The Exact One-Half Threshold for Arbitrary-Capacity Bottleneck Multiple Subset Sum*

Matěj Kripner Václav Rozhoň Robert Šámal
Martin Šindelář Adrián Zámečník

hexagon:2611.00054v1

Abstract

In bottleneck multiple subset sum, items with weights are packed into knapsacks with capacities, possibly leaving items unpacked, so as to maximize the minimum load. For identical capacities the best possible approximation ratio is 2/3, and Chen et al. asked whether 2/3 is also achievable for arbitrary capacities. We show that it is not: for arbitrary capacities we give a 1/2-approximation using $O(n \log(n+1) + m \log(m+1))$ arithmetic and comparison operations for n items and m knapsacks, and prove that no fixed ratio above 1/2 is achievable unless $\mathsf{P} = \mathsf{NP}$. We also give a short proof of the classical 1/2 guarantee of Friesen and Langston for unit profits.

1 Introduction

1.1 Background and the source problem

Packing as much material as possible and ensuring that every container receives a substantial amount are different objectives. The latter leads to *bottleneck multiple subset sum*: each item has a nonnegative weight, each knapsack has a capacity, and disjoint subsets of items are assigned to maximize the smallest total load. Items may remain unpacked. The objective is an absolute load, not a fraction of the receiving knapsack's capacity. For identical capacities, Caprara, Kellerer, and Pferschy [1] established the exact polynomial-time approximation threshold 2/3, under $\mathsf{P} \neq \mathsf{NP}$.

Allowing different capacities changes which combinations of items can be assigned to which knapsacks. Chen et al. [3] study the more general bottleneck multiple knapsack problem, where profits and weights are independent. Their Theorems 2–3 give a $(1/2 - \varepsilon)$-approximation for arbitrary capacities and hardness above 1/2, already with unit profits. Their published concluding Section 7 then asks whether the special case with profit equal to weight retains the identical-capacity guarantee:

Question 1.1 (Chen et al., [3, Section 7, p. 60:18])**.** Is a 2/3-approximation achievable for bottleneck multiple subset sum when the knapsack capacities are arbitrary?

The same discussion asks the following separate question.

Question 1.2 (Chen et al., [3, Section 7, p. 60:18])**.** Does arbitrary-capacity unit-profit BMKP admit a polynomial-time 1/2-approximation?

These questions have different histories. The unit-profit problem is the older problem of maximizing the minimum number of pieces in unequal bins. Friesen and Langston [4] already proved a 1/2 guarantee with zero additive loss using a first-fit-decreasing implementation, together with hardness of every larger ratio with zero additive loss. The unit-profit proof below is an alternative elementary argument for that known result. The main subject of this note is the first question, for profits equal to weights.

*This work was produced within the Bolzano project (https://bolzano.app), using AI systems for mathematical research and manuscript preparation. The listed authors contributed to the project and the preparation of this release. The manuscript received only limited human checking.

1.2 Contribution

The arbitrary-capacity threshold for bottleneck MSSP is $1/2$. The two parts of this assertion are proved separately.

1. Theorem 3.6 gives a $1/2$-approximation using $O(n \log(n+1) + m \log(m+1))$ arithmetic and comparison operations and $O(n+m)$ space on rational inputs, with no ε loss.
2. Theorem 4.2 excludes every fixed ratio greater than $1/2$, unless $\mathsf{P} = \mathsf{NP}$; the real-ratio extension is made explicit in Corollary 4.4. In particular, the $2/3$ guarantee in the first question cannot hold for arbitrary capacities. Corollary 4.3 shows that this obstruction persists when both the capacities and the item weights have arbitrarily small relative spread.

The algorithm separates the two ways a knapsack can receive half a target load T. A single item of weight at least $T/2$ suffices, but may fit only some knapsacks. A maximum bipartite matching assigns as many such items as possible. The unmatched knapsacks are filled using only items of weight less than $T/2$; adding these items until a load first reaches $T/2$ keeps the load below T. Thus, if all capacities are at least T, sufficient total small-item weight can fill the unmatched knapsacks without violating capacity.

