A polynomial-time approximation scheme for the airplane refueling problem

Iftah Gamzu · Danny Segev

Published online: 16 May 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
We consider the airplane refueling problem that was introduced by Gamow and Stern in their classical book *Puzzle-Math* (1958). In this setting, we wish to deliver a bomb the farthest possible distance, being much greater than the range of any individual airplane at our disposal. For this purpose, the only feasible option is to better utilize our fleet via mid-air refueling. Starting with a fleet of airplanes that can instantaneously refuel one another and gradually drop out of formation, how would we design the best refueling policy, i.e., one that maximizes the distance traveled by the last remaining plane? Even though Gamow and Stern provided an elegant characterization of the optimal refueling policy for the special case of identical airplanes, the general problem with arbitrary tank volumes and consumption rates has remained widely open, as pointed out by Woeginger (Albers et al., Dagstuhl seminar proceedings 10071, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2010). To our knowledge, other than a logarithmic approximation, which can be attributed to folklore, improved performance guarantees have not been obtained to date. In this paper, we propose a polynomial-time approximation scheme for the airplane refueling problem in its utmost generality. Our approach employs widely-known techniques related to geometric rounding, time stretching, guessing arguments, and timeline partitions. These are augmented by additional insight and ideas, that enable us to devise reductions to well-structured instances of generalized assignment and to exploit LP-rounding algorithms for the latter problem. We complement this result by presenting a fast and easy-to-implement algorithm that attains a constant factor approximation for the optimal refueling policy.

Keywords Scheduling · Approximation algorithms · PTAS · Generalized assignment

1 Introduction

We study the airplane refueling problem that was introduced in 1958 by the physicists George Gamow and Marvin Stern in the Aeronautics chapter of their classical book *Puzzle-Math* (Gamow and Stern 1958). Sticking to the original story behind this problem, suppose we have a given fleet of airplanes, with the goal of delivering a bomb in some distant point of the globe. However, the distance to the target is much greater than the range of any individual plane, and the only feasible option to carry out this mission is to better utilize our fleet via mid-air refueling. Starting with several planes that can refuel one another, and gradually drop out of formation until the single plane carrying the bomb reaches the target, how would one plan the optimal refueling policy? For simplicity, it is assumed that fuel consumption is independent of the airplane load, and that refueling actions are completed instantaneously.

Problem definition Formally, an instance of the airplane refueling problem consists of a fleet of $n$ airplanes, to which we refer as $A_1, \ldots, A_n$. Each plane $A_j$ is characterized by two basic attributes: (1) A tank volume of $v_j$ measured, say, in gallons; and (2) A fuel consumption rate of $c_j$ measured in gallons per mile. So, without any refueling actions, this plane can travel $v_j/c_j$ miles by itself. The objective is to decide on the order by which planes drop out, and on how the remaining fuel is distributed at that time, trying to maximize the overall distance travelled by the last plane remaining.

To express the objective function, one can consider the fleet of airplanes as a single entity whose momentary fuel
consumption rate is given by the sum of individual fuel consumption rates of all active planes. Namely, if \( \sigma(1), \ldots, \sigma(n) \) stands for the drop out permutation, then the tank volume \( v_{\sigma(j)} \) of plane \( A_{\sigma(j)} \) could be thought of as being consumed by planes \( A_{\sigma(1)}, \ldots, A_{\sigma(n)} \) at their combined consumption rate, \( \sum_{k=1}^{n} c_{\sigma(k)} \). One can easily observe that, in an optimal policy, \( A_{\sigma(j)} \) drops out as soon as the total amount of fuel retained by the remaining fleet equals the overall capacity of all currently active planes excluding \( A_{\sigma(j)} \). Based on this observation, the problem is actually that of deciding on the order of plane dropouts. Specifically, the objective can be restated as computing a permutation \( \sigma \) that maximizes the distance travelled by the last airplane, \( \sum_{j=1}^{n} (v_{\sigma(j)}/ \sum_{k=1}^{n} c_{\sigma(k)}) \).

For simplicity, we focus on a modified objective function, which is completely equivalent to the one above. Our alternative definition considers the original permutation \( \sigma \) in reverse order. That is, we are interested in computing a drop out permutation \( \pi \) that maximizes

\[
\text{dist}(\pi) = \frac{\sum_{j=1}^{n} v_{\pi(j)}}{\sum_{k=1}^{n} c_{\pi(k)}},
\]

noting that one can recover the original formulation by replacing each \( \pi(j) \) with \( \sigma(n-j+1) \). We also assume without loss of generality that \( c_{\min} = \min_j c_j = 1 \); otherwise, all \( v_j \)'s and \( c_j \)'s can be normalized by \( c_{\min} \), leaving the objective value unchanged. Finally, we let \( C = \sum_{j=1}^{n} c_j \), and designate the inner term in the modified objective function by \( \text{dist}_j(\pi) = v_{\pi(j)}/ \sum_{k=1}^{n} c_{\pi(k)} \), which is referred to as the distance contribution of airplane \( A_{\pi(j)} \). Using this notation, we show in Sect. 1.2 that the airplane refueling problem can be thought of as a single-machine scheduling problem in disguise, where one wishes to maximize the weighted sum of inverse completion times.

Computational status As previously mentioned, the airplane refueling problem was introduced by Gamow and Stern in their book *Puzzle-Math* (Gamow and Stern 1958), which ignited numerous lines of research in areas such as fairness considerations in resource allocation (Thomson 2011; Procaccia 2013), probabilistic reasoning (Knuth 1969; Wuffe 1982; De and Sen 1996), and epistemic logic (Baltag et al. 2008; Ditmarsch et al. 2008), just to mention a few examples. The main contribution of Gamow and Stern was to provide a complete characterization of the optimal refueling policy for the special case of identical airplanes. In this case, all tank volumes are equal to \( V \) and all consumption rates are normalized to 1. As the flight proceeds, when the amount of fuel left in any one of the \( n \) planes reaches \((1 - 1/n) \cdot V\), there is just enough fuel to completely refuel \( n - 1 \) planes. At this moment, the entire fuel tank of one of the airplanes is used to refuel the remaining planes, this plane drops out, while the others proceed with a full tank. The next drop out occurs when the fuel tank of each plane reaches \((1 - 1/(n-1)) \cdot V\), so on and so forth. Thus, the optimal distance to be travelled is \( H_n \cdot V \), where \( H_n = \sum_{k=1}^{n} (1/k) \) is the \( n \)-th harmonic number.

In spite of this elegant and easy-to-analyze policy, the airplane refueling problem in its utmost generality, with arbitrary tank volumes and consumption rates, has remained widely open ever since, as recently pointed out by Woeginger (Albers et al. 2010). On the one hand, it appears as if there is very little structure to exploit in an attempt to establish any hardness result. On the other hand, it is not entirely obvious even if constant-factor performance guarantees are within reach, as the best-known algorithm prior to our work achieves an \( \Omega(1/\log n) \)-approximation ratio. For completeness, this algorithm and its analysis, which can be attributed to folklore, are described in Appendix B.

### 1.1 Our results

**Polynomial-time approximation scheme** The main contribution of this paper is to develop a polynomial-time approximation scheme (PTAS) for the airplane refueling problem in its utmost generality. That is, given an accuracy parameter \( \epsilon \in (0, 1) \), we show that the distance value attained by the optimal drop out permutation can be efficiently approximated within factor \( 1 - \epsilon \). As explained in Sect. 2, this result is achieved by synthesizing widely known techniques related to geometric rounding, time stretching, guessing arguments, and timeline partitions, along with additional insight and ideas, that allow us to propose reductions to well-structured instances of generalized assignment and to utilize LP-rounding algorithms in this context. To obtain a better understanding of the former techniques, we refer the reader to a selection of papers where these ideas have previously been explored and exploited (Hall et al. 1997; Afrati et al. 1999; Chuzhoy et al. 2006; Gandhi et al. 2008; Correa et al. 2012; Sitters 2014; Lübbecke et al. 2016). The core ideas of our approach are provided in Sects. 4 and 5, where we propose a pseudo-PTAS, whose running time is improved to a true PTAS in Sect. 6.

**Theorem 1** The airplane refueling problem admits a polynomial-time approximation scheme. The running time of our algorithm is \( \text{poly}(|I|) \cdot O(n^{O(1 + \log^{3/2} 1/\epsilon)}) \), where \( |I| \) stands for the given input size.

**Fast \( O(1) \)-approximation** Even though our main result progresses the current state of knowledge regarding the airplane refueling problem, the techniques involved may result in lengthy running times due to extensive guessing work. Therefore, we present a fast and easy-to-implement algorithm that approximates the airplane refueling problem to within
a constant factor. In addition to providing an efficient solution method, the marginal contribution of this algorithm is in serving as a brief prologue to the more technical PTAS that follows, allowing us to incrementally present our terminology, ideas, and algorithmic tools, without delving into complicated analysis. In particular, we draw crucial connections between the airplane refueling problem and the maximum generalized assignment problem that are exploited later on. Further details are given in Sect. 3.

Theorem 2 The airplane refueling problem can be approximated within factor 1/12 in time \( O(n \log C) \).

1.2 Related work

The application domain of the airplane refueling problem extends well beyond the original setting described by Gamow and Stern (1958). In particular, airplane refueling can be interpreted as a scheduling problem in disguise. From this perspective, \( n \) jobs should be scheduled on a single machine, where each job \( j \) has a value of \( v_j \) and a processing time of \( c_j \). Letting \( C_j \) stand for the completion time of job \( j \), the objective is to compute a non-preemptive schedule that maximizes the weighted sum of inverse completion times, \( \sum_{j=1}^{n} (v_j/C_j) \). This setting captures latency-sensitive scenarios in which the value gained from jobs diminishes over time. Such scenarios are motivated, for example, by applications arising in the context of network devices transmitting packets that bear time-dependent data, such as audio or video (Fiat et al. 2008; Feldman and Naor 2017).

Unlike our “dual” problem formulation of maximizing the weighted sum of inverse completion times, there has been a tremendous amount of work on scheduling problems where the objective is to minimize some function of the completion times. We refer the reader to the excellent survey by Chekuri and Khanna (2004), Höhn’s thesis (2014, Part 1), and the references therein for a more comprehensive literature review. That said, a particularly relevant line of work considers minimizing the objective \( \sum_{j=1}^{n} f_j(C_j) \), where each \( f_j \) is an arbitrary nonnegative non-decreasing function. The first constant-factor approximation for this setting was developed by Bansal and Pruhs (2014), who showed how to obtain a schedule whose cost is within factor 16 of optimal. This ratio was subsequently improved to \( 4 + \epsilon \) by Cheung et al. (2017), whereas an \((\epsilon + \epsilon)\)-approximation in quasi-polynomial time has been attained by Höhn et al. (2014). For the special case where \( f_j(C_j) = v_j f(C_j) \), Megow and Verschae (2013) devised a PTAS. Later on, Höhn (2014, Section 1.1) observed that any instance of the airplane refueling problem can be rephrased in terms of this minimization problem by replacing the term \( 1/C_j \) in the objective function with \( 1 - 1/C_j \). Unfortunately, this transformation does not preserve approximation guarantees, and we are not aware of any way to utilize the above-mentioned results for approximating the airplane refueling problem.

