem. While transmitting n number of images with size s_i where $n \in \mathbb{N}$,$i \leq 1 \leq n$ the presence of I_m unavailable time intervals should be considered. However, the channel is able to transmit only one image at a time. If an unavailability happens on the meantime, the image will be retransmitted. Only then the image is considered successfully transmitted. The unavailabilites come with a duration l_i as well, and $i \leq 1 \leq m$. The objective is to transmit all images soon as possible. Given the aforementioned information, we have:

- n images $\{f_1, f_2, \ldots, f_n\}$ where image f_i is of real-number size s_i
- s_i as the size of the image who needs s_i time units to be transmitted successfully
- m unavailable time intervals $\{I_1, I_2, \ldots, I_m\}$ also known as the periods where the image is unable to be transmitted.
- I_j time interval has duration:$[t_j, t_j + l_j]$ where t_j and l_j are the start time and the length at the j-th unavailable interval.

3 Preliminaries

▶ **Definition 1.** *An optimization problem* Π *consists of a set of instances or jobs* $\mathcal{J}$*, a set of feasible solutions* $\mathcal{O}$*, and a cost function*

$$\text{cost } : \mathcal{O} \longmapsto \mathbb{R}.$$

Every instance $J \in \mathcal{J}$ is a sequence of requests $J = (x_1, x_2, \ldots, x_n)$ and every feasible solution $O \in \mathcal{O}_J \subset \mathcal{O}$ is a sequence of answers $O = (y_1, y_2, \ldots, y_n)$, where $n \in \mathbb{N}$. Note that $\mathcal{O} = \bigcup_{J \in \mathcal{J}} \mathcal{O}_J$. Given an instance J and a corresponding feasible solution $O \in \mathcal{O}_J$, the cost associated with solution O is denoted by $\text{cost}(O)$. Whether the goal is to minimize or maximize the cost function, optimization problems can be further divided into minimization and maximization problems.

▶ **Definition 2.** *An optimal solution for an instance $J \in \mathcal{J}$ of a minimization (optimization) problem Π as in 1 is a solution $OPT(J) \in \mathcal{O}_J$ such that,*

$$\mathrm{cost}(OPT(J)) = \min_{O \in \mathcal{O}_J} \mathrm{cost}(O).$$

i.e., an optimal solution for a minimization problem is a feasible solution that obtains the minimum cost.

▶ **Definition 3.** *An offline problem is an optimization problem Π as in 1 such that the set of instances $\mathcal{J}$ is available all at once.*

▶ **Definition 4.** *An online problem is an optimization problem Π as in 1 such that the input instances $J \in \mathcal{J}$ are revealed sequentially.*

▶ **Definition 5.** *An offline algorithm is a rule to solve an offine problem Π. Note that due to the nature of offline problems, an offline algorithm is allowed to consider the entire set of instances $\mathcal{J}$ to compute the optimal solution of problem Π.*

▶ **Definition 6.** *An online algorithm is a rule to solve an online problem Π. Note that due to the nature of online problems, an online algorithm must make a decision upon the arrival of each request $J \in \mathcal{J}$ without knowledge about the future. Moreover, the decisions are irrevocable. That is, the decisions are permanent and cannot be changed afterwards.*

▶ **Definition 7.** *Consider a minimization online problem . An online algorithm ALG is c-competitive if*

$$\exists \alpha \in \mathbb{R} : \forall J \in \mathcal{J}, \quad \mathrm{cost}(ALG(J)) \leq c \cdot \mathrm{cost}(OPT(J)) + \alpha.$$

i.e., there exists a constant α such that for every finite instance $J \in \mathcal{J}$ the cost incurred by the online algorithm ALG is bounded by c times the cost incurred by the optimal solution.