More precisely, if q_T is the maximum matching size and W_T is the total small-item weight, the sufficient condition is

$$T \le B_{\min}, \qquad W_T \ge (m - q_T)T.$$

An optimal assignment itself certifies this inequality at $T = \mathrm{OPT}$: choose one big item in every knapsack containing one, and count the small weight in the other knapsacks. The neighborhoods in the matching graph are nested by item weight. Scanning the items from heaviest to lightest and assigning each fitting item to the largest unused capacity computes the maximum matching size for every suffix in one pass. The item classes change only at the $O(n)$ breakpoints $2w_i$. With the strict-small convention, these give intervals $(a, b]$ on which the certificate is linear in T and its largest feasible value is attained. Prefix sums and the suffix matching sizes therefore find the largest certified rational target in linear time after sorting. Neither denominator scaling nor a general matching algorithm is required. For example, capacities $5, 8$ and weights $6, 2, 2, 1$ certify $T = 5$. The big item 6 is assigned to capacity 8, while the small items have total weight 5; two weight-2 items already raise the other knapsack past $T/2$.

The hardness proof uses a different aspect of heterogeneous capacities: they encode prescribed pair sums. A balanced Numerical Matching with Target Sums instance [5, Problem SP17, p. 224] is embedded using low weights $A + x_i$, high weights $A + D + y_i$, and capacities $2A + D + b_j$. An assignment above a suitable threshold must use exactly two items per knapsack. High–high pairs do not fit; counting then forces one low and one high item in every knapsack. Equality of total weight and total capacity forces every pair to meet its prescribed sum. A large common offset makes the no-instance/yes-instance ratio arbitrarily close to $1/2$. This proves hardness within the profit-equals-weight subclass, which does not follow merely from hardness for unrestricted profits. The common offset can also make all capacities, and separately all item weights, arbitrarily close in relative terms. Thus the change from the identical-capacity threshold is already present under arbitrarily small capacity variation; it does not require very large and very small knapsacks in the same instance.

Section 5 gives the short sorted-block argument for the classical unit-profit endpoint (Theorem 5.2). Its prefix-count proof explains why $\lceil \mathrm{OPT}/2 \rceil$ items can be placed in every knapsack after sorting both capacities and weights. The matching certificate and numerical-matching reduction establish the main MSSP threshold.

1.3 Related work and the role of the objective

Covering problems. Maximizing the minimum load is the covering counterpart of scheduling and bin packing. In machine covering, all jobs must be assigned to machines without capacities and the

minimum completion time is maximized; Deuermeyer, Friesen and Langston analyzed the LPT rule [6], and Woeginger gave a PTAS [7]. In bin covering, one maximizes the number of bins whose load reaches a common threshold; Assmann, Johnson, Kleitman and Leung introduced the problem [8], and Jansen and Solis-Oba gave an asymptotic FPTAS [9]. Bottleneck multiple subset sum differs from both: every knapsack has an upper capacity, items may be discarded, and the guarantee concerns the worst knapsack.

Max-sum packing. The max-sum version of multiple subset sum has a different approximation landscape. Caprara, Kellerer, and Pferschy give a PTAS for identical capacities [1] and, in a companion paper, for different capacities [2]. Chekuri and Khanna [10] extend the PTAS conclusion to multiple knapsack with independent profits and weights. These schemes maximize total packed profit. Losing the contents of one knapsack can cause a small relative loss in that total and still destroy a minimum-load guarantee, so these theorems do not settle the bottleneck question.

Fair allocation. The max-min objective also appears in fair allocation. Bansal and Sviridenko [11] study allocation with resource values that may depend on the recipient, without the individual weight-capacity constraints used here. Hummel [12, arXiv version, Theorem 4.5 and Lemma 6.4] obtains algorithmic 2/5 maximin-share guarantees for hereditary and budget-constrained valuations, using sufficiently accurate polynomial-time valuation oracles. With a common capacity and common item profits, the maximin-share benchmark agrees with the bottleneck optimum, explaining its connection to identical-capacity BMKP. With different budgets, each agent's maximin share is a separate benchmark and should not be identified with the common optimum considered here. Rohwedder, Rouhani, and Wennmann [13] instead impose a single global budget on allocation costs in their Budgeted Santa Claus problem. That budget constraint is also distinct from imposing a separate load bound on every knapsack.

Against this background, the result here supplies the exact 1/2 endpoint for arbitrary-capacity MSSP and transfers the hardness threshold into this restricted subclass. For general profits, the $(1/2-\varepsilon)$ algorithm of Chen et al. uses rounding and configuration and assignment linear programs [3, Section 5 of the full version]; the equality of profit and weight makes the simple matching-and-filling certificate possible here.