As hinted earlier, an important ingredient of our approach consists of constructing near-optimal solutions for well-structured instances of the maximum generalized assignment problem. In this setting, we are given a set of knapsacks, each of which has a capacity constraint, and a set of items that have possibly different size and value when assigned to each knapsack. The goal is to pack a maximum-value subset of items into the knapsacks while respecting their capacity constraints. Shmoys and Tardos (1993) seem to have been the first to (implicitly) study this problem. They presented an LP-based algorithm for the minimization variant, which was shown by Chekuri and Khanna (2005) to provide a \( 1/2 \)-approximation for the maximization variant, with small modifications. Chekuri and Khanna also identified several APX-hard special cases of generalized assignment. Fleischer et al. (2011) considered the separable assignment problem, which extends generalized assignment. In particular, they developed an LP-based \((1-1/e)\)-approximation for the latter problem. Feige and Vondrak (2010) proved that the \( 1-1/e \) factor is suboptimal by proposing an LP-based algorithm that attains an approximation factor of \( 1-1/e + \epsilon \), for some absolute constant \( \epsilon > 0 \). Additional papers in this context, with closely related variants, are those of Cohen et al. (2006), and Nutov et al. (2006).

2 Informal overview

Attaining constant performance guarantees We identify a number of inherent connections between the airplane refueling problem and maximum generalized assignment, that can be leveraged to obtain an elegant \( \Omega(1) \)-approximation for the former problem. The high-level idea is to consider the sequence of cumulative sums of consumption rates induced by the optimal permutation \( \pi^* \):

\[
c_{\pi^*(1)}, \ (c_{\pi^*(1)} + c_{\pi^*(2)}), \ (c_{\pi^*(1)} + c_{\pi^*(2)} + c_{\pi^*(3)}), \ldots, \ (c_{\pi^*(1)} + \cdots + c_{\pi^*(n)})
\]

This sequence can be viewed as an increasing sequence of points on the timeline \([1,C]\). We partition this timeline geometrically by powers of 2 into a collection of disjoint buckets, \( I_1, I_2, \ldots, I_{\lfloor \log C \rfloor} \). With respect to this partition, we say that airplane \( A_j \) is fully packed in bucket \( I_i \) when, letting \( \ell = \pi^* - 1(j) \), the points \( c_{\pi^*(1)} + \cdots + c_{\pi^*(\ell-1)} \) and \( c_{\pi^*(1)} + \cdots + c_{\pi^*(\ell)} \) are both located in \( I_i \). Given this perspective, there are two crucial observations:

1. Suppose that airplane \( A_j \) is fully packed in bucket \( I_i = [\alpha, 2\alpha) \). Then, the contribution \( \text{dist}_{\pi^* - 1(j)}(\pi^*) \) of \( A_j \) to
the optimal objective value is bounded between \( v_j/(2\alpha) \)
and \( v_j/\alpha \). Hence, if we are willing to lose a factor of 2, the
exact position of plane \( A_j \) in the permutation becomes
immaterial as long as we pack it in \( I_j \). This observation
motivates a reduction of our problem to an instance of the
maximum generalized assignment problem, for which
constant-factor approximations are known (see Sect. 1.2).

2. Unfortunately, there could be quite a few airplanes that
are not fully packed in any bucket. To bypass this obsta-
cle, we argue in Sect. 3 that there is a way of padding
the buckets so that every airplane is fully packed in some
bucket, making the above-mentioned reduction applicable.
In particular, we explain why a small amount of
padding suffices, and why the additional loss is only a
small constant.

\section*{Improvements to obtain a PTAS}

In Sects. 4–6, our efforts are devoted to proving that all components of the above
algorithm entailing some constant loss in optimality can be
“fixed” to efficiently obtain a near-optimal drop out permuta-
tion. Specifically, we show that each of these components can
be replaced by an improved mechanism that, despite being
significantly more involved, leads to an \( \epsilon \)-loss rather than to an
\( \Omega(1) \)-loss. Moreover, we explain how to combine them
without incurring extra cost. In what follows, we highlight
some of these components and explain how to fix them at a
very high level:

1. \textit{Better geometric jumps} It seems natural to improve our
initial approach by geometrically partitioning the time-
line into buckets by powers of \( 1+\epsilon \) instead of 2. However,
after inspecting the corresponding analysis, one realizes
that the jump size affects the amount of padding needed
to ensure that all airplanes can be packed into buckets,
i.e., smaller jumps mean more padding. For this reason,
the first technical idea is to sharpen the initial padding
arguments, so that they are not significantly affected by
jump sizes, allowing us to employ jumps by powers of
\( 1+\epsilon \).

2. \textit{Refined timeline partition} To avoid over-padding, the sec-
ond technical idea is to refine the geometric partition by
forming additional buckets, meant to capture airplanes
that cross two or more original buckets. This, in turn,
introduces a couple of inherent difficulties: (a) extensive
guesswork is needed in order to define the approximate
length and location of newly added buckets; and (b) as
new buckets are meant to capture a single crossing air-
plane, the underlying packing problem becomes more
involved. In this setting, some buckets can be used to pack
multiple airplanes, whereas others may pack at most one.

3. \textit{Maximum generalized assignment} Even special cases of
this problem are known to be APX-hard (Chekuri
and Khanna 2005), and without additional enhance-
ments, existing algorithms would lead to constant loss
in optimality. Thus, another technical idea is to prove that
slightly infeasible generalized assignment solutions
are good enough for the purpose of approximating the
airplane refueling problem. Nevertheless, the standard
practice of guessing “large assignments” does not seem
to be applicable here, and as a result, we are required to
devise more elaborated guessing methods.

4. \textit{Avoiding pseudo-polynomial time} Even though this fact
has not been mentioned yet, the length of our underlying
timeline \([1, C]\) is not necessarily polynomial in the num-
ber of airplanes, \( n \). Consequently, the resulting algorithm
is, up until now, a pseudo-PTAS. To obtain a true PTAS,
our final technical idea is to incorporate a random prepro-
cessing step, in which the original collection of airplanes
is divided into subsets, each defining an independent air-
plane refueling instance. With appropriate scaling, we
show that the timeline length of each instance is indeed
polynomial in the input size, and prove that the resulting
permutations can be merged into a single permutation
with a negligible loss in optimality.

\section*{3 A fast constant-factor approximation}

In what follows, we establish Theorem 2 by developing a
fast constant-factor approximation algorithm for the airplane
refueling problem. Technically speaking, our approach can
be broken down into two main steps. First, we reduce a
given instance of the problem to an instance of maximum
generalized assignment, arguing that a \( \rho \)-approximation for
generalized assignment implies a \( \rho/4 \)-approximation for our
problem. Then, we solve the resulting instance by essentially
any fast constant-factor approximation for the latter problem.

\subsection*{3.1 Structural alterations}

We begin by presenting an alteration of the optimal solution
so that it satisfies a number of helpful structural properties.
This step will simplify our reduction and its analysis later on.
Consider the sequence of cumulative sums of consumption
rates induced by the optimal permutation \( \pi^* \), that is,

\[ c_{\pi^*(1)} + (c_{\pi^*(1)}+c_{\pi^*(2)}) + (c_{\pi^*(1)}+c_{\pi^*(2)}+c_{\pi^*(3)}) + \ldots + c_{\pi^*(n)} \]

This increasing sequence can be viewed as a collection of
points on a timeline. Recall that \( C = \sum_{j=1}^n c_j \), and observe
that these points are all contained in \([1, C]\), by the assump-
tion that \(c_{\min} = 1\). We partition this timeline geometrically by powers of 2 into a collection of \(O(\log C)\) disjoint buckets, \(I_0, I_1, \ldots, I_{[\log C]}\). Specifically, the first bucket \(I_0\) spans \([0, 2^0)\), the second one \(I_1\) spans \([2^0, 2^1)\), then \(I_2\) spans \([2^1, 2^2)\), so on and so forth; in general, bucket \(I_2\) spans \([2^1, 2^2)\).

We define an assignment function \(\psi: \{A_1, \ldots, A_n\} \to \{I_1, \ldots, I_{[\log C]}\}\) from the set of airplanes to the above collection of buckets, according to the relative location of each airplane within the optimal permutation \(\pi^*\). Formally, the span of airplane \(A_{\pi^*(j)}\), located at the \(j\)-th position of \(\pi^*\), is defined as the interval \((\sum_{k=1}^{j-1} c_{\pi^*(k)}, \sum_{k=1}^{j} c_{\pi^*(k)})\), of length \(c_{\pi^*(j)}\). With this definition, \(\psi\) assigns each airplane \(A_j\) to the maximal index bucket intersected by its span, namely, \(\psi(A_j) = I_i\) for the maximal index \(i\) with \(\text{span}(A_j) \cap I_i \neq \emptyset\).

It is worth noting that no airplane is assigned to \(I_0\), since \(c_{\min} = 1\).

An important property of the assignment \(\psi\), which is stated below, is that the combined interval lengths over all airplanes \(\psi^{-1}(I_i)\) assigned to each bucket \(I_i = [2^{i-1}, 2^i)\) is at most twice the size of this bucket. To see this, note that these airplane intervals are pairwise-disjoint, each fully contained in \([0, 2^i)\), or otherwise, it would have been assigned by \(\psi\) to a higher-index bucket.

**Observation 1** For every bucket \(I_i\),

\[
\sum_{A_j \in \psi^{-1}(I_i)} c_j \leq 2^i.
\]

Having this assignment in mind, we proceed by defining a modified permutation \(\tilde{\pi}^*\) for our instance. We first pad all original buckets \(I_1, \ldots, I_{[\log C]}\) by doubling their size, to obtain \(\tilde{I}_1, \ldots, I_{[\log C]}\). Here, \(\tilde{I}_1 = [2^1, 2^2)\), \(\tilde{I}_2 = [2^2, 2^3)\), and so on. Based on the assignment \(\psi\), airplane intervals are now placed within their respective padded buckets, wherein each one, the intervals are ordered arbitrarily by \(\tilde{\pi}^*\). That is, if \(\psi(A_j) = I_i\), then airplane \(A_j\) is placed in \(\tilde{I}_i\). We emphasize that, by Observation 1, each target bucket can indeed accommodate all its assigned airplane intervals since we have just doubled its size. The modified permutation \(\tilde{\pi}^*\) is then obtained by concatenating the permutations generated for the buckets \(\tilde{I}_1, \tilde{I}_2, \ldots, I_{[\log C]}\) in this order. The next lemma shows that this modification degrades the resulting solution by only a constant factor, and specifically, the distance value of \(\tilde{\pi}^*\) approximates that of \(\pi^*\) within factor 4.

**Lemma 1** \(\text{dist}(\tilde{\pi}^*) \geq \text{dist}(\pi^*)/4\).

**Proof** To prove the desired claim, we show that the distance contribution of each airplane with respect to \(\tilde{\pi}^*\) is at least \(1/4\) of its contribution with respect to \(\pi^*\). For this purpose, consider airplane \(A_{\pi^*(j)}\), located at the \(j\)-th position of the optimal permutation \(\pi^*\), and suppose it is assigned to bucket \(I_i = \psi(A_{\pi^*(j)})\); note that \(i \geq 1\), since no airplane is assigned to \(I_0\). Thus,

\[
\text{dist}_j(\pi^*) = \frac{v_{\pi^*(j)}}{\sum_{k=1}^{j} c_{\pi^*(k)}} \leq \frac{v_{\pi^*(j)}}{2^{i-1}},
\]

since bucket \(I_i\) spans \([2^{i-1}, 2^i)\). On the other hand, the corresponding padded bucket \(\tilde{I}_i\) is precisely \([2^i, 2^{i+1})\). As a result, since the modified solution places \(A_{\pi^*(j)}\) in \(\tilde{I}_i\), its distance contribution to \(\tilde{\pi}^*\) (being at position \(\tilde{\pi}^{i-1}(\pi^*(j))\) of this permutation) is

\[
\text{dist}_{\tilde{\pi}^{i-1}(\pi^*(j))}(\tilde{\pi}^*) \geq \frac{v_{\pi^*(j)}}{2^{i+1}} \geq \frac{1}{4} \cdot \text{dist}_j(\pi^*).
\]

□

As a side note, it is worth pointing out that the buckets and interval assignments can be defined more carefully, so that similar structural properties are satisfied, with a value loss better than \(1/4\). However, for ease of exposition, we are not attempting to optimize constants in this section.