▶ **Definition 8.** *Consider a minimization online problem Π and an online algorithm ALG. If there exists an instance J such that*

$$\frac{\mathrm{cost}(ALG(J))}{\mathrm{cost}(OPT(J))} \geq l$$

for some constant $l \in \mathbb{R}$, by definition 7 we know that, ALG cannot be c-competitive for any $c < l$. We call the constant $l \in \mathbb{R}$ a competitive ratio lower bound of the online algorithm ALG

▶ **Definition 9.** *Consider a minimization offline problem Π. The linear programming formulation or LP formulation of problem Π is,*

$$\min \sum_{i=1}^{n} c_i x_i,$$

subject to

$$\begin{array}{c} \sum_{i=1}^{n} a_{i1} x_i \leq b_1, \\ \vdots \\ \\ \sum_{i=1}^{n} a_{im} x_i \leq b_m, \\ x_i \geq 0, \quad \forall i \in \{1, \ldots, n\}. \end{array}$$

The LP formulation of offline minimization problem Π *is a way of writing down the problem such that the solution is encoded by* $n \in \mathbb{N}$ *variables* $x_1, \ldots, x_n$ *called decision variables with associated costs* $c_1, \ldots, c_n$ *and the objective is to minimize the total cost. Therefore, the objective function is given by the expression* $\min \sum_{i=1}^{n} c_i x_i$. *The* n *decision variables are subject to* $m \in \mathbb{N}$ *constraints of the form* $\sum_{i=1}^{n} a_{ij} x_i \leq b_j$, *where* $a_{ij}, b_j \in \mathbb{R}$*; as well as* n *domain constraints,* $x_i \geq 0$. *An optimal solution in this context is any solution that satisfies all the constraints and achieves minimal cost.*

▶ **Definition 10.** *Consider a minimization offline problem* Π. *The integer linear programming formulation or ILP formulation of problem* Π *is,*

$$\min \sum_{i=1}^{n} c_i x_i,$$

subject to

$$\begin{gathered} \textstyle\sum_{i=1}^{n} a_{i1} x_i \leq b_1, \\ \vdots \\ \textstyle\sum_{i=1}^{n} a_{im} x_i \leq b_m, \\ x_i \in \mathbb{Z}_+, \quad \forall i \in \{1, \ldots, n\}. \end{gathered}$$

Note that the ILP formulation of offline minimization problem Π *only differs from the LP formulation in the* n *domain constraints. In the case of ILP the decision variables* x_i *are forced to be non negative integers.*

▶ **Definition 11.** *In computational complexity theory, NP (nondeterministic polynomial time) is a complexity class used to classify decision problems. NP is the set of decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time by a deterministic Turing machine, or alternatively the set of problems that can be solved in polynomial time by a nondeterministic Turing machine.[1]*

An equivalent definition of NP is the set of decision problems solvable in polynomial time by a nondeterministic Turing machine. This definition is the basis for the abbreviation NP; "nondeterministic, polynomial time." These two definitions are equivalent because the algorithm based on the Turing machine consists of two phases, the first of which consists of a guess about the solution, which is generated in a nondeterministic way, while the second phase consists of a deterministic algorithm that verifies if the guess is a solution to the problem

▶ **Definition 12.** *NP-complete problem, any of a class of computational problems for which no efficient solution algorithm has been found. Many significant computer-science problems belong to this class—e.g., the traveling salesman problem, satisfiability problems, and graph-covering problems. So-called easy, or tractable, problems can be solved by computer algorithms that run in polynomial time; i.e., for a problem of size n, the time or number of steps needed to find the solution is a polynomial function of n. Algorithms for solving hard, or intractable, problems, on the other hand, require times that are exponential functions of the problem size n. Polynomial-time algorithms are considered to be efficient, while exponential-time algorithms are considered inefficient, because the execution times of the latter grow much more rapidly as the problem size increases. A problem is called NP (nondeterministic polynomial) if its solution can be guessed and verified in polynomial time; nondeterministic means that no particular rule is followed to make the guess. If a problem is NP and all other NP problems are polynomial-time reducible to it, the problem is NP-complete. Thus, finding an efficient algorithm for any NP-complete problem implies that an efficient algorithm can be found for all such problems, since any problem belonging to this class can be recast into any other* *member of the class. It is not known whether any polynomial-time algorithms will ever be found for NP-complete problems, and determining whether these problems are tractable or intractable remains one of the most important questions in theoretical computer science. When an NP-complete problem must be solved, one approach is to use a polynomial algorithm to approximate the solution; the answer thus obtained will not necessarily be optimal but will be reasonably close.[2]*