2 Models and notation

Let I be a finite item set and let $K = \{1, \dots, m\}$ be the knapsack set, where $m \geq 1$. Item i has nonnegative weight w_i and nonnegative profit p_i; knapsack j has nonnegative capacity B_j. A feasible assignment consists of pairwise disjoint sets $S_j \subseteq I$ such that

$$\sum_{i \in S_j} w_i \leq B_j \qquad (j \in K).$$

Its bottleneck value is $\min_j \sum_{i \in S_j} p_i$, and OPT denotes the maximum such value.

For a maximization problem, an algorithm has approximation ratio $\alpha \in [0, 1]$ if it always returns a feasible solution of value at least $\alpha\,\mathrm{OPT}$. Arithmetic inputs are binary-encoded rationals.

In the *unit-profit* model, $p_i = 1$ for every item, so the value is $\min_j |S_j|$. In bottleneck MSSP, $p_i = w_i$, so the value is the minimum assigned load. All algorithms below work directly with rational input. An arithmetic or comparison operation on these numbers counts as one operation in the stated running times; their bit costs are discussed separately. No bound on the magnitudes of the weights or capacities is assumed.

3 Multiple subset sum: half-certifiable targets

Fix a target $T > 0$. Call item i *small* if $w_i < T/2$ and *big* otherwise. Let W_T be the total weight of the small items. Form a bipartite graph G_T whose left side consists of the big items and whose

right side consists of the knapsacks; join item i to knapsack j when $w_i \le B_j$. Let q_T be the size of a maximum matching in G_T.

Definition 3.1. A target T is *half-certifiable* if

$$T \le B_{\min} \qquad \text{and} \qquad W_T \ge (m - q_T)T.$$

Lemma 3.2. *Given a half-certifiable target T and a maximum matching in G_T, one can construct in $O(n+m)$ further operations an assignment of load at least $T/2$ in every knapsack.*

Proof. Assign the big items of a maximum matching in G_T to their matched knapsacks. Each of the q_T matched knapsacks already has load at least $T/2$. There remain $r = m - q_T$ knapsacks and small items of total weight at least rT.

Process the remaining knapsacks in any order, adding unused small items until the load first reaches $T/2$. The final load is strictly less than T, since the last item has weight less than $T/2$; hence it fits because $T \le B_{\min}$. Each completed knapsack consumes less than T small-item weight. More explicitly, after completing $k < r$ of the unmatched knapsacks, at most kT has been consumed, so at least

$$rT - kT = (r-k)T \ge T$$

small-item weight remains. The next knapsack therefore reaches $T/2$. This invariant proves that the process cannot run out of items. □

Lemma 3.3. *The optimum value $O = \mathrm{OPT}$ of a positive-valued* MSSP *instance is half-certifiable.*

Proof. Fix an optimal assignment. Necessarily $O \le B_{\min}$. Let K' be the set of knapsacks containing an assigned item of weight at least $O/2$. Selecting one such item from each knapsack in K' gives a matching in G_O, so $q_O \ge |K'|$. Every item assigned to a knapsack outside K' is small, and each such knapsack has load at least O. Disjointness of the assigned sets therefore gives

$$W_O \ge (m - |K'|)O \ge (m - q_O)O.$$

Thus O satisfies both certificate inequalities. □

Lemma 3.4 (All suffix matchings)**.** *Suppose positive item weights $w_1 \le \cdots \le w_n$ and capacities $B_1 \le \cdots \le B_m$ are sorted. For each $i \in \{1, \ldots, n+1\}$, let q_i be the maximum number of items from the suffix $w_i, \ldots, w_n$ that can be assigned injectively to fitting knapsacks, where $q_{n+1} = 0$. All q_i can be computed in $O(n+m)$ operations. A maximum matching for any selected suffix can be recovered within the same bound.*

Proof. Scan the items from n down to 1. Maintain a pointer to the largest unused capacity. If the current item fits that capacity, assign it there and advance the pointer to the next smaller capacity. Otherwise skip the item. Record the number of assignments after every step.

We prove by induction that the recorded number is q_i. The claim holds for the empty suffix. Suppose the already processed suffix has maximum matching size q. If the next item fits the largest unused capacity, the scan constructs a matching of size $q+1$. Adding one item increases the maximum matching size by at most one, so this is optimal. If no capacity remains, the matching already has the maximum possible size m. In the remaining case the current item does not fit the largest unused capacity. Every previously processed item is at least as heavy, so none of the processed items fits any of the unused capacities. Only the q occupied capacities can be used by a matching of the new suffix, and the existing matching attains this upper bound.