### 3.2 The algorithm

**Step 1: Reduction to maximum generalized assignment**

We are now ready to finalize our reduction to the maximum generalized assignment problem:

- There is a bipartite graph, with \(n\) (airplane interval) items on one side and \([\log C]\) (padded bucket) knapsacks on the other side. Every item \(j\) has a size of \(c_j\), and each knapsack \(i\) has a capacity of \(2^i\), corresponding to the length of the padded bucket \(\tilde{I}_i\).
- There is an edge between every item \(j\) and knapsack \(i\) for which that assignment is feasible (namely, \(c_j \leq 2^i\)), with value \(v_j/2^{i+1}\). Recall that \(2^{i+1}\) is the upper endpoint of bucket \(\tilde{I}_i\).

Some important remarks about this reduction are in place. First, the reduction can easily be implemented in \(O(n \log C)\) time. Second, any feasible solution for the reduced generalized assignment instance implies a feasible permutation for the original airplane refueling instance with distance value at least as good. This permutation can be generated in \(O(n + \log C)\) time, similar to the way the modified solution above was generated, namely, we generate an arbitrary ordering of the (airplane interval) items assigned to each (bucket) knapsack, and then concatenate those orderings according to increasing knapsack indices. Finally, notice that Lemma 1, along with our construction of the generalized assignment instance, guarantees that the optimum value of the latter instance is at least \(1/4\) of the optimal distance value, \(\text{dist}(\pi^*)\).
Step 2: Solving the generalized assignment instance Clearly, we can utilize any approximation algorithm for maximum generalized assignment to solve our reduced instance. Nevertheless, one should take into account the trade-offs between the resulting approximation guarantees of these algorithms and their running time, as explained below.

3.3 Concluding the proof of Theorem 2

We are now ready to establish Theorem 2, noting again that the emphasis of this section is more on developing a fast and easy-to-implement algorithm, rather than on optimizing constants. Our algorithm begins by reducing an airplane refueling instance to an instance of the maximum generalized assignment problem. This can done in \(O(n \log C)\) time, as discussed earlier. Cohen et al. (2006) devised a 1/3-approximation for generalized assignment that runs in \(O(NM)\) time, where \(N\) is the number of items and \(M\) is the number of knapsacks. As a result, we obtain a solution for the generalized assignment instance in \(O(n \log C)\) time. The value of this solution is at least 1/12 times the optimal distance value, due to using the 1/3-approximation for generalized assignment, and due to losing an additional factor of 1/4 during the reduction (see Lemma 1). This solution is then converted to a permutation for the airplane refueling instance with at least the same value in additional \(O(n + \log C)\) time.

4 PTAS phase 1: reduction to maximum generalized assignment

We now turn our attention to establishing Theorem 1, stating that the optimal drop out permutation for the airplane refueling problem can be efficiently approximated to within any degree of accuracy. To this end, we show that every component of the algorithm proposed in Sect. 3, entailing some constant loss in optimality, can be efficiently “fixed” to obtain a near-optimal solution, and that all those components can be glued together into an approximation scheme. To simplify the presentation, we initially concentrate on devising a pseudo-PTAS in Sects. 4 and 5, whose running time is \(\text{poly}(|\mathcal{I}|) \cdot O(C^{\frac{1}{2} \log^2 \frac{1}{\Delta}})\), where \(|\mathcal{I}|\) stands for the input size of a given instance. In Sect. 6, we describe the additional improvements needed to convert this algorithm to a true PTAS, where the dependency on \(C\) is eliminated.

4.1 Preliminary partitioning into buckets

Where does 1/4 come from? When one inspects the analysis of our constant-factor approximation, it becomes apparent that the reduction described in Sects. 3.1 and 3.2 actually loses a multiplicative factor of 1/2 twice. First, we used the upper endpoint of each bucket’s segment for bounding the distance contribution of all airplanes assigned to that bucket; these segments were geometrically increasing by powers of 2. Second, we doubled the size of each bucket so that it could accommodate all of its assigned airplane intervals. Unfortunately, there is a trade-off between the bucket jump sizes and the amount of padding needed to ensure that all airplanes intervals can be packed into the stretched buckets. In fact, it is not difficult to verify that \((1 + \epsilon)\)-jumps require stretching each bucket by a factor of \(\Omega(1/\epsilon)\), meaning that we cannot simultaneously decrease both sources of optimality loss.

4.2 Refinement to single-item and multi-item buckets

Guessing the position of single-item buckets To better understand the subsequent discussion, we advise the reader to consult Fig. 1a. In what follows, we assume without loss of generality that \(1/\epsilon\) takes an integer value. For the purpose of defining single-item buckets, we proceed by guessing the approximate position of all bucket-crossing airplane intervals in the optimal permutation \(\pi^*\). Clearly, the lower endpoint of each such interval is located in some bucket, whereas its upper endpoint is located in a higher-index bucket (not necessarily successive). For \(i \geq 1\), we divide each bucket \(I_i\) to \(1/\epsilon\) equal-length parts. Formally, letting \(\Delta(I_i) = (1+\epsilon)^i - (1+\epsilon)^{i-1}\) denote the length of bucket \(I_i\), we partition \(I_i\) into \(1/\epsilon\) parts, each of length \(\epsilon \cdot \Delta(I_i)\). This way, bucket \(I_i\) is associated for purposes of analysis with two values:

1. The \(\epsilon \cdot \Delta(I_i)\)-sized part containing the lower endpoint of a bucket-crossing interval starting in \(I_i\) (and extending to a higher-index bucket). We refer to the lower endpoint of this \(\epsilon \cdot \Delta(I_i)\)-sized part as \(s^*_i\).
2. The $\epsilon \cdot \Delta(I_i)$-sized part containing the upper endpoint of
a bucket-crossing interval ending in $I_i$ (and extending to
a lower-index bucket). We refer to the upper endpoint of
this $\epsilon \cdot \Delta(I_i)$-sized part as $s^+_i$.

Clearly, there may be buckets that contain only one endpoint
of a bucket-crossing interval, or contain no such endpoints
at all; these cases will also be treated below.

Now, for each bucket $I_i$, we guess $s^-_i$ and $s^+_i$ by means
of exhaustive enumeration. First, out of the $1/\epsilon$ lower end-
points of $\epsilon \cdot \Delta(I_i)$-sized parts, we guess which one serves
as $s^-_i$, or indicate by a separate value $\perp$ that $s^-_i$ is unde-
defined for this bucket. Second, out of the $1/\epsilon$ upper end-
points, we guess which one serves as $s^+_i$, again using $\perp$ to
indicate that $s^+_i$ is undefined. Finally, we guess whether or not
there is at least one airplane interval that is fully contained in
$I_i$. Note that since there are only three separate guesses for
each of the $O(1/\epsilon \log C)$ buckets, where the first two guesses
are picked out of $O(1/\epsilon)$ distinct values each, and the third
guess takes true/false values, the overall number of tested
options is $O((1/\epsilon)^O(1/\epsilon \log C)) = O(C^{O(1/\epsilon \log 1/\epsilon)})$.

Adding single-item buckets Our guesses directly translate into
an approximate guess for the position and length of each
bucket-crossing airplane interval. In particular, since these
intervals are linearly ordered along $[0, C]$, when we guessed
$s^-_i$ in bucket $I_i$, the upper endpoint of the crossing airplane
interval that starts at $I_i$ corresponds to our next guess $s^+_i$
in a higher-index bucket $I_{i'}$. For any bucket indexed strictly
between $i$ and $i'$, our guesses would indicate $\perp$ for both $s^-_i$
and $s^+_i$, i.e., there are no endpoints at all. These pairs of $s^-_i$
and $s^+_i$, with $\perp$ filled in between, define the endpoints of our
single-item buckets, which are referred to as $J_1, \ldots, J_K$. For
convenience, we define an order $\preceq$ between these buckets,
according to their left-to-right position on the timeline, i.e.,
$J_1 \preceq \cdots \preceq J_K$.

We proceed by creating an adjusted version of each origi-
nal bucket $I_i$, and define multi-item bucket $\hat{I}_i$ to be the
remainder of bucket $I_i$ after eliminating the $\epsilon \cdot \Delta(I_i)$-sized
parts that have just become occupied by single-item buck-
ets. As before, the resulting length of $\hat{I}_i$ is denoted by $\Delta(\hat{I}_i)$.
These definitions, as well as subsequent ones, are illustrated in
Fig. 1.b. It is important to note that some multi-item buck-
ets $\hat{I}_i$ could end up with zero length. Nevertheless, we refer to
multi-item buckets $\hat{I}_i$ whose original version $I_i$ fully con-
tains at least one airplane interval as being active, and denote
their index set by $A \subseteq [L]$; note that our guessing pro-
dure also delivers the identity of $A$. Prior to finalizing the
reduction, we still need to make one crucial adjustment to
this partition. Here, a modified length of $\Delta(\hat{I}_i)$ is defined for
each active multi-item bucket $\hat{I}_i$, by introducing an additive
factor of $2\epsilon \cdot \Delta(I_i)$, that is, $\Delta(\hat{I}_i) = \Delta(I_i) + 2\epsilon \cdot \Delta(I_i)$.
Intuitively, the purpose of this adjustment is to compen-
sate for the length loss that occurred due to our guessing
procedure, where the endpoints of bucket-crossing intervals
were “rounded” to multiples of $\epsilon \cdot \Delta(I_i)$-sized parts. Lastly,
we align the position of all single-item buckets and active
multi-item buckets along the timeline, by ordering the buck-
ets according to their original relative positions, and extend
the order $\preceq$ to the combined set of buckets. Note that the
extended order is indeed well-defined, due to disregarding
inactive multi-item buckets.

4.3 Constructing the generalized assignment
instance

We are now ready to describe how the resulting generalized
assignment instance is defined:

- There is a bipartite graph, with $n$ (airplane interval) items
  on one side and $K + |A| = O(1/\epsilon \log C)$ knapsacks on
  the other side, corresponding to the single-item buckets
  $J_1, \ldots, J_K$ and to the active multi-item buckets
  $\{\hat{I}_i\}_{i \in A}$.
- Every item $j$ has a size of $c_j$. For each single-item bucket
  $J_i$, its representative knapsack has a capacity of $\kappa(J_i) = \Delta(J_i)$.
  However, for each active multi-item bucket $\hat{I}_i$, its
  representative knapsack has a capacity of $\kappa(\hat{I}_i) = \Delta(\hat{I}_i)$.
- There is an edge between every item $j$ and knapsack
  $i$ for which that assignment is feasible (capacity-wise),
  with value $v_{j,i}$. Here, $c_j$ is the upper endpoint of
  the corresponding bucket, after the realignment on the time-
  line.
- Finally, knapsacks corresponding to single-item buckets
can contain at most one item, whereas those correspond-
ing to multi-item buckets are allowed to contain any
number of items. We explain in detail how to handle such
constraints in Sect. 5.

Lemma 2 There is a feasible solution to the resulting gen-
eralized assignment instance with objective value at least
$(1 - 9\epsilon) \cdot \text{dist}(\pi^*)$.

Proof Based on the optimal permutation $\pi^*$, we construct an
assignment function

$$\psi : \{A_1, \ldots, A_n\} \to \{J_1, \ldots, J_K\} \cup \{\hat{I}_i\}_{i \in A}$$

from the set of airplanes to single-item and active multi-
item buckets. This function will be shown to constitute a
feasible solution to the generalized assignment instance, and
moreover, to attain an objective value of at least $(1 - 9\epsilon) \cdot \text{dist}(\pi^*)$.