▶ **Definition 13.** *A decision problem H is NP-hard when for every problem L in NP, there is a polynomial-time many-one reduction from L to H.[3].An equivalent definition is to require that every problem L in NP can be solved in polynomial time by an oracle machine with an oracle for H.[4] Informally, an algorithm can be thought of that calls such an oracle machine as a subroutine for solving H and solves L in polynomial time if the subroutine call takes only one step to compute.*

Another definition is to require that there be a polynomial-time reduction from an NP-complete problem G to H.[3] As any problem L in NP reduces in polynomial time to G, L reduces in turn to H in polynomial time so this new definition implies the previous one. Awkwardly, it does not restrict the class NP-hard to decision problems, and it also includes search problems or optimization problems.

3.1 Google OR-Tools

Google OR-Tools is a free and open-source software suite developed by Google for solving linear programming (LP), mixed integer programming (MIP), constraint programming (CP), vehicle routing (VRP), and related optimization problems. OR-Tools is a set of components written in C++ but provides wrappers for Java, .NET and Python. OR-Tools provides popular solvers such as Gurobi or CPLEX, or open-source solvers such as SCIP, GLPK, or Google's GLOP and award-winning CP-SAT.

4 Results

In this section we will be going into more depth regarding the formulation of the problem and the constrains used.In the next section we will see in more detail the time complexity as well as competitiveness of the algorithm compared to an online one.

4.1 Algorithms

Offline algorithm

Sets and indices

$i \in$ I :set of images
$j \in$ J : set of unavailable time intervals

Parameters
s_i :size of image i and $1 \leq i \leq n$
l_j :duration of unavailable interval j and $1 \leq j \leq m$
t_j :the time when unavailable interval j starts $1 \leq j \leq m$
Decision variables
x_{ij} :the variable equals to 1 if the image i will be transmitted at interval j,

otherwise 0 and $1 \leq j \leq m$, $1 \leq i \leq n$
y_j :the variable equals to 1 if interval j is used, otherwise 0 ,$1 \leq j \leq m$

Objective function

We can assume that the images i will finish transmission in time T.In the objective function we try to minimize the overall cost of image transmission by minimizing any gap that may occur by any permutation of images put in each interval.This will minimize the cost by arranging the images as tight as possible.

$$min \sum_{j \in J} t_j y_j - \sum_{i \in I} s_i x_{ij} \tag{1}$$

Constraints

The Objective function is subject to constraints.At each time we can only transmit at most one image in the channel.The math formula for this is the following:

$$\sum_{j=1}^{m} x_{ij} \leq 1 \; \forall j \in J \text{ and } 1 \leq j \leq m \tag{2}$$

At each time interval we can transmit the images as long as the sum of them is smaller than t_j.Then the following formulation is :

$$\sum_{i=1}^{n} s_j x_{ij} \leq t_j y_j, \forall j \in J \text{ and } 1 \leq j \leq m \text{ and } \forall i \in I \text{ and } 1 \leq i \leq n \tag{3}$$

Image f_i will be transmitted before image $f_i + 1$

$$f_i \leq f_i + 1 \forall i = \{1, \ldots, n\} \tag{4}$$

$$x_{ij}, y_j \in \{0,1\} \, \forall i = \{1, \ldots, n\} \, and \forall j \, \{0, \ldots, m\} \tag{5}$$

5 Theoretical results

5.1 Time complexity

In this section we would like to give a few results regarding the complexity of the Telescope Scheduling problem . To do so, we first consider a more restrictive decision version of the Telescope Scheduling problem. Recall the Telescope Scheduling problem as stated below is:

Given a set of images I, and a set J the number of unavailable time intervals.Transmit all the n images to Earth as soon as possible. (1)

The restrictive version of our problem is :

Assuming that there are standard unavailable time intervals happening every j-th time that creates intervals of capacity $t_j = C \; \forall j \in \mathbb{Z}+$ and $C > 0$

So it transforms to the following decision problem :

Given a set of images J, can all the images be transmitted with using at most k intervals? (2)