Each item is examined once and each capacity is used at most once. Rerunning the scan on a specified suffix records the required item–knapsack pairs. □

Lemma 3.5 (Exact rational target search)**.** *After sorting the weights and capacities, the largest positive half-certifiable target can be found in $O(n+m)$ arithmetic and comparison operations, or one can determine that no such target exists.*

Proof. If $B_{\min} = 0$, no positive target is certified. Otherwise discard zero-weight items and retain the positive weights in their sorted order as in Lemma 3.4. Compute the suffix matching sizes q_i and prefix sums $P_j = \sum_{i=1}^{j} w_i$, including $P_0 = 0$.

Form the distinct sorted breakpoints

$$\{0, B_{\min}\} \cup \{2w_i : 0 < 2w_i < B_{\min}\}.$$

Consecutive breakpoints $a < b$ give intervals $(a, b]$ partitioning $(0, B_{\min}]$. Throughout such an interval, an item is small precisely when $2w_i \le a$: indeed, there is no breakpoint strictly between a and b, and an item with $2w_i = b$ is still big at $T = b$. Thus the small items form a fixed prefix, say of length j, and

$$W_T = P_j, \qquad q_T = q_{j+1} \qquad (a < T \le b).$$

If $q_{j+1} = m$, the interval contributes the certified target b. Otherwise its largest certified target is

$$c = \min\left\{b, \frac{P_j}{m - q_{j+1}}\right\}, \tag{1}$$

provided $c > a$; if $c \le a$, the interval contains no certified target. The right endpoint convention ensures that every nonempty feasible portion has a largest element.

There are at most $n + 1$ intervals. A monotone pointer through the sorted weights identifies all their prefix lengths in total $O(n)$ operations; each candidate then requires constant work. Returning the largest contributing candidate gives the asserted maximum. All values are computed exactly as rational numbers. □

Theorem 3.6. *For n items and $m \ge 1$ knapsacks, arbitrary-capacity bottleneck* MSSP *admits a $1/2$-approximation using $O(n \log(n+1) + m \log(m+1))$ arithmetic and comparison operations and $O(n + m)$ space. The number of operations is independent of the magnitudes of the rational input values, and the bit complexity is polynomial in the input length.*

Proof. If $B_{\min} = 0$, return the empty assignment. Otherwise sort the weights and capacities, retaining original item and knapsack indices, and apply Lemma 3.5. If no positive target is half-certifiable, then $\mathrm{OPT} = 0$ by Lemma 3.3, and the empty assignment suffices.

Let T^* be the largest target found when one exists. Its certificate and Lemma 3.2 imply that a positive-valued assignment exists. Consequently Lemma 3.3 applies and gives $T^* \ge \mathrm{OPT}$. Recover a maximum matching for the suffix of items with $2w_i \ge T^*$ using Lemma 3.4, and greedily fill the unmatched knapsacks as in Lemma 3.2. The resulting value is at least $T^*/2 \ge \mathrm{OPT}/2$.

Sorting takes the stated number of comparisons. Prefix sums, suffix matching, interval optimization, matching reconstruction, and filling all take $O(n + m)$ further operations and space. Apart from multiplication or division by two, the rational numbers used are input values, sums of at most n input weights, or such sums divided by an integer at most m. Their numerators and denominators have polynomial bit length in the binary input length. Exact arithmetic and comparisons therefore give polynomial bit complexity as well. □

4 Hardness above one half for multiple subset sum

We use Numerical Matching with Target Sums, a strongly NP-complete problem listed by Garey and Johnson [5, Appendix A, Problem SP17, p. 224]. An instance consists of positive integers

$$x_1, \ldots, x_q, \qquad y_1, \ldots, y_q, \qquad b_1, \ldots, b_q,$$

and asks whether there are permutations π, σ of $[q]$ such that

$$x_{\pi(j)} + y_{\sigma(j)} = b_j \qquad (j \in [q]).$$

A yes-instance necessarily satisfies $\sum_i x_i + \sum_i y_i = \sum_j b_j$, but this equality is not part of the standard input promise. For the reduction below, preprocess an unequal-sum instance to the fixed balanced no-instance

$$X = (1, 1), \qquad Y = (1, 1), \qquad (b_1, b_2) = (1, 3).$$

Both sides have total sum four, while the target one cannot be expressed as a sum of two positive integers. Equal-sum instances are left unchanged. Thus the preprocessed instance is balanced and has the same yes/no answer.