In order to define $\psi$, for each airplane $A_j$ we consider two
cases, depending on where its interval is located with respect
to the original bucket partition $I_0, \ldots, I_L$:
1. The interval of \( A_j \) is bucket-crossing. Here, this interval gives rise to a single-item bucket, say \( J_i \), and we set \( \psi(A_j) = J_i \). Notice that the capacity \( \kappa(J_i) = \Delta(J_i) \) of bucket \( J_i \) is indeed large enough to accommodate \( A_j \), as an immediate consequence of our guessing procedure (i.e., the lower endpoint of this bucket can only be smaller than the lower endpoint of \( A_j \)'s interval, and similarly, the upper one can only be larger that the upper endpoint). It is worth mentioning that, since each bucket-crossing interval corresponds to a unique single-item bucket, this capacity constraint remains satisfied when additional airplanes are assigned by \( \psi \).

2. The interval of \( A_j \) is fully contained in some bucket. In this case, if the latter bucket is \( I_i \), we set \( \psi(A_j) = I_i \), that is, airplane \( A_j \) is assigned to the multi-item bucket \( I_i \), which is necessarily active. The important observation is that the capacity \( \kappa(I_i) = \Delta(I_i) \) of bucket \( I_i \) is sufficiently large to accommodate its assigned set of airplanes \( \psi^{-1}(I_i) \). Indeed, while a capacity of \( \Delta(I_i) \) may be short by at most \( 2\epsilon \cdot \Delta(I_i) \) due to our guessing procedure for the endpoints of single-item buckets, the subsequent length adjustment guarantees that \( \Delta(I_i) = \Delta(\bar{I}_i) + 2\epsilon \cdot \Delta(I_i) \).

By the preceding discussion, we conclude that \( \psi \) is a feasible solution to the generalized assignment instance. To complete the proof, it suffices to focus on a single airplane \( A_{\pi^*(j)} \), located at the \( j \)-th position of the optimal permutation \( \pi^* \), and to argue that its value contribution with respect to the generalized assignment instance is at least \((1 - 9\epsilon) \cdot \text{dist}_j(\pi^*) \). We consider case 2 above, where airplane \( A_{\pi^*(j)} \) is assigned by \( \psi \) to an active multi-item bucket, \( I_i \); the analogous explanations for case 1 are nearly identical (and therefore omitted). Since the interval of \( A_{\pi^*(j)} \) is fully contained in the original bucket \( I_i = [(1 + \epsilon)^{i-1}, (1 + \epsilon)^i) \),

\[
\text{dist}_j(\pi^*) = \frac{v_{\pi^*}(j)}{\sum_{k=1}^j c_{\pi^*(k)}} \leq \frac{v_{\pi^*}(j)}{(1 + \epsilon)^{i-1}}.
\]

On the other hand, the assignment \( \psi(A_{\pi^*(j)}) = \bar{I}_i \) generates a value of \( v_{\pi^*}(j)/e_i \), where \( e_i \) is the upper endpoint of \( \bar{I}_i \). To derive an upper bound on the latter endpoint, recalling that \( \bar{I}_i \preceq \bar{I}_i \) indicates that single-item bucket \( J_i \) appears on the timeline to the left of \( \bar{I}_i \), we have

\[
e_i = \Delta(I_i) + \sum_{k=1}^i \Delta(I_k) + \sum_{J_i \preceq \bar{I}_i} \Delta(J_k) \\
\leq 1 + (1 + 4\epsilon) \cdot \sum_{k=1}^i \Delta(I_k) \\
= 1 + (1 + 4\epsilon) \cdot \sum_{k=1}^i \left((1 + \epsilon)^k - (1 + \epsilon)^{k-1}\right) \\
\leq (1 + 4\epsilon) \cdot (1 + \epsilon)^i.
\]

Here, the first inequality is a worst-case upper bound, where each bucket-crossing interval resides between two successive original intervals; this scenario makes our guessing procedure and length adjustments to incur an overall length increase in at most \( 1 + 4\epsilon \). As a result, the assignment \( \psi(A_{\pi^*(j)}) = \bar{I}_i \) has a value of \( v_{\pi^*}(j)/e_i \)

\[
\geq \frac{1}{(1 + 4\epsilon) \cdot (1 + \epsilon)^i} \cdot \frac{v_{\pi^*}(j)}{(1 + \epsilon)^{i-1}} \\
\geq (1 - 9\epsilon) \cdot \text{dist}_j(\pi^*).
\]

5 PTAS phase 2: approximating the generalized assignment instance

In what follows, we develop a pseudo-polynomial approximation scheme for the generalized assignment instance constructed in Sect. 4, that will be shown to over-pack certain knapsacks, thereby creating slightly infeasible solutions.
This result is derived by utilizing the special structural properties of the reduced instance, along with additional insight into the inner-workings of the Shmoys–Tardos LP-rounding algorithm (Shmoys and Tardos 1993). Furthermore, we argue toward the end of this section that such infeasible assignments can still be translated back to the original airplane refueling instance, losing only a negligible fraction of the objective value, thus proving the next theorem.

**Theorem 3** For any \( \epsilon \in (0, 1) \), the airplane refueling problem can be approximated within factor \( 1 - \epsilon \). Our algorithm runs in time \( \text{poly}(|I|) \cdot O(C^{O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon})}) \), where \(|I|\) stands for the input size of a given instance.

As previously mentioned, we describe in Sect. 6 the additional ingredients needed to convert this algorithm to a true PTAS, where the dependency on \( C \) is eliminated.

### 5.1 Size classes and the greedy assignment rule

**Creating size classes** Prior to presenting our algorithm, we slightly modify the generalized assignment instance that results from our reduction in Sect. 4, by introducing item size classes and by appropriately stretching knapsacks. This modification makes some future claims easier to establish while incurring a negligible loss in optimality. Let \( c_{\text{max}} = \max_j c_j \), and consider the partition of items into \( O(\frac{1}{\epsilon} \log c_{\text{max}}) \) size classes \( C_1, \ldots, C_R \) by powers of \( 1 + \epsilon \). Specifically, recalling that \( c_{\text{min}} = 1 \), the first class \( C_1 \) consists of all items whose size resides within \( [1, 1 + \epsilon) \), the second class \( C_2 \) corresponds to items with size in \( [1 + \epsilon, (1 + \epsilon)^2) \), and so on and so forth. With these definitions, we round up the size \( c_j \) of each item \( j \) to the upper endpoint of the size class in which it resides.

At the moment, this modification leads to a potential infeasibility problem, since the assignment \( \psi \) we constructed during the proof of Lemma 2 could now exceed the capacity of some knapsacks. However, since each item blow-ups in size by a factor of at most \( 1 + \epsilon \), bucket capacities are violated by at most this factor as well. Therefore, to restore feasibility, we increase the capacity of each knapsack by a factor of \( 1 + \epsilon \), and denote these stretched capacities by \( \tilde{c} \). In addition, the value \( v_j/e_j \) gained from each item-to-knapsack assignment (see Sect. 4.3) is adjusted by plugging-in the new upper endpoint of this knapsack. We emphasize that, since the upper endpoint of each knapsack increases by a factor of \( 1 + \epsilon \), the value of the resulting solution may degrade by at most \( 1 + \epsilon \).

**The greedy assignment rule** Now let \( \psi^* \) be an optimal solution to the modified generalized assignment instance. The crucial observation is that, for every size class \( C_r \), looking at the underlying buckets by their left-to-right order \( \preceq \) on the timeline, the items in \( C_r \) meet along the way (as assigned by \( \psi^* \)) appear in non-increasing \( v_j \) values. This property, formally stated below, follows from an elementary swapping argument. Specifically, given two items \( j_1 \neq j_2 \) in class \( C_r \) with \( v_{j_1} > v_{j_2} \), if \( \psi^*(j_2) < \psi^*(j_1) \) then swapping between these items strictly increases the objective function while preserving bucket capacities (due to identical item sizes).

**Observation 2** (Greedy assignment rule) For every size class \( C_r \), and for every pair of items \( j_1 \neq j_2 \) in this class, if \( v_{j_1} > v_{j_2} \) then \( \psi^*(j_1) \preceq \psi^*(j_2) \).

Unfortunately, the greedy assignment rule is only partially constructive. That is, if we had known in advance the number of items each knapsack contains from each size class in the optimal solution \( \psi^* \), we could have greedily assigned the items of each class to the relevant knapsacks and completely recover \( \psi^* \). However, the number of guesses for these quantities is \( \prod_{r=1}^R |C_r|^{|A|} \cdot 2^K = \Omega(n^{O(\frac{1}{\epsilon} \log C \log c_{\text{max}})}) \), which is too large for our purposes.

In Sects. 5.2 and 5.3, we show how to test only \( f(\epsilon) = O(1/\epsilon)O(\frac{1}{\epsilon} \log \frac{1}{\epsilon}) \) options for each size class, and to obtain a well-structured partial assignment whose properties can be exploited within an LP-rounding algorithm, drawing on the ideas of Shmoys and Tardos (1993). Note that since there are only \( O(\frac{1}{\epsilon} \log c_{\text{max}}) = O(\frac{1}{\epsilon} \log C) \) size classes, the overall number of options to test in this step is

\[
O(f(\epsilon))O(\frac{1}{\epsilon} \log C) = O\left(C^{O(\frac{1}{\epsilon} \log f(\epsilon))}\right) = O\left(C^{O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon})}\right).
\]

### 5.2 Structural properties

Prior to delving into the specifics of our guessing procedure, we describe the structural properties of the resulting generalized assignment instance (see Sect. 4) that enable us to test only \( f(\epsilon) \) options per size class. To avoid deviating from the surrounding discussion, the next two claims are proven in Appendix A.

**Lemma 3** For every \( i \in A \), the capacity \( \tilde{c}(\tilde{I_i}) \) of active multi-item knapsack \( \tilde{I_i} \) is at most \( 3\epsilon \cdot (1 + \epsilon)^r \).

**Lemma 4** For every \( i \in A \), the overall capacity \( \tilde{c}(\cdot) \) of all knapsacks before (according to \( \preceq \)) and including active multi-item knapsack \( \tilde{I_i} \) is at most \( (1 + 4\epsilon) \cdot (1 + \epsilon)^{r+1} \).

### 5.3 The per-class guessing procedure

We focus on a single class \( C_r \), consisting of items whose size is \( (1 + \epsilon)^r \). With respect to this class, we partition the collection of knapsacks into three consecutive groups (each potentially empty) according to the order \( \preceq \) on their corresponding buckets on the timeline.

(1) \( C_r \)-small knapsacks. Let \( p = r + \lfloor \log_{1+\epsilon}(1/(3\epsilon)) \rfloor - 1 \). Notice that, by Lemma 3, each of the active multi-item knapsacks in \( \{\tilde{I_i}\}_{i \in A:r \leq p} \) has a capacity of at most \( 3\epsilon \cdot (1 + \epsilon)^p < \)}
items of class $C_r$ that are assigned by $\psi^*$ to the single-item knapsacks $\{I_i\}_{i \leq \mu_{\text{small}}}$. We guess the precise number of such items, which requires testing at most $f_{\text{small}}(\epsilon)$ options. Suppose that $t$ is our guess in this case, then we utilize the greedy assignment rule (see Observation 2) and temporarily reserve the $t$ items with the highest $v_j$ values in the size class $C_r$, without assigning them to any of the knapsacks yet.