The aim is to show the following :

▶ **Theorem 14.** *The Telescope Scheduling problem is NP-Complete*

To be able to prove that our problem is NP-complete we will have to prove that our problem is NP and NP-Hard :

To show the Telescope Scheduling problem is NP, we need to show it can be verified in polynomial time. For an instance of Jobs to be transmitted in the respective time intervals $\langle \mathcal{F}, S, t_j, T \rangle = \{\{s_{1,1}, \ldots, s_{1,n}\}, \ldots, \{s_{n,1}, \ldots, s_{n,n_n}\}\}$ we find a suitable certificate to a feasible transmission of all the images such that:

$c = \{(S_i, t_j, T) \mid S_i \subseteq S_{i,n}$ the subset of images that have to be transmitted, $S_i \leq t_j$,and finish transmission before time T for $1 \leq i \leq n$ and $1 \leq j \leq m\}$,

Our verifier V will check $\langle \langle S, t_j, T \rangle, c \rangle$ if:
1) The certificate c is a sequence of S.If not true, then **reject**.
2) The certificate c uses at most k time intervals which finish at the latest at time T.If not true,then **reject**.
3)It will parse each image transmission interval in S and verify it respects the capacity of the time interval t_j(not overlapping with the blackout).If it respects it **accept**, else **reject**
4)If steps (1),(2),(3) pass then accept,else **reject**

Hence, n is the number of images to be transmitted in S and m the number of unavailable time intervals.Such instance is always feasible since it has to check for step 1 at most n^2 times for each Job and m times for each unavailable time interval.In the worst case it will have to parse all Jobs in S and unavailable time intervals which accounts to m+n.By following the previous steps, its proved that our verifier runs in polynomial time.We can conclude the following:

▶ **Lemma 15.** *The decision problem (2) is NP.*

Since problem (2) is the decision problem of the Telescope Scheduling problem we can conclude the following corollary:

▶ **Corollary 16.** *The Telescope Scheduling problem, is in NP.*

To prove the NP-Hardness of our problem we need to make a reduction from another NP-hard problem [5, 6] i.e Partition Problem.
Given a set of positive integers S,is it possible that S can be partitioned into two subsets such that

$$\sum_{x \in T} x = \sum_{y \in S/T} y \tag{1}$$

Assuming that the unavailable time intervals happen at every j-th interval.Standard intervals, will produce k standard sized "bins" with capacity $t_j = C \ \forall j \in \mathbb{Z}+$ and $C > 0$ (2)

To give the reduction we define a function f which takes an instance of problem (1), and transforms that instance into one for problem (2) in polynomial time. This function f takes an instance S and we construct S' and k as follows.For each element $a_i \in S$, there is an element $u_i \in S'$ and $s(u_i) = \frac{2*a_i}{X}$ where X is half the of the sum of elements in S. We set . The construction can be done in polynomial time. Now we prove that the reduction works. Suppose that there is a partition of S,S_1 and S_2 For all elements $a_i \in S_1$, the sum is X. The sum of corresponding u_i's is 1, so the corresponding items can be placed in one bin. It also holds for S_2. Hence, 2 bins would suffice(k=2)[7]. For the other direction, suppose that the items in S' can be packed in two bins. Each of the bin has total size 1 since the total size of all items in S is $\sum is(u_i) = \frac{\sum i*a_i}{X} = 2$. The corresponding two subsets of S has equal size and form a partition[8]. Above it was shown that f is a polynomial reduction from an NP-Hard problem to the restricted problem of the Telescope Scheduling problem. Since the restricted version of the Telescope Scheduling problem is NP-Hard then also the Telescope Scheduling problem is NP-Hard since it has more arbitrary variables which increase its complexity.From this proof we obtain that

▶ **Corollary 17.** *The Telescope Scheduling problem is NP-Complete*

5.2 Competitive ratio

For the competitive ratio we will be considering the case where we have one unavailable time interval, **m=1**.This will create the situation where we only have two time intervals.For this problem we will assume that we have an online algorithm $ALG_{SmallToBig}$.The way the Online algorithm works is that it will always transmit the smallest images first and after the bigger ones.Our problem definition contains the following variables:

- set of images $S = \{S_1, \ldots, S_n\}$
- unavailable time interval $B = [t_s, t_e]$

Proof. We immediately distinguish two cases in this problem:
- Case 1: $size(S) \leq t_s$
- Case 2: $size(S) > t_s$

In the first case our OPT(S) will be the same as $ALG_{SmallToBig}$ since $size(S)$ ends transmission before the unavailable interval even starts.Then, OPT(S) = $ALG_{SmallToBig} = S$.The competitive ration in this case will be $c = 1$

In the second case We will assume that $t_s \rightarrow S$ then our offline instance will pay OPT(S) = $l_s + S$ where l_s is the duration of the unavailable interval from t_s to t_e.On the other hand, $ALG_{SmallToBig} = S + l_s + S$ since $t_s \rightarrow S$ it won't manage to transmit the set of images in the first time interval.The competitive ratio by definition is defined as follows

$$\boxed{\frac{ALG_{SmallToBig}}{OPT(S)} \leq c.}$$

By plugging the cost of each instance we get : $\frac{S+l_s+S}{l_s+S} \leq \frac{2S+l_s}{l_s+S}$.

For this instance we assume that l_s is really small and $l_s \rightarrow 0$ then the ratio is $\frac{2S}{S} \leq 2$.

Hence: $\boxed{\frac{ALG_{SmallToBig}}{OPT(S)} \leq 2 \Rightarrow ALG_{SmallToBig} \leq 2 * OPT(S)}$ ◀

5.3 Tight analysis

We show the analysis is tight by designing an instance such that the ratio of the algorithm cost to the optimal cost on the same instance matches the competitive ratio upper bound.We consider the instance that the unavailable interval starts at the $(t_s - e)$-th unit of time, where e > 0 and $e \to 0$. On this instance, the algorithm cost is $ALG_{SmallToBig} = S + l_s + S$, while the optimal cost is $l_s + S$. The ratio of the algorithm cost on the input to the optimal cost on the same input $\frac{S+l_s+S}{l_s+S} \leq \frac{2S+l_s}{l_s+S}$ and with $l_S \to 0, \frac{2S}{S} = 2$ which matches the upper bound of the algorithm's competitive ratio. Therefore, **the analysis is tight**.

5.4 Lower bound

Similarly to [9] for the lower bound we will use the previous Algorithm $ALG_{SmallToBig}$.To prove a strong lower bound we need to construct such a case that our algorithm will have its cost increased as much as possible since we are talking about a minimization problem.Lets consider the set of unavailabilities $I = \{I_1, \dots, I_m\}$ and $S = \{S_1, \dots, S_n\}$ the set of images that have to be transmitted.The online algorithm transmits from smallest to biggest size of image.Each unavailability tends to have the size of the respective image for example $I_1 \to S_1$.In this case the adversary will place the unavailabilities in an opposing manner from the online algorithm.When the algorithm will start transmission with image S_1 it will actually cost the size of the largest image of the sorted set which is S_n.At some moment $\frac{n}{2}$ the size of the image will fit the $\frac{n}{2}$-th time interval perfectly.After the $\frac{n}{2}$-th interval the following intervals will fit the $\frac{n}{2}$ smaller images that have already been transmitted.That means that until all unavailable time intervals finish,no $\frac{n}{2}$ bigger images will be transmitted.After the n-th unavailable time interval the $ALG_{SmallToBig}$ will transmit the remaining $\frac{S}{2}$ images. The offline solution will cost $OPT = S + n * l_s$.$ALG_{SmallToBig}$ will first transmit $\frac{S}{2}$ small images and after it will wait $\frac{S}{2}$ since none of the bigger images will be able to fit in the intervals.After the last unavailable interval $ALG_{SmallToBig}$ will send $\frac{S}{2}$ bigger images.In the end $ALG_{SmallToBig}$ cost is $\frac{S}{2} + \frac{S}{2} + \frac{S}{2} + n * l_s$