Lemma 4.1. *For every fixed rational $\alpha \in (1/2, 1]$ and every positive integer U, there are positive integers $A, D = O_\alpha(U)$ such that*

$$D > U, \qquad A - D > 3U, \qquad \frac{A + D + U}{2A + D} < \alpha.$$

Proof. Choose a rational $\rho \in (0, 1)$ with $(1 + \rho)/(2 + \rho) < \alpha$. Let $\beta = \alpha - (1 + \rho)/(2 + \rho) > 0$ and choose an integer constant

$$C > \max\left\{\rho^{-1}, \frac{4}{1 - \rho}, \frac{1}{\beta(2 + \rho)}\right\}.$$

Set $A = C(U + 1)$ and $D = \lceil \rho A \rceil$. Then $D > U$ and $A - D \geq (1 - \rho)A - 1 > 3U$. Moreover,

$$\frac{A + D + U}{2A + D} \leq \frac{(1 + \rho)A + (U + 1)}{(2 + \rho)A} = \frac{1 + \rho}{2 + \rho} + \frac{1}{C(2 + \rho)} < \alpha.$$

Both parameters are linear in U, with constants depending only on α. Since ρ and C are hard-coded constants for fixed α, the binary lengths of A and D are $O_\alpha(\log U)$. □

Theorem 4.2. *For every fixed rational $\alpha \in (1/2, 1]$, arbitrary-capacity bottleneck* MSSP *has no polynomial-time α-approximation unless* $\mathsf{P} = \mathsf{NP}$.

Proof. Take a balanced NMTS instance obtained by the preprocessing above and set

$$U = \max\{x_i, y_i, b_j : i, j \in [q]\}.$$

Choose A, D from Lemma 4.1. Construct q knapsacks and $2q$ items. For each x_i create a low item of weight $A + x_i$; for each y_i create a high item of weight $A + D + y_i$. Knapsack j has capacity

$$B_j = 2A + D + b_j.$$

Profits equal weights. Put

$$T = 2A + D + \min_j b_j, \qquad H = A + D + U.$$

The parameter choice gives $H < \alpha T$.

If the NMTS instance is positive, match the low and high items representing a pair $x_i + y_k = b_j$ and place them in knapsack j. Every knapsack is filled exactly to capacity, so its load is at least T. The smallest capacity is T, hence $\mathrm{OPT} = T$.

Suppose conversely that some feasible assignment gives every knapsack load strictly greater than H. Every item has weight at most H, so each knapsack contains at least two items. Three items cannot fit: each weighs at least A, while

$$3A > 2A + D + U \geq \max_j B_j$$

by $A - D > 3U$. Thus every knapsack contains exactly two items and, because there are $2q$ items, every item is used.

No two high items fit together, since $2A + 2D > 2A + D + U \geq \max_j B_j$. If p_{LL}, p_{LH}, p_{HH} count the three pair types, this first shows $p_{HH} = 0$. Use of all q low and q high items also gives

$$2p_{LL} + p_{LH} = q = 2p_{HH} + p_{LH}.$$

Therefore $p_{LL} = p_{HH} = 0$, and every knapsack contains one low and one high item.

The total item weight equals

$$q(2A + D) + \sum_i x_i + \sum_i y_i = q(2A + D) + \sum_j b_j = \sum_j B_j.$$

All items are used and no capacity is exceeded, so every knapsack is filled exactly. If knapsack j contains the items associated with x_i and y_k, exact filling yields $x_i + y_k = b_j$. These pairs solve the NMTS instance, so every assignment of value greater than H induces an NMTS solution.