(II) $C_r$-medium knapsacks. For this group, let $q = r + \lceil \log_{1+\epsilon}(1/(2\epsilon^3)) \rceil$ and let $\mu_{\text{medium}} = \max\{i \in \mathcal{A} : p < i \leq q\}$. We define the set of $C_r$-medium knapsacks as those that are not $C_r$-small and appear before and including active multi-item knapsack $I_{\mu_{\text{medium}}}$. By Lemma 4, the overall capacity of all $C_r$-medium knapsacks is at most

$$(1 + 4\epsilon) \cdot (1 + \epsilon)^{\mu_{\text{medium}}+1}$$

$$(1 + 4\epsilon) \cdot (1 + \epsilon)^{q+1}$$

$$(1 + 4\epsilon) \cdot (1 + \epsilon)^{\lceil \log_{1+\epsilon}(1/(2\epsilon^3)) \rceil + 1}$$

$$\leq \frac{2}{\epsilon} \cdot (1 + \epsilon)^r.$$

Consequently, the optimal assignment $\psi^*$ places at most $5/\epsilon^3$ items of class $C_r$ in the collection of $C_r$-medium knapsacks. Since the number of $C_r$-medium knapsacks is at most

$$2(q - p) = 2 \cdot \left[ \log_{1+\epsilon} \left( \frac{1}{2\epsilon^3} \right) \right] - \left[ \log_{1+\epsilon} \left( \frac{1}{3\epsilon} \right) \right] + 1$$

$$= O \left( \frac{1}{\epsilon} \log \frac{1}{\epsilon} \right),$$

guessing the number of items of class $C_r$ that are assigned by $\psi^*$ to each of these knapsacks amounts to at most

$$f_{\text{medium}}(\epsilon) = \left( \frac{5}{\epsilon^3} \right)^{2(q - p)} = O \left( \left( \frac{1}{\epsilon} \right)^{O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})} \right).$$

options. Given these guesses, we now make use of the greedy assignment rule. Specifically, we process the $C_r$-medium knapsacks according to the order $\leq$, and assign items from size class $C_r$ by non-increasing $v_j$’s, starting from the item with the $(t+1)$th highest $v_j$ value. This way, we completely duplicate $\psi^*$ between the size class $C_r$ and its related medium knapsacks, which guarantees in particular not to utilize any of the $t$ items that were reserved to $C_r$-small knapsacks (without being assigned yet).

(III) $C_r$-big knapsacks. To the remaining knapsacks, that are neither $C_r$-small or $C_r$-medium, we refer as being $C_r$-big, and do not make any guesses or assignments in advance. Still, an important observation is stated in Lemma 5 below, regarding the capacity $\tilde{k}(I_i)$ of any active multi-item knapsack $I_i$ that is $C_r$-big. The inherent claim is that the size $(1 + \epsilon)^r$ of any item in class $C_r$ forms only an $\epsilon$-fraction of the capacity $\tilde{k}(I_i)$.

Lemma 5 Suppose that active multi-item knapsack $I_i$ is $C_r$-big. Then, $(1 + \epsilon)^r \leq \epsilon \cdot \tilde{k}(I_i)$.

Proof In order to obtain the desired bound, note that

$$\tilde{k}(I_i) = (1 + \epsilon) \cdot \tilde{\Delta}(I_i)$$

$$\geq (1 + \epsilon) \cdot 2\epsilon \cdot \Delta(I_i)$$

$$= 2\epsilon \cdot ((1 + \epsilon)^{r+1} - (1 + \epsilon)^{r})$$

$$= 2\epsilon^2 \cdot (1 + \epsilon)^r.$$

Here, the first inequality follows from the definition of $\Delta(I_i) = \Delta(I_i) + 2\epsilon \cdot \Delta(I_i)$, the second equality holds since $I_i = [(1 + \epsilon)^{r-1}, (1 + \epsilon)^{r})$. Now, since $I_i$ is $C_r$-big, we must have $i > q$, or otherwise, $I_i$ would have been $C_r$-medium or $C_r$-small. Therefore,

$$\tilde{k}(I_i) > 2\epsilon^2 \cdot (1 + \epsilon)^q \geq \frac{1}{\epsilon} \cdot (1 + \epsilon)^r,$$

where the last inequality is obtained by substituting $q = r + \lceil \log_{1+\epsilon}(1/(2\epsilon^3)) \rceil$.

In summary, the preceding discussion proves that, as claimed at the end of Sect. 5.1, the total number of options to be tested for each size class can indeed be upper bounded by a function $f(\epsilon)$ of only $\epsilon$, since $f(\epsilon) \leq f_{\text{small}}(\epsilon) \cdot f_{\text{medium}}(\epsilon) = O((1/\epsilon)^{O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})}).$

5.4 Leveraging the Shmoys–Tardos algorithm

Shmoys and Tardos (1993) developed an LP-based algorithm, making use of parametric pruning to approximate the minimum generalized assignment problem. Even though this algorithm was originally designed for the minimization variant, Chekuri and Khanna (2005) showed that it can be adapted to the maximization variant with small modifications. Our approach is based on strengthening their LP-relaxation with:

- Item-to-knapsack assignment constraints, based on the guessing procedure described in Sect. 5.3.
– Extra constraints, ensuring that single-item knapsacks cannot be assigned multiple items.

In what follows, we present the details of this relaxation, for which the (integral) assignment \( \psi^* \) constitutes a feasible solution. We then explain how to round an optimal fractional solution to an integral assignment, while potentially overpacking certain knapsacks and thereby slightly exceeding their capacity. However, we conclude the analysis by arguing that such solutions can be translated back to the original airplane refueling instance, losing only a negligible fraction of the objective value.

The linear program Referring back to Sect. 4.3, we recall that our generalized assignment instance consists of \( n \) items and \( K + |\mathcal{A}| \) knapsacks. For convenience, we define a combined set of knapsacks \( \mathcal{K} \), defined as the union of single-item ones \( \{I_i, \ldots, J_K\} \) and multi-item ones \( \{I_{\mathcal{A}}\} \). Each item \( j \) has size \( c_j \), and each knapsack \( i \in \mathcal{K} \) has capacity \( \kappa(i) \). We denote the value \( v_j / \epsilon \) gained by packing item \( j \) in knapsack \( i \) as \( v_{ij} \). Furthermore, let \( M_j \) be the set of all active multi-item knapsacks that are \( c_{\log_{1+\epsilon} c_j} \)-medium, i.e., identified by item \( j \) (of size \( c_j \)) as being medium for its size class. Finally, let \( T_i \) be the set of items assigned to active multi-item knapsack \( I_i \) in our guessing step, noting that \( I_i \in M_j \), for every \( j \in T_i \), and let \( T = [n] \setminus \bigcup_{i \in \mathcal{K}} T_i \) be the set of items that have not been assigned yet. We consider the following linear program:

\[
\begin{align*}
(\text{LP}) \quad \text{max} & \quad \sum_{j \in [n]} \sum_{i \in \mathcal{K}} v_{ij} x_{ij} \\
(1) & \quad \sum_{i \in \mathcal{K}} x_{ij} = 1 \quad \forall j \in [n] \\
(2) & \quad \sum_{j \in [n]} c_j x_{ij} \leq \kappa(i) \quad \forall i \in \mathcal{K} \\
(3) & \quad \sum_{j \in [n]} x_{ij} \leq 1 \quad \forall i \in \{J_1, \ldots, J_K\} \\
(4) & \quad x_{ij} = 1 \quad \forall i \in \mathcal{A}, j \in T_i \\
(5) & \quad x_{ij} = 0 \quad \forall j \in T, i \in M_j \\
(6) & \quad x_{ij} = 0 \quad \forall j \in [n], i \in \mathcal{K} : c_j > \kappa(i) \\
(7) & \quad x_{ij} \geq 0 \quad \forall i \in \mathcal{K}, j \in [n]
\end{align*}
\]

In an integral solution, the variable \( x_{ij} \) indicates whether item \( j \) is packed in knapsack \( i \). Constraint (1) ensures that each item is packed in exactly one knapsack. Constraint (2) guarantees that the capacity of each knapsack is respected. Constraint (3) ensures that each single-item knapsack is assigned at most one item. Constraints (4) and (5) guarantee that all items guessed to reside in an active multi-item knapsack \( i \) are indeed assigned to it, and conversely, that the latter knapsack cannot be assigned any other item that identified \( i \) as being medium for its size class. Finally, constraint (6) prevents an item from being assigned to a knapsack where it does not fit by itself.

We emphasize again that the feasibility of (LP) is guaranteed by our construction and guessing procedure, which preserve the (integral) assignment \( \psi^* \) as a feasible solution. Moreover, as an immediate consequence of Lemma 2, it follows that \( \text{OPT}(\text{LP}) \geq (1 - 9\epsilon) \cdot \text{dist}(\pi^*) \), i.e., the optimal objective value of (LP) nearly matches the optimal distance value of the original airplane refueling instance.

The rounding procedure Given an optimal fractional solution \( x^* \) to the linear program (LP), we employ the deterministic rounding algorithm of Shmoys and Tardos (1993) to obtain an integral assignment. For our purposes, it is sufficient to mention that, for the minimization variant, their algorithm is based on interpreting \( x^* \) as a fractional degree-constrained matching of identical cost in an auxiliary bipartite graph. Here, the items \( [n] \) appear on one side and the knapsacks \( \mathcal{K} \) appear on the other, with an edge \((i, j)\) of weight \( v_{ij} \) between any item-knapsack pair with \( x^*_{ij} > 0 \). Moreover, each knapsack \( i \in \mathcal{K} \) is associated with a degree constraint, determined by rounding up the fractional number of items \( \sum_{j \in [n]} x^*_{ij} \) assigned to this knapsack, i.e., there is an upper bound of \( \lceil \sum_{j \in [n]} x^*_{ij} \rceil \) on the degree of \( i \). The important observation of Shmoys and Tardos (1993) is that, since the vertices of bipartite degree-constrained matching polytopes are integral, one can compute an integral matching (i.e., item-to-knapsack assignment \( \hat{x} \)) whose weight is at most \( \sum_{j \in [n]} \sum_{i \in \mathcal{K}} v_{ij} x^*_{ij} \). Alternatively, as noted by Chekuri and Khanna (2005), one can efficiently identify an integral assignment \( \hat{x} \) whose weight (or value) is at least \( \sum_{j \in [n]} \sum_{i \in \mathcal{K}} v_{ij} x^*_{ij} \) for the maximization variant.

That being said, since the degree constraint of each knapsack is defined by rounding up its fractional degree (with respect to \( x^* \)), the resulting assignment \( \hat{x} \) may violate the capacity constraint associated with each knapsack. Therefore, prior to proceeding with our algorithm, we briefly summarize the objective value guarantees of \( \hat{x} \) as well as its additional structural properties:

1. The assignment \( \hat{x} \) has an objective value of

\[
\sum_{j \in [n]} \sum_{i \in \mathcal{K}} v_{ij} \hat{x}_{ij} \geq \sum_{j \in [n]} \sum_{i \in \mathcal{K}} v_{ij} x^*_{ij}
= \text{OPT}(\text{LP}) \\
\geq (1 - 9\epsilon) \cdot \text{dist}(\pi^*)
\]

2. The number of items assigned to each knapsack \( i \) is \( \sum_{j \in [n]} \hat{x}_{ij} \geq \lceil \sum_{j \in [n]} x^*_{ij} \rceil \).
3. Each item \( j \) fits by itself into the knapsack \( i \) to which it is assigned, namely, if \( \hat{x}_{ij} = 1 \) then \( c_j \leq \kappa(i) \).
4. For every knapsack \( i \), if its capacity is violated (i.e., \( \sum_{j \in [n]} c_j \hat{x}_{ij} > \kappa(i) \)), there exists a single infeasibility item \( j_{\text{inf}(i)} \) with \( \hat{x}_{j_{\text{inf}(i)} i} = 1 \) whose removal restores the feasibility of that knapsack. Furthermore, this item will...
not be associated with an explicit assignment constraint \( x_{j \hat{u}(i)} = 1 \) of type (4).