From the following we get that:

$$\boxed{\frac{ALG_{SmallToBig}}{OPT(S)} \geq l \Rightarrow \frac{\frac{3*S}{2} + n * l_s}{S + n * l_s}}$$

with $l_s \to 0$ the following ratio becomes as follows:

$$\boxed{\frac{ALG_{SmallToBig}}{OPT(S)} \geq \frac{3}{2}}$$

From the above we get the following

▶ **Corollary 18.** *The Telescope problem has a lower bound of* $\Omega(3/2)$

6 Practical results

6.1 Technical specifications

The programming language of choice in order to implement the offline algorithm was Python 3.10. The code was executed on MacOS Ventura. The offline solver ran on an M1 Pro chip-set that has 8-core CPU with a maximum utilization of 31,4 %. Memory utilization varied and it did not exceed 3 GBs for the cases tested. Moreover,it should be noted that for the offline setting the algorithm formulation was reproduced and solver using the SCIP solver which is provided by Google OR-Tools, an open source software suite for LP,MIP and ILP optimization. In the practical results we will only examine the offline algorithm case and see how it performs with the help of figures.

6.2 Test instances

During my experiments the biggest contribution on raising the cost was the placement of unavailable intervals in a manner of not allowing the transmission of the image set.Images can also play a key role in the cost but the unavailabilities can scale up the cost more than large image sizes

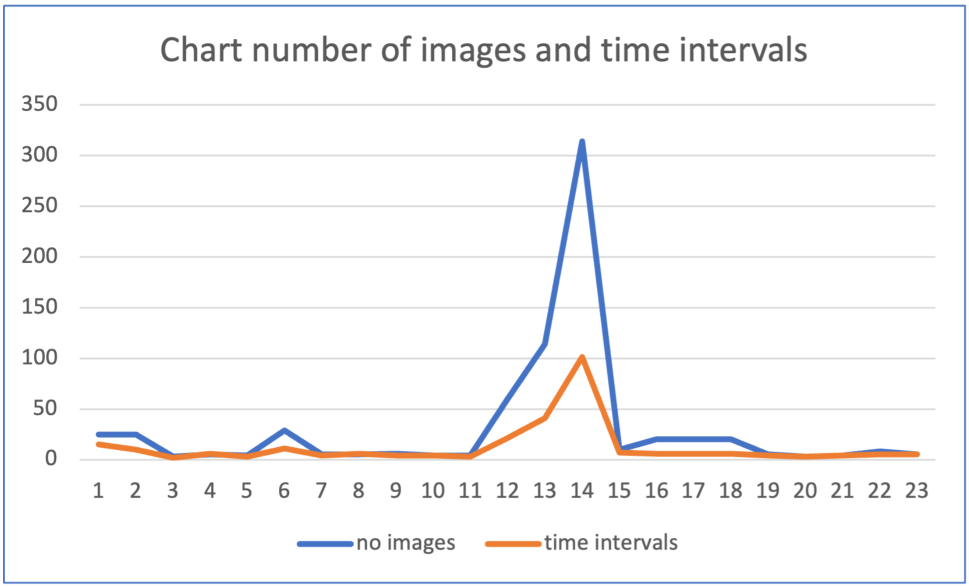

Figure 1 The number of images doesn't affect to a big degree the intervals

On the other hand the number of unavailable intervals explicitly defines the available space size in which images can be transmitted.