Consequently, in a negative NMTS instance every assignment has value at most H, whereas in a positive instance the optimum is T and $H < \alpha T$. An α-approximation would distinguish the two cases in polynomial time. □

Corollary 4.3 (Arbitrarily small heterogeneity)**.** *Fix $\gamma > 0$ and a rational $\alpha \in (1/2, 1]$. Unless* P = NP*, no polynomial-time α-approximation exists even on instances with $2m$ positive-weight items and m positive-capacity knapsacks satisfying*

$$\frac{B_{\max}}{B_{\min}} < 1 + \gamma, \qquad \frac{w_{\max}}{w_{\min}} < 1 + \gamma.$$

Proof. In Lemma 4.1, choose the fixed rational ρ small enough that also $\rho < \gamma/2$, and increase the fixed integer C so that also $C > 2/\gamma$. All inequalities used in Theorem 4.2 remain valid. Its reduction already uses exactly two items per knapsack. Since $A = C(U + 1)$ and $D = \lceil \rho A \rceil$, the capacity ratio satisfies

$$\frac{B_{\max}}{B_{\min}} \leq \frac{2A + D + U}{2A + D} = 1 + \frac{U}{2A + D} < 1 + \frac{1}{2C} < 1 + \gamma.$$

Every item has weight at least A and at most $A + D + U$, so

$$\frac{w_{\max}}{w_{\min}} \leq 1 + \frac{D + U}{A} \leq 1 + \rho + \frac{U + 1}{A} = 1 + \rho + \frac{1}{C} < 1 + \gamma.$$

The same yes/no gap therefore proves the claim on the restricted instances. □

Corollary 4.4. *Unless* P = NP*, the optimal polynomial-time approximation threshold for arbitrary-capacity bottleneck* MSSP *is exactly* $1/2$.

Proof. Theorem 3.6 attains $1/2$, while Theorem 4.2 excludes every fixed larger rational ratio. If an algorithm guaranteed any fixed real ratio $\gamma > 1/2$, then choosing a rational $\alpha \in (1/2, \min\{\gamma, 1\}]$ would contradict the theorem. □

5 A short alternative proof for unit profits

Under unit profits, BMKP is the max–min cardinality packing problem studied by Friesen and Langston [4]. For unequal bin capacities they gave a polynomial-time FFD implementation with absolute ratio $1/2$, and proved that no larger ratio with zero additive loss is possible unless P = NP. The sorted-block proof below is different and particularly short.

Order the capacities and item weights as

$$B_1 \leq \cdots \leq B_m, \qquad a_1 \leq \cdots \leq a_n.$$

For an integer $q \geq 0$ with $mq \leq n$, the *q-block assignment* gives knapsack r the items of ranks $(r - 1)q + 1, \ldots, rq$. Prefix sums test its feasibility in $O(m)$ time. The 0-block is the empty assignment.

Lemma 5.1 (Prefix exchange)**.** *For an integer $q \geq 1$, if a unit-profit instance has a feasible assignment of value at least $2q-1$, then its q-block assignment is feasible.*

Proof. Delete items from the reference assignment until each knapsack contains exactly $2q-1$ items. Denote the resulting pairwise disjoint item sets by $O_1, \ldots, O_m$, indexed in nondecreasing order of capacity.

Run the block algorithm greedily. Immediately before it fills knapsack r, exactly $q(r-1)$ items have been removed. The union $O_1 \cup \cdots \cup O_r$ originally contains $r(2q-1)$ items, so at least

$$r(2q-1) - q(r-1) = r(q-1) + q > r(q-1)$$

of them remain. Hence some O_s with $s \leq r$ still contains at least q items. Choose any q of those items. Their total weight is at most B_s, and therefore at most B_r. The greedy block consists of the q lightest remaining items, so its total weight is no larger. Thus knapsack r is filled feasibly. The argument applies at every step. □

Theorem 5.2. *There is a polynomial-time $1/2$-approximation for unit-profit* BMKP *with arbitrary capacities.*

Proof. Test all block sizes $q = 0, 1, \ldots, \lfloor n/m \rfloor$ and return the largest feasible one, say Q. Let $T = \mathrm{OPT}$. If $T = 0$, the claim is immediate. Otherwise T is a positive integer, $mT \leq n$, and

$$q^* = \lceil T/2 \rceil \leq T \leq \lfloor n/m \rfloor.$$

Thus q^* is among the tested values. Since $2q^* - 1 \leq T$, Lemma 5.1 shows that the q^*-block assignment is feasible. Consequently $Q \geq q^* \geq T/2$.