**Converting \( \hat{x} \) to an airplane refueling solution** Rather than eliminating infeasibility items and potentially losing a constant factor of the objective value, we prove that the infeasible assignment \( \hat{x} \) actually translates back to the original airplane refueling instance, while only losing an \( O(\epsilon) \)-factor. This translation is created by first picking, for every knapsack \( i \in K \), an arbitrary permutation \( \hat{\pi}_{\{i\}} \) for the subset of airplane indices \( \{ j \in [n] : \hat{x}_{ij} = 1 \} \), corresponding to items that are assigned by \( \hat{x} \) to this knapsack. Then, the permutations \( \{ \hat{\pi}_{\{i\}} \}_{i \in K} \) are concatenated into a full permutation \( \hat{\pi} \) of all airplanes according to the knapsack order \( \preceq \) on the timeline. In other words, letting \( e_{i1} < e_{i2} < \cdots \) be the sorted sequence of the upper endpoints \( \{ e_i \}_{i \in K} \), the resulting permutation is \( \hat{\pi} = (\hat{\pi}_{\{i1\}}, \hat{\pi}_{\{i2\}}, \ldots) \).

### 5.5 Concluding the proof of Theorem 3

To complete the derivation of Theorem 3, we show that the permutation \( \hat{\pi} \) approximates the optimal distance value of the original airplane refueling instance within factor \( 1 - O(\epsilon) \), namely, \( \text{dist}(\hat{\pi}) = (1 - O(\epsilon)) \cdot \text{dist}(\pi^*) \). To this end, we first prove an auxiliary claim, stating that there is a very specific relation between knapsacks whose capacity is violated by the assignment \( \hat{x} \) and their infeasible items.

**Lemma 6** Suppose that \( \hat{x} \) exceeds the capacity of knapsack \( i \), i.e., \( \sum_{j \in [n]} c_j \hat{x}_{ij} > \hat{k}(i) \). Then,

1. Knapsack \( i \) must be an active multi-item one.
2. The infeasibility item \( j_{\text{inf}(i)} \) identifies knapsack \( i \) as being big for its size class \( C_{\log_4(i) + 3} \).

**Proof** To prove the first claim, we show that knapsack \( i \) cannot be of the single-item type. For this purpose, note that by property 2 of the assignment \( \hat{x} \), the number of items assigned to knapsack \( i \) is \( \sum_{j \in [n]} \hat{x}_{ij} \leq \lfloor \sum_{j \in [n]} x_{ij}^* \rfloor \). However, for single-item knapsacks, constraint (3) ensures that \( \sum_{j \in [n]} x^*_{ij} \leq 1 \). Therefore, had knapsack \( i \) been single-item, it would have been assigned by \( \hat{x} \) at most one item, meaning that its capacity cannot be violated, since property 3 guarantees that each item fits by itself into the knapsack to which it is assigned.

For the second claim, it is worth pointing out that the infeasibility item \( j_{\text{inf}(i)} \) indeed exists, as asserted by property 4. Now, clearly, knapsack \( i \) could not have been identified by \( \hat{j}_{\text{inf}(i)} \) as being \( C_{\log_4(i) + 3} \)-small, as none of the \( C_{\log_4(i) + 3} \)-small active multi-item knapsacks are large enough to contain item \( j_{\text{inf}(i)} \) (see guessing procedure for group I in Sect. 5.3), contradicting the fact that infeasibility items are not associated with such constraints, by property 4.

Now, on the one hand, the distance value of our final permutation \( \hat{\pi} \) decomposes into the individual contributions of all airplanes, i.e., \( \text{dist}(\hat{\pi}) = \sum_{j \in [n]} \text{dist}_j(\hat{\pi}) \). On the other hand, by property 1 of the assignment \( \hat{x} \), we have \( \sum_{j \in [n]} \sum_{i \in K} v_{ij} \hat{x}_{ij} \geq (1 - 9\epsilon) \cdot \text{dist}(\pi^*) \). Therefore, for the purpose of proving that \( \text{dist}(\hat{\pi}) = (1 - O(\epsilon)) \cdot \text{dist}(\pi^*) \), it suffices to show that the distance contribution \( \text{dist}_{\pi^{-1}(j)}(\hat{\pi}) \) of each airplane \( A_j \) is within factor \( 1 - O(\epsilon) \) of the assignment value \( \sum_{i \in K} v_{ij} \hat{x}_{ij} \) due to item \( j \), as established in the next claim.

**Lemma 7** For every \( j \in [n] \),

\[
\text{dist}_{\pi^{-1}(j)}(\hat{\pi}) \geq (1 - \epsilon) \cdot \sum_{i \in K} v_{ij} \hat{x}_{ij} .
\]

**Proof** Recall that, in Sect. 5.4, our resulting permutation \( \hat{\pi} \) was defined as \( (\hat{\pi}_{\{i1\}}, \hat{\pi}_{\{i2\}}, \ldots) \). Here, \( e_{i1} < e_{i2} < \cdots \) is the sorted sequence of the knapsacks’ upper endpoints \( \{ e_i \}_{i \in K} \), and each \( \hat{\pi}_{\{i\}} \) is an arbitrary permutation of the airplane indices \( \{ j \in [n] : \hat{x}_{ij} = 1 \} \), or equivalently, the items assigned by \( \hat{x} \) to knapsack \( i \).

With this notation, let us focus on some airplane \( A_j \), and suppose that its corresponding item \( j \) was assigned by \( \hat{x} \) to knapsack \( i \). As a result, we have in particular

\[
\sum_{i \in K} v_{ij} \hat{x}_{ij} = v_{ij} = \frac{v_j}{e_{i1}} = \sum_{\ell \in [\hat{k}(i)]} v_j \hat{x}_{i\ell} .
\]

where the last equality is obtained by expressing the upper endpoint \( e_{i1} \) of knapsack \( i \) as the overall capacity of all knapsacks before (according to \( \preceq \)) and including \( i \). However, since item \( j \) was assigned to knapsack \( i \), we must have \( j \) appearing in the permutation \( \hat{\pi}_{\{i\}} \), and by definition of \( \hat{\pi} \), it follows that

\[
\text{dist}_{\pi^{-1}(j)}(\hat{\pi}) \geq \sum_{\ell \in [\hat{k}(i)]} v_j \hat{x}_{i\ell} .
\]

Thus, to prove that \( \text{dist}_{\pi^{-1}(j)}(\hat{\pi}) \geq (1 - \epsilon) \cdot \sum_{i \in K} v_{ij} \hat{x}_{ij} \), it is sufficient to show that

\[
\sum_{\ell \in [\hat{k}(i)]} \sum_{\ell \in [\hat{k}(i)]} c_\ell \hat{x}_{i\ell} \leq (1 + \epsilon) \cdot \sum_{\ell \in [\hat{k}(i)]} \hat{k}(i) .
\]

For this purpose, let us first restrict attention to the summands in the left-hand side of the above inequality due to single-item knapsacks. By the first claim in Lemma 6, it follows that \( \hat{x} \) does not exceed the capacity of any single-item
knapsack, implying that
\[
\sum_{r \in \mathcal{R}} \sum_{i \in [n]} c_r \hat{\mathbf{x}}_{i_r \ell} \leq \sum_{r \in \mathcal{R}} \tilde{k}(i_r).
\]

Now, in order to bound summations due to active multi-item knapsacks, consider one such knapsack, \(i_r\). By the second claim in Lemma 6, if \(\hat{x}\) exceeds the capacity of \(i_r\), the infeasibility item \(j_{\text{inf}}(i_r)\) identifies knapsack \(i_r\) as being big for its size class \(C_{\log 1+\epsilon} c_{\text{inf}(i_r)}\). However, by instantiating Lemma 5 with \(r = \log (1+\epsilon) c_{\text{inf}(i_r)}\), it follows that the size of item \(j_{\text{inf}}(i_r)\) satisfies \(c_{\text{inf}(i_r)} \leq \epsilon \cdot \tilde{k}(i_r)\). By combining this observation with property 4 of the assignment \(\hat{x}\), we conclude that \(\hat{x}\) exceeds the capacity \(\tilde{k}(i_r)\) of knapsack \(i_r\) by a fraction of at most \(\epsilon\). Consequently,
\[
\sum_{r \in \mathcal{R}} \sum_{i \in [n]} c_r \hat{\mathbf{x}}_{i_r \ell} \leq (1 + \epsilon) \sum_{r \in \mathcal{R}} \tilde{k}(i_r).
\]

To obtain the desired inequality, we now combine the summations due to single-item and active multi-item knapsacks:
\[
\sum_{r \in \mathcal{R}} \sum_{i \in [n]} c_r \hat{\mathbf{x}}_{i_r \ell} = \sum_{r \in \mathcal{R}} \sum_{i \in [n]} c_r \hat{\mathbf{x}}_{i_r \ell} + \sum_{r \in \mathcal{R}} \sum_{i \in [n]} c_r \hat{\mathbf{x}}_{i_r \ell} \\
\leq (1 + \epsilon) \sum_{r \in \mathcal{R}} \tilde{k}(i_r).
\]

\(\square\)

### 6 Running time improvement

Up until now, we have shown how to approximate the airplane refueling problem within factor \(1 - \epsilon \) of optimal. However, as stated in Theorem 3, our algorithm incurs a running time of \(\text{poly}(|\mathcal{I}|) \cdot O(C_{\log \frac{1}{1+\epsilon}}^1)\), which is only pseudo-polynomial for any fixed \(\epsilon\), since \(C = \sum_{j=1}^m c_j\) is not necessarily polynomial in the input size \(|\mathcal{I}|\). To derive Theorem 1, we next show how to improve this running time and attain a true polynomial-time approximation scheme. The general idea behind this improvement is to partition the collection of airplanes \(A_1, \ldots, A_n\) into groups such that:

1. The ratio between \(c_{\max}\) and \(c_{\min}\) within each group is at most \((n/\epsilon)^{O(1/\epsilon)}\). Consequently, after normalizing all consumption rates by \(c_{\min}\), we would have \(C = (n/\epsilon)^{O(1/\epsilon)}\). Our pseudo-PTAS can now be used to separately obtain a \((1 - \epsilon)\)-approximation for each group by itself, in time \(\text{poly}(|\mathcal{I}|) \cdot O((n/\epsilon)^{O(1/\epsilon \log 1/\epsilon)})\).
2. The collection of approximate solutions to the different groups can be glued together, forming a near-optimal solution for the original instance.

#### 6.1 The classify-and-delete approach

*Introducing large size gaps* We begin by partitioning the set of airplanes to fuel consumption rate classes \(S_1, S_2, \ldots\) by powers of \(n/\epsilon\). Specifically, recalling that \(c_{\min} = 1\), the first class \(S_1\) consists of all airplanes whose consumption rate is in \([1, n/\epsilon]\), the second class \(S_2\) corresponds to consumption rates in \([n/\epsilon, (n/\epsilon)^2]\), so on and so forth. Let \(r \in \{0, 1, \ldots, 1/\epsilon - 1\}\) be a parameter whose value will be determined later, noting that \(1/\epsilon\) was assumed to take integer values in Sect. 4.2. We modify the original instance \(\mathcal{I}\) by eliminating all airplanes in each class \(S_r\) for which \(\ell \equiv (r \mod 1/\epsilon)\), and denote the resulting instance by \(\mathcal{I}_r\). Now let \(\pi^*_r\) be the optimal permutation with respect to the instance \(\mathcal{I}_r\). The next claim proves that there exists a value of \(r\) such that the optimum value \(\text{dist}^r(\pi^*_r)\) for the instance \(\mathcal{I}_r\) nearly matches the optimum value \(\text{dist}^2(\pi^*)\) of the original instance \(\mathcal{I}\). To avoid confusion, the superscripts \(\mathcal{I}_r\) and \(\mathcal{I}\) over \(\text{dist}(\cdot)\) designate the specific instance being considered.