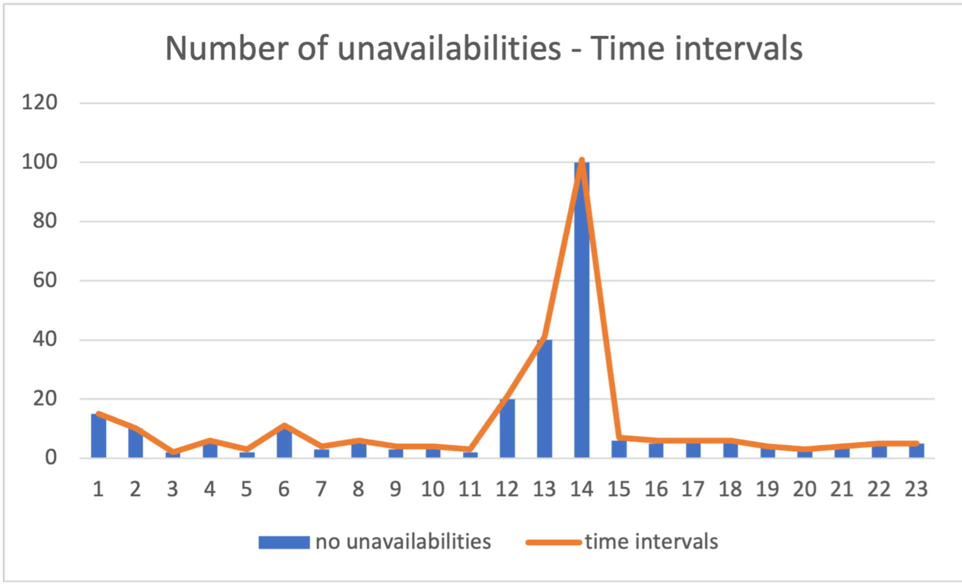

Figure 2 Chart showing the correlation between number of unavailabilities and time intervals

The tests performed on the offline solver ranged from having intervals that just fit the images tightly to others that stress the algorithm with the choice it has to make as it shouldn't act greedily.

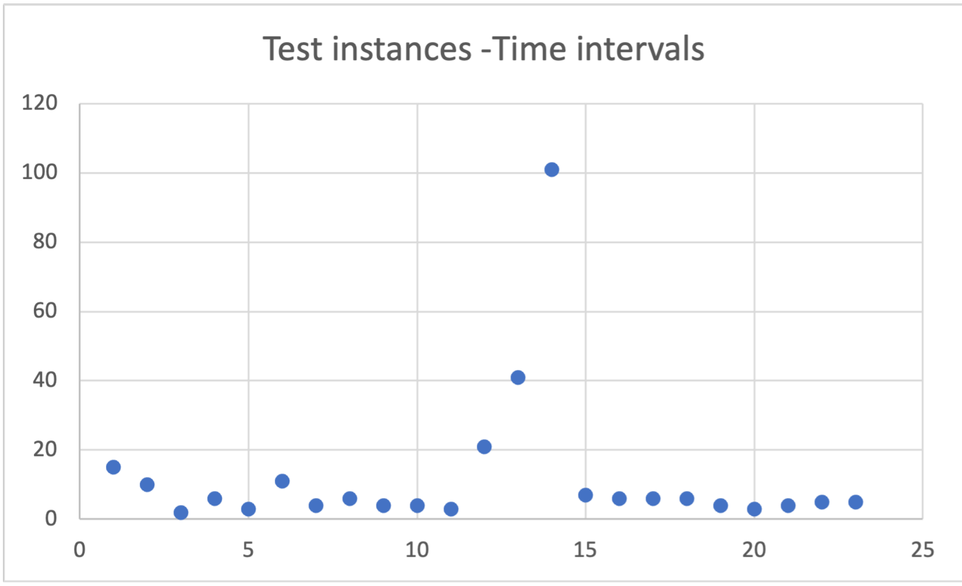

Figure 3 Scatter showing the range of time intervals created by test instances ran on the offline solver

Below it is shown how the offline algorithm performed in the test instances.By cross validating the results the offline solver always finds the optimal even in situations that are constructed to delay the transmission of images as much as possible.