After sorting and computing prefix sums, all block tests together take $O(n+m)$ arithmetic and comparison operations: there are at most $\lfloor n/m \rfloor + 1$ values of q, each tested on m knapsacks. The total running time is $O(n \log n + m \log m)$ arithmetic and comparison operations. With binary-encoded rational weights and capacities, every prefix sum has polynomial bit length, so the bit complexity is polynomial as well. □

Corollary 5.3 (A posteriori certificate)**.** *The returned value satisfies $Q \leq \mathrm{OPT} \leq 2Q$.*

Proof. The lower bound follows from feasibility. If $Q < \lfloor n/m \rfloor$, then the $(Q+1)$-block assignment is infeasible. The contrapositive of Lemma 5.1, with $q = Q+1$, yields $\mathrm{OPT} < 2Q+1$, hence $\mathrm{OPT} \leq 2Q$ because OPT is integral. If $Q = \lfloor n/m \rfloor$, the item count itself gives $\mathrm{OPT} \leq \lfloor n/m \rfloor = Q$. □

Example 5.4. Take capacities $10, 10$ and four unit-profit items of weights $4, 4, 6, 6$. Pairing a 4 with a 6 in each knapsack gives optimum value two. The two-block assignment instead uses blocks $(4, 4)$ and $(6, 6)$ and is infeasible, whereas the one-block assignment is feasible. Thus the algorithm can return one when the optimum is two.

6 Concluding remarks

Chen et al. [3, Section 7, p. 60:18] also ask for a $(2/3 - \varepsilon)$-approximation when the number of distinct capacities is bounded by a constant. The reduction above uses a potentially growing number of capacity values and therefore does not settle that question.

References

[1] A. Caprara, H. Kellerer, and U. Pferschy, The multiple subset sum problem, *SIAM Journal on Optimization* **11** (2000), no. 2, 308–319, doi:10.1137/S1052623498348481.

[2] A. Caprara, H. Kellerer, and U. Pferschy, A PTAS for the multiple subset sum problem with different knapsack capacities, *Information Processing Letters* **73** (2000), no. 3–4, 111–118, doi:10.1016/S0020-0190(00)00010-7.

[3] L. Chen, T. Hu, Y. Mao, Y. Chen, L. Mei, A. Zhang, G. Chen, and G. Zhang, Near-tight approximation algorithms for bottleneck multiple knapsack problems, in S. Bhattacharya, D. Nanongkai, M. Benedikt, and G. Puppis (eds.), *53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026)*, LIPIcs 374, 60:1–60:19, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2026, doi:10.4230/LIPIcs.ICALP.2026.60; full version arXiv:2605.05233.

[4] D. K. Friesen and M. A. Langston, Bin packing: On optimizing the number of pieces packed, *BIT Numerical Mathematics* **27** (1987), no. 2, 148–156, doi:10.1007/BF01934179.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979; see Appendix A, Numerical Matching with Target Sums, Problem SP17, p. 224.

[6] B. L. Deuermeyer, D. K. Friesen, and M. A. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM J. Algebraic Discrete Methods*, 3(2):190–196, 1982. https://doi.org/10.1137/0603019.

[7] G. J. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Oper. Res. Lett.*, 20(4):149–154, 1997. https://doi.org/10.1016/S0167-6377(96)00055-7.

[8] S. F. Assmann, D. S. Johnson, D. J. Kleitman, and J. Y.-T. Leung. On a dual version of the one-dimensional bin packing problem. *J. Algorithms*, 5(4):502–525, 1984. https://doi.org/10.1016/0196-6774(84)90004-X.

[9] K. Jansen and R. Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoret. Comput. Sci.*, 306(1–3):543–551, 2003. https://doi.org/10.1016/S0304-3975(03)00363-3.

[10] C. Chekuri and S. Khanna, A polynomial time approximation scheme for the multiple knapsack problem, *SIAM Journal on Computing* **35** (2006), no. 3, 713–728, doi:10.1137/S0097539700382820.

[11] N. Bansal and M. Sviridenko, The Santa Claus problem, in *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 2006)*, 31–40, doi:10.1145/1132516.1132522.

[12] H. Hummel, Maximin shares in hereditary set systems, *ACM Transactions on Economics and Computation* **13** (2025), no. 3, Article 12, 1–33, doi:10.1145/3727149; arXiv:2404.11582.

[13] L. Rohwedder, A. Rouhani, and L. Wennmann, Cost preserving dependent rounding for allocation problems, in *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, LIPIcs 334, 127:1–127:20, doi:10.4230/LIPIcs.ICALP.2025.127; arXiv:2502.08267v2.