**Lemma 8** \(\text{dist}^r(\pi^*_r) \geq (1 - \epsilon) \cdot \text{dist}^2(\pi^*)\) for some \(r \in \{0, 1, \ldots, 1/\epsilon - 1\}\).

**Proof** Our proof is based on a simple application of the probabilistic method (Alon and Spencer 2016). Specifically, we show that by picking a random uniformly distributed number \(R\) from the set \(\{0, 1, \ldots, 1/\epsilon - 1\}\), we have \(E(\text{dist}^R(\pi^*_R)) \geq (1 - \epsilon) \cdot \text{dist}^2(\pi^*)\). This implies that at least one value of \(r\) in the latter set satisfies the desired inequality.

Let us focus on an arbitrary realization \(\mathcal{I}_r\) of our randomly constructed instance. One possible solution to \(\mathcal{I}_r\) is obtained by deleting from the optimal permutation \(\pi^*\) all airplanes that were eliminated during the creation of \(\mathcal{I}_r\); we refer to the resulting permutation as \(\tilde{\pi}_r\). Clearly, the distance contribution of every surviving airplane with respect to \(\tilde{\pi}_r\) is at least as large as its contribution with respect to \(\pi^*\), meaning that
\[
\text{dist}^r(\pi^*_r) \geq \text{dist}^r(\tilde{\pi}_r) \geq \sum_{j \in [n]} \text{dist}^r_j(\pi^*) \cdot [A_{\pi^*(j)} \in \mathcal{I}_r].
\]

Based on this observation,
\[
E\left[\text{dist}^R(\pi^*_R)\right] \geq \sum_{j \in [n]} \text{dist}^R_j(\pi^*) \cdot \Pr[A_{\pi^*(j)} \in \mathcal{I}_R]
\]
where the first equality holds since each airplane is removed with probability \( \epsilon \).

**Blocks and their permutations** By testing each of the values 0, 1, \ldots, 1/\epsilon - 1 as potential candidates, we assume from this point on that a value of \( r \) satisfying Lemma 8 is known in advance. We define a block to be a maximal set of undeleted classes out of \( S_1, S_2, \ldots \) that have consecutive indices. For example, suppose that \( r = 0 \), then the first block is defined as \( B_1 = S_1 \cup \cdots \cup S_{(1/\epsilon)-1} \), the second block is \( B_2 = S_{(1/\epsilon)+1} \cup \cdots \cup S_{(2/\epsilon)-1} \), and so on. Note that all blocks consist of \( 1/\epsilon - 1 \) consecutive classes except for possibly the first and last blocks that may consist of fewer classes. By definition, within each block the ratio between the largest and smallest consumption rates is at most \((n/\epsilon)^{1/\epsilon}\). As a result, for each block of airplanes \( B_i \), we can separately utilize our pseudo-PTAS to approximate the instance \( \mathcal{I}_B \) induced by this block to within factor \( 1-\epsilon \), while incurring a running time of \( \text{poly}(|\mathcal{I}|) \cdot O(n(\frac{1}{\epsilon})^{\frac{3}{2}}) \). We denote by \( \tilde{\pi}_{[B_i]} \) the resulting permutation for block \( B_i \).

**Merging the permutations** For the purpose of creating a near-optimal permutation \( \tilde{\pi}_r \) for the modified instance \( \mathcal{I}_r \), let us denote the blocks associated with this instance by \( B_1, B_2, \ldots \), indexed in order of increasing consumption rates. Now, the permutations computed for these blocks are concatenated into a full permutation \( \tilde{\pi}_r \) of all airplanes surviving in \( \mathcal{I}_r \) according to the block order, i.e., \( \tilde{\pi}_r = \langle \tilde{\pi}_{[B_1]}, \tilde{\pi}_{[B_2]}, \ldots \rangle \).

### 6.2 Concluding the proof of Theorem 1

In Lemma 9 below, we show that the objective value \( \text{dist}^{\mathcal{I}_r}(\tilde{\pi}_r) \) of the permutation \( \tilde{\pi}_r \) is at least \((1-\epsilon)^2\) times the optimum value \( \text{dist}^{\mathcal{I}_r}(\pi^*_r) \) for the instance \( \mathcal{I}_r \). By combining this claim with Lemma 8, and appending the airplanes missing from \( \mathcal{I}_R \) at the end of \( \tilde{\pi}_r \), we conclude the proof of Theorem 1.

**Lemma 9** \( \text{dist}^{\mathcal{I}_r}(\tilde{\pi}_r) \geq (1-\epsilon)^2 \cdot \text{dist}^{\mathcal{I}_r}(\pi^*_r) \).

**Proof** Our proof consists of establishing the next two claims for every block \( B_i \):

1. The optimum value for the instance \( \mathcal{I}_{B_i} \) is at least as large as the total distance contribution of all airplanes in block \( B_i \) with respect to the permutation \( \pi^*_r \). In other words, letting \( \pi^*_{[B_i]} \) be an optimal permutation for the instance \( \mathcal{I}_{B_i} \), we have

\[
\text{dist}^{\mathcal{I}_{B_i}}(\pi^*_{[B_i]}) \geq \sum_{j \in B_i} \text{dist}^{\mathcal{I}_{\pi^*_{[B_i]}^{-1}(j)}}(\pi^*_r) .
\]

2. The total distance contribution of all airplanes in block \( B_i \) with respect to the permutation \( \tilde{\pi}_r \) is at least \((1-\epsilon)^2\) times the optimum value for the instance \( \mathcal{I}_{B_i} \), i.e.,

\[
\sum_{j \in B_i} \text{dist}^{\mathcal{I}_{\tilde{\pi}_r^{-1}(j)}}(\tilde{\pi}_r) \geq (1-\epsilon)^2 \cdot \text{dist}^{\mathcal{I}_{B_i}}(\pi^*_R) .
\]

The combination of these claims indeed implies the desired bound, since

\[
\text{dist}^{\mathcal{I}_{\tilde{\pi}_r^{-1}}(j)}(\tilde{\pi}_r) \geq (1-\epsilon)^2 \cdot \text{dist}^{\mathcal{I}_{B_i}}(\pi^*_R)
\]

where the first and second inequalities follow are implied by claims 2 and 1, respectively.

**Proof of claim 1** Suppose we modify the permutation \( \pi^*_r \), such that the airplanes in block \( B_i \) appear first according to their internal order with respect to \( \pi^*_r \), followed by the remaining planes in arbitrary order. Clearly, as a result of this modification, the distance contribution of each airplane in \( B_i \) can only increase. However, their internal permutation is in particular a feasible solution to the instance \( \mathcal{I}_{B_i} \), and since \( \pi^*_{[B_i]} \) is optimal for the latter instance,

\[
\text{dist}^{\mathcal{I}_{B_i}}(\pi^*_{[B_i]}) \geq \sum_{j \in B_i} \text{dist}^{\mathcal{I}_{\pi^*_{[B_i]}^{-1}(j)}}(\pi^*_r) .
\]

**Proof of claim 2** This is precisely the part where creating large gaps between consecutive blocks plays an important role. Let us focus on an arbitrary airplane \( A_j \) in block \( B_i \). The distance contribution of this airplane with respect to the permutation \( \tilde{\pi}_r \) is given by

\[
\text{dist}^{\mathcal{I}_{\tilde{\pi}_r^{-1}}(j)}(\tilde{\pi}_r) \geq \sum_{\ell < i} \sum_{k \in B_{\ell}} c_k + \sum_{k \leq \tilde{\pi}_{[B_i]}^{-1}(j)} c_{\tilde{\pi}_{[B_i]}(k)}
\]

\[
\geq (1-\epsilon) \cdot \sum_{k \leq \tilde{\pi}_{[B_i]}^{-1}(j)} c_{\tilde{\pi}_{[B_i]}(k)}
\]

\[
= (1-\epsilon) \cdot \text{dist}^{\mathcal{I}_{B_i}}(\tilde{\pi}_{[B_i]}^{-1}(j)) .
\]

Here, the first equality holds since, in the permutation \( \tilde{\pi}_r \), the blocks appearing before airplane \( A_j \) in \( B_1, \ldots, B_{i-1} \), and within block \( B_i \), the airplanes appearing before and including \( A_j \) are those located in positions \( 1, \ldots, \tilde{\pi}_{[B_i]}^{-1}(j) \).
of the permutation $\tilde{\pi}(B_i)$. The succeeding inequality follows from observing that, due to having a gap of at least $n/\epsilon$ between the largest consumption rate in $B_1, \ldots, B_{i-1}$ and the smallest consumption rate in $B_i$, it follows that

$$\sum_{k \leq \tilde{\pi}(B_i)(j)} c_j \leq \left( \sum_{k \leq \tilde{\pi}(B_i)(j)} \right) \cdot \frac{\epsilon}{n} \cdot c_j \leq \epsilon \cdot \sum_{k \leq \tilde{\pi}(B_i)(j)} c_j,$$

Now, summing the lower bound on distance contribution of $A_j$ with respect to the permutation $\tilde{\pi}$ over all airplanes in block $B_i$, we have

$$\sum_{j \in B_i} \text{dist}_{\tilde{\pi}^{-1}(j)}(\tilde{\pi}) \geq (1 - \epsilon) \cdot \sum_{j \in B_i} \text{dist}_{\tilde{\pi}^{-1}(j)}(\tilde{\pi}(B_i)) = (1 - \epsilon) \cdot \text{dist}_{\tilde{\pi}^{-1}}(\tilde{\pi}(B_i)) \geq (1 - \epsilon)^2 \cdot \text{dist}_{\tilde{\pi}^{-1}}(\tilde{\pi}(B_i)),$$

where the last inequality holds since the permutation $\tilde{\pi}(B_i)$ provides a $(1 - \epsilon)$-approximation for the optimum value of the instance $\mathcal{I}_{B_i}$. \QED

A Additional proofs

A.1 Proof of Lemma 3

The desired upper bound follows by observing that

$$\tilde{\kappa}(\tilde{I}_i) = (1 + \epsilon) \cdot \Delta(\tilde{I}_i) = (1 + \epsilon) \cdot \left( \Delta(\tilde{I}_i) + 2 \epsilon \cdot \Delta(I_i) \right) \leq (1 + \epsilon) \cdot (1 + 2 \epsilon) \cdot \Delta(I_i) \leq (1 + 2 \epsilon) \cdot \left( (1 + \epsilon)^i - 1 \right) \leq 3 \epsilon \cdot (1 + \epsilon)^i,$$

where the second equality is precisely the definition of $\Delta(\tilde{I}_i)$ in Sect. 4.2, the first inequality holds since $\Delta(\tilde{I}_i) \leq \Delta(I_i)$, and the second inequality is obtained by noting that $I_i = [(1 + \epsilon)^{i-1}, (1 + \epsilon)^i)$. \QED

A.2 Proof of Lemma 4

As multi-item buckets are indexed by their left-to-right order on the timeline, the active ones that appear before and including $\tilde{I}_i$ are precisely $\{\tilde{I}_k\}_{k \in A: k \leq i}$. Along with the relevant single-item buckets, we bound the overall capacity in question as follows:

$$\sum_{k \in A: k < i} \tilde{\kappa}(\tilde{I}_k) + \sum_{J_i < \tilde{I}_i} \tilde{\kappa}(J_i) = (1 + \epsilon) \cdot \left( \sum_{k \in A: k < i} \Delta(\tilde{I}_k) + \sum_{J_i < \tilde{I}_i} \Delta(J_i) \right) \leq (1 + \epsilon) \cdot (1 + 4 \epsilon) \cdot \sum_{k = 1}^{i} \Delta(\tilde{I}_k) \leq (1 + 4 \epsilon) \cdot (1 + \epsilon)^i.$$

Similarly to the proof of Lemma 2, the first inequality is a worst-case upper bound, where each bucket-crossing interval resides between two successive original intervals, in which case we incur an overall length increase in at most $1 + 4 \epsilon$. The last inequality holds since $\Delta(\tilde{I}_i) = (1 + \epsilon)^i - (1 + \epsilon)^{i-1}$. \QED

B A logarithmic approximation

In what follows, we provide a succinct description of a folklore algorithm for approximating the airplane refueling problem within factor $\Omega(1/\log n)$. Specifically, we first argue that a greedy decision rule can be used to compute an optimal drop out permutation when all planes have identical $v_j/c_j$ ratios. This observation is then employed in conjunction with the classify-and-select method to obtain a logarithmic approximation for the general case.