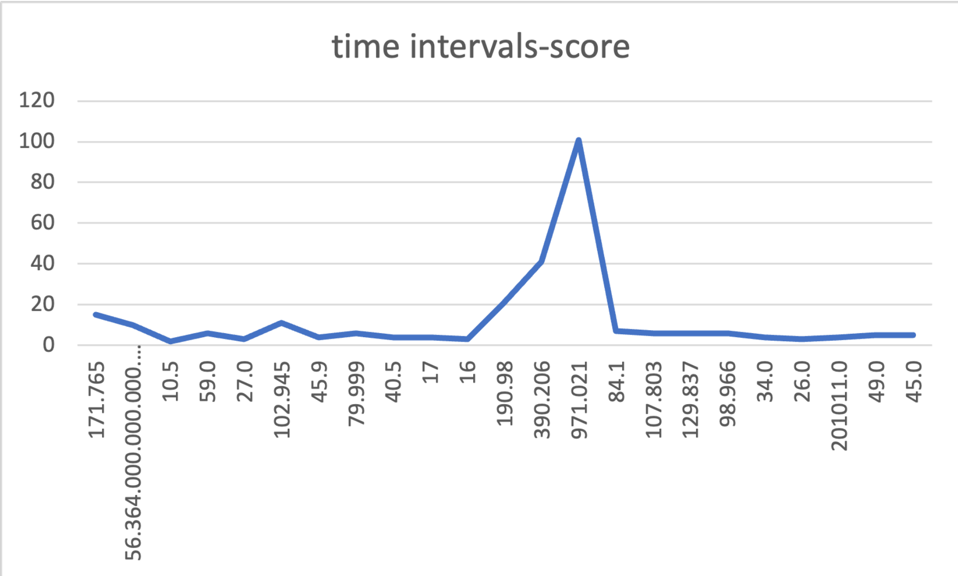

Figure 4 The number of time intervals compared to the total score.The correlation between them is not so high since the size of images can simply raise the cost even with few unavailabilities

6.2.1 Final remarks

The implementation of the offline algorithm compared to the test cases proved its validity to produce optimal results by always maximizing the packing of the images in the time intervals but excluding the last interval after all unavailable time intervals.Even in the most adversarial test cases the offline setting chose to place images in a non-greedy manner by sometimes sacrificing tight packing to achieve a better minimum overall cost.

7 Conclusion

In this chapter we researched about the offline Telescope scheduling problem and how the time intervals are correlated with the number of images and the total cost.The algorithm was tested from the instances submitted by the fellow colleagues and they vary from simple ones to ones that are adversarial for the algorithm .The complexity of the solution escalates when there are many unavailable intervals in combination with image sizes that need to be packed tightly.Additionally, the offline instance must not act like a greedy algorithm would,but instead calculate the best fit of items to result in the minimal cost.A 2-competitive algorithm was showcased for the case of one unavailable time interval combining it with a tight analysis.Finally, for the general case a lower bound of $\Omega(3/2)$ was found.For future, work the offline algorithm could be compared with an online algorithm that uses a heuristic for permutating the images so they could be transmitted faster thus reducing the cost and the competitive ratio.

References

[1] Jon Kleinberg. “Approximation Algorithms”. In: *Cornell University* 1.1 (2006), p. 464.

[2] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC ’71. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158. ISBN: 9781450374644. DOI: `10.1145/800157.805047`. URL: `https://doi.org/10.1145/800157.805047`.

[3] Jan van Leeuwen. “Algorithms and Complexity”. In: *Handbook of Theoretical Computer Science* 2.1 (1998).

[4] V. J. Rayward-Smith. “A First Course in Computability”. In: *Handbook of Theoretical Computer Science* (1986), p. 159.

[5] Richard M Karp. “Reducibility among combinatorial problems”. In: *In Complexity of computer 590 computations* 1.1 (1972), pp. 80–105.

[6] David S Johnson. “The NP-completeness column: An ongoing guide”. In: *Journal of Algorithms* 3.3 (1982), pp. 288–300. ISSN: 0196-6774. DOI: `https://doi.org/10.1016/0196-6774(82)90026-8`. URL: `https://www.sciencedirect.com/science/article/pii/0196677482900268`.

[7] Bhuvaneshwaran Ravi, Kameswaran Rangasamy and Serlin Tamilselvam. “BIN PACKING Proof of NP Completeness and Hardness”. In: *Special/Advanced Topics in Algorithms* 1.1 (2020), pp. 1–3.

[8] Alison Liu. *NP-Completeness*. University Lecture. 2022.

[9] Abdolahad Zehmakan. “Bin Packing Problem: Two Approximation Algorithms”. In: *International Journal in Foundations of Computer Science Technology* 5 (Aug. 2015). DOI: `10.5121/ijfcst.2015.5401`.

[10] Subhash Suri. “Approximation Algorithms”. In: *cs130b* 1.1 (2017), pp. 1–8.