Lemma 10 Given an airplane refueling instance where all $v_j/c_j$ ratios are equal, an optimal permutation $\pi^*$ orders the airplanes according to non-decreasing $c_j$ values.

Proof Suppose that $\pi^*$ violates the above-mentioned rule. In particular, this means that there are two airplanes $A_j$ and $A_l$ with $c_j > c_l$ that appear one after the other, i.e., $\pi^*(\ell) = j$ and $\pi^*(\ell + 1) = i$. Let $C_{\ell-1} = \sum_{k=1}^{\ell-1} c_{\pi^*(k)}$, and notice that the combined distance contribution of these two airplanes is

$$\text{dist}(\pi^*) + \text{dist}(\pi^*) = \frac{v_j}{C_{\ell-1} + c_j} + \frac{v_j}{C_{\ell-1} + c_j + c_i},$$

However, if we switch the positions of $A_j$ and $A_l$, to obtain a new permutation $\pi^{**}$, their distance contribution becomes

$$\text{dist}(\pi^{**}) + \text{dist}(\pi^{**}) = \frac{v_l}{C_{\ell-1} + c_i} + \frac{v_j}{C_{\ell-1} + c_i + c_j},$$

while the contribution of any other airplane remains unchanged. As a result,

$$\text{dist}(\pi^{**}) - \text{dist}(\pi^*) = \frac{v_l c_j (C_{\ell-1} + c_j) - v_l c_j (C_{\ell-1} + c_i)}{(C_{\ell-1} + c_j)(C_{\ell-1} + c_j + c_i)} = \frac{v_j c_i - v_l c_j}{C_{\ell-1} + c_j}.$$
general case, where the overall distance contribution of all airplanes in class \( R_\ell \) satisfies

\[
\frac{c_j v_j (c_j - c_i)}{(C_{\ell-1} + c_i)(C_{\ell-1} + c_j)(C_{\ell-1} + c_i + c_j)} > 0 ,
\]

where the third equality holds since \( v_j/c_j = v_i/c_i \), and the last inequality follows as \( c_i < c_j \). Therefore, dist(\( \pi^- \)) > dist(\( \pi^+ \)), contradicting the optimality of \( \pi^+ \).

\[ \square \]

We now utilize the above lemma to approximate the general case, where the \( v_j/c_j \) ratios are arbitrary. For this purpose, we denote the maximum such ratio over all airplanes by \( \rho_{\text{max}} = \max_j (v_j/c_j) \). We partition the set of airplanes into \( O(\log n) \) ratio classes \( R_{\text{small}}, R_1, \ldots, R_L \), geometrically by powers of 2. Specifically, \( R_1 \) consists of all airplanes with \( v_j/c_j \in (\rho_{\text{max}}/(2n), \rho_{\text{max}}/n] \). \( R_2 \) consists of those with \( v_j/c_j \in (\rho_{\text{max}}/n, 2 \cdot \rho_{\text{max}}/n] \), so on and so forth, where the additional class \( R_{\text{small}} \) consists of airplanes with \( v_j/c_j \in (0, \rho_{\text{max}}/(2n)] \).

**Lemma 11** There exists an index \( 1 \leq \ell^* \leq L \) for which the overall distance contribution of all airplanes in class \( R_\ell \) satisfies

\[
\sum_{j \in R_\ell^*} \text{dist}_{\pi^-^{-1}(j)}(\pi^+) \geq \frac{1}{2L} \cdot \text{dist}(\pi^+) ,
\]

where \( \pi^+ \) stands for the optimal permutation.

**Proof** The desired claim follows by observing that the combined distance contribution of all airplanes in classes \( R_1, \ldots, R_L \) is

\[
\sum_{\ell \in [L]} \sum_{j \in R_\ell^*} \text{dist}_{\pi^-^{-1}(j)}(\pi^+) = \text{dist}(\pi^+) - \sum_{j \in R_{\text{small}}} \text{dist}_{\pi^-^{-1}(j)}(\pi^+) \geq \text{dist}(\pi^+) - \rho_{\text{max}} \geq \frac{1}{2} \cdot \text{dist}(\pi^+) .
\]

Here, the first inequality holds since the distance contribution of each airplane \( A_j \) with respect to any permutation is upper bounded by \( v_j/c_j \). The second inequality follows from the definition of \( R_{\text{small}} \), ensuring that \( v_j/c_j \leq \rho_{\text{max}}/(2n) \) for each airplane in this class. The last inequality holds since the optimal permutation \( \pi^+ \) clearly satisfies \( \text{dist}(\pi^+) \geq \rho_{\text{max}} \).

From this point on, we assume that the index \( \ell^* \) is known in advance. This assumption can be justified by testing each of the values \( 1, \ldots, L \) as a candidate for the correct value of \( \ell^* \). By Lemma 11, in order to obtain an \( \Omega(1/\log n) \)-approximation for the original airplane refueling instance, it remains to show that the optimal permutation for the class \( R_{\ell^*} \) can be approximated to within a constant factor. Our final solution will simply append the remaining airplanes (in arbitrary order) to the resulting permutation, which can only improve the objective value. Now, for class \( R_{\ell^*} \), since its airplanes have a \( v_j/c_j \) ratio in the interval \( (2^{\ell^*-2} \cdot \rho_{\text{max}}/n, 2^{\ell^*-1} \cdot \rho_{\text{max}}/n] \), all ratios can be made identical by rounding down the \( v_j \) value of each plane, losing a factor of at most 2 in optimality. For this setting, we can apply the greedy decision rule, that computes an optimal permutation by Lemma 10.

**References**

Afrati, F. N., Bampis, E., Chekuri, C., Karger, D. R., Kenyon, C., Khanna, S., et al. (1999). Approximation schemes for minimizing average weighted completion time with release dates. In 40th Annual IEEE symposium on foundations of computer science (pp. 32–44).

Albers, S., Barua, S. K., Möhring, R. H., & Pruhs, K. (2010). 10071 open problems—scheduling. In Dagstuhl seminar proceedings 10071. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

Alon, N., & Spencer, J. H. (2016). The probabilistic method (4th ed.). Hoboken: Wiley.

Baltag, A., van Ditmarsch, H., & Moss, L. (2008). Epistemic logic and information update. In P. Adriaans & J. van Benthem (Eds.), Handbook of the philosophy of information (pp. 361–456). Amsterdam: Elsevier.

Bansal, N., & Pruhs, K. (2014). The geometry of scheduling. SIAM Journal on Computing, 43(5), 1684–1698.

Chekuri, C., & Khanna, S. (2004). Approximation algorithms for minimizing average weighted completion time. In J. Y. T. Leung (Ed.), Handbook of scheduling: Algorithms, models, and performance analysis. Boca Raton: CRC Press.

Chekuri, C., & Khanna, S. (2005). A polynomial time approximation scheme for the multiple knapsack problem. SIAM Journal on Computing, 35(3), 713–728.

Cheung, M., Mestre, J., Shmoys, D. B., & Verschae, J. (2017). A primal-dual approximation algorithm for min-sum single-machine scheduling problems. SIAM Journal on Discrete Mathematics, 31(2), 825–838.

Chuzhoy, J., Ostrovsky, R., & Rabani, Y. (2006). Approximation algorithms for the job interval selection problem and related scheduling problems. Mathematics of Operations Research, 31(4), 730–738.

Cohen, R., Katzir, L., & Raz, D. (2006). An efficient approximation for the generalized assignment problem. Information Processing Letters, 100(4), 162–166.

Correa, J. R., Skutella, M., & Verschae, J. (2012). The power of preemption on unrelated machines and applications to scheduling orders. Mathematics of Operations Research, 37(2), 379–398.

De, S., & Sen, A. (1996). The generalised Gamow–Stern problem. The Mathematical Gazette, 80(488), 345–348.

Feige, U., & Vondrák, J. (2010). The submodular welfare problem with demand queries. Theory of Computing, 6(1), 247–290.

Feldman, M., & Naor, J. S. (2017). Non-preemptive buffer management for latency sensitive packets. Journal of Scheduling, 20(4), 337–353.
Fiat, A., Mansour, Y., & Nadav, U. (2008). Competitive queue management for latency sensitive packets. In Proceedings of the 19th annual ACM-SIAM symposium on discrete algorithms (pp. 228–237).

Fleischer, L., Goemans, M. X., Mirrokni, V. S., & Sviridenko, M. (2011). Tight approximation algorithms for maximum separable assignment problems. Mathematics of Operations Research, 36(3), 416–431.

Gamow, G., & Stern, M. (1958). Puzzle-math. New York City: Viking Press.

Gandhi, R., Halldórsson, M. M., Kortsarz, G., & Shachnai, H. (2008). Improved bounds for scheduling conflicting jobs with minsum criteria. ACM Transactions on Algorithms, 4(1), 11:1–11:20.

Hall, L. A., Schulz, A. S., Shmoys, D. B., & Wein, J. (1997). Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. Mathematics of Operations Research, 22(3), 513–544.

Höhn, W. (2014). Complex single machine scheduling: Theoretical and practical aspects of sequencing. Ph.D. thesis, TU Berlin.

Höhn, W., Mestre, J., & Wiese, A. (2014). How unsplittable-flow-covering helps scheduling with job-dependent cost functions. In Proceedings of the 41st international colloquium on automata, languages, and programming (pp. 625–636).

Knuth, D. E. (1969). The Gamow–Stern elevator problem. Journal of Recreational Mathematics, 2, 131–137.

Lübbeke, E., Maurer, O., Megow, N., & Wiese, A. (2016). A new approach to online scheduling: Approximating the optimal competitive ratio. ACM Transactions on Algorithms, 13(1), 15:1–15:34.

Megow, N. & Verschae, J. (2013). Dual techniques for scheduling on a machine with varying speed. In Proceedings of the 40th international colloquium on automata, languages, and programming (pp. 745–756).

Nutov, Z., Beniaminy, I., & Yuster, R. (2006). A (1−1/ e)-approximation algorithm for the generalized assignment problem. Operations Research Letters, 34(3), 283–288.

Procaccia, A. D. (2013). Cake cutting: Not just child’s play. Communications of the ACM, 56(7), 78–87.

Shmoys, D. B., & Tardos, É. (1993). An approximation algorithm for the generalized assignment problem. Mathematical Programming, 62, 461–474.

Sitters, R. (2014). Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. In Proceedings of the 25th annual ACM-SIAM symposium on discrete algorithms (pp. 604–616).

Thomson, W. (2011). Fair allocation rules. In K. J. Arrow, A. Sen, & K. Suzumura (Eds.), Handbook of social choice and welfare (Vol. 2, pp. 393–506). Amsterdam: North-Holland Publishing Co.

van Ditmarsch, H., Ruan, J., & Verbrugge, R. (2008). Sum and product in dynamic epistemic logic. Journal of Logic and Computation, 18(4), 563–588.

Wuffle, A. (1982). The pure theory of elevators. Mathematics Magazine, 55(1), 30–37.