

A Comprehensive Technical Analysis of the Karmarkar-Karp (1982) Bin Packing Algorithm

Research Report

November 20, 2025

Abstract

This report provides a detailed technical deconstruction of the seminal 1982 algorithm by Narendra Karmarkar and Richard M. Karp for the one-dimensional Bin Packing Problem (1DBPP). It explores the algorithm's foundational role in establishing the $O(\log^2 OPT)$ additive integrality gap, analyzing its core mechanisms: geometric grouping, the configuration linear program, the use of the Knapsack problem as a separation oracle, and the iterative rounding procedure.

Contents

1	Introduction: The 30-Year Benchmark	2
2	Preliminaries: The Gilmore-Gomory LP Relaxation	2
2.1	Structure	2
2.2	The Computational Challenge	2
2.3	The Core Problem	3
3	The KK Mechanism (Part 1): Taming the Constraints via Grouping	3
3.1	Elimination of Small Items	3
3.2	Geometric Grouping	3
4	The KK Mechanism (Part 2): Solving the Exponential LP	3
4.1	The Dual LP	3
4.2	The Ellipsoid Method & The Separation Oracle	3
4.3	The "Aha!" Moment: The Oracle is the Knapsack Problem	4
4.4	The Full Stack	4
5	The KK Mechanism (Part 3): Iterative Rounding & The $O(\log^2 OPT)$ Guarantee	4
5.1	The Iterative Loop	4
5.2	Source of the $O(\log^2 OPT)$ Guarantee	4
6	Conclusion: The KK82 Legacy	5
7	Comparative Summary Table	5

1 Introduction: The 30-Year Benchmark

The one-dimensional Bin Packing Problem (1DBPP) asks for the minimum number of unit-capacity bins to pack n items, each with a size $s_i \in (0, 1]$. While simple offline heuristics like First-Fit Decreasing (FFD) are fast ($O(n \log n)$) and provide good multiplicative guarantees (e.g., $A(I) \leq \frac{11}{9}OPT(I) + 4$), the frontier of theoretical research focuses on *additive* guarantees.

For over three decades, the seminal 1982 algorithm by Narendra Karmarkar and Richard M. Karp (KK82) stood as the undisputed benchmark in this area. It was the first algorithm to break from multiplicative errors and provide a polynomial-time solution with a provably small *additive* error, a monumental breakthrough at the time.

The KK82 algorithm achieves a final packing $A(I)$ using at most:

$$A(I) \leq OPT(I) + O(\log^2 OPT(I)) \quad (1)$$

bins. This $O(\log^2 OPT)$ term—an error that scales polylogarithmically with the *optimal solution size* rather than linearly—established the additive integrality gap for the standard LP relaxation. This result became the “benchmark to beat” until the work of Rothvoß (2013) and Hoberg & Rothvoß (2015), which refined the bound to $O(\log OPT)$.

This analysis deconstructs the core mechanisms of the KK82 algorithm to explain *how* this $O(\log^2 OPT)$ guarantee is achieved.

2 Preliminaries: The Gilmore-Gomory LP Relaxation

Like the subsequent breakthroughs, the KK82 algorithm is built upon the foundation of the **Gilmore-Gomory Linear Program (LP) Relaxation**, often called the Configuration LP.

2.1 Structure

A “pattern” or “configuration” p is a multiset of items that can fit into a single bin. Let \mathcal{P} be the set of all possible valid patterns. The LP is formulated with a variable x_p for each pattern, representing the (fractional) number of times that pattern is used.

- Let m be the number of distinct item *types*.
- Let b_i be the number of items of type i that must be packed.
- Let t_{pi} be the number of items of type i in pattern p .

The LP relaxation is formulated as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{p \in \mathcal{P}} x_p \quad (\text{Minimize total bins}) \\ \text{subject to} & \sum_{p \in \mathcal{P}} t_{pi} x_p \geq b_i \quad (\text{for each item type } i = 1..m) \\ & x_p \geq 0 \quad (\text{for all patterns } p \in \mathcal{P}) \end{array}$$

2.2 The Computational Challenge

The LP has two main computational hurdles:

1. **Exponential Variables:** The number of patterns $N = |\mathcal{P}|$ can be exponential in the number of items n .
2. **Many Constraints:** The number of constraints m (distinct item types) can be as large as n .

2.3 The Core Problem

The LP yields an optimal *fractional* solution, OPT_f (also denoted $\text{lin}(I)$). The entire challenge is to "round" this fractional solution \mathbf{x} into an integer packing $A(I)$ while introducing only a tiny additive error. The $O(\log^2 OPT)$ term is, precisely, the "price of integrality" paid during this rounding process.

3 The KK Mechanism (Part 1): Taming the Constraints via Grouping

The first challenge, having $m = n$ constraints, is handled by a pre-processing step that reduces the number of distinct item types. This is a "main innovation" of the KK82 paper.

3.1 Elimination of Small Items

First, a threshold g (e.g., $g = 1/n$) is chosen, and all "small" items $s_i \leq g$ are removed and set aside. This is critical because it ensures the remaining items are "large," meaning any bin can contain at most $1/g$ (e.g., n) of them. These small items are added back at the very end, fitting into existing gaps.

3.2 Geometric Grouping

The remaining n' items are "grouped" to reduce the number of distinct types m from $O(n')$ to $O(\text{poly}(\log n'))$. This works in two phases:

1. **Logarithmic Partition:** Items are partitioned into groups I_r based on their size, where $I_r = \{\text{items } i \mid s_i \in (1/2^{r+1}, 1/2^r]\}$.
2. **Linear Grouping:** Within *each* group I_r , items are sorted and further partitioned into k buckets (where k is a parameter). The sizes of all items in a bucket are then rounded *up* to the size of the largest item in that bucket.

This grouping process creates a new, simpler instance K with a manageable (polynomial) number of item types m . The "cost" of this rounding-up adds a small, bounded error in each iteration, which contributes to the final $O(\log^2 OPT)$ term.

4 The KK Mechanism (Part 2): Solving the Exponential LP

After grouping, the LP has m (polynomial) constraints, but still N (exponential) variables. KK82 solves this using a now-classic approach based on duality.

4.1 The Dual LP

The algorithm considers the *dual* of the Configuration LP. By duality principles, variables and constraints are swapped. The dual has m variables (polynomial) but N constraints (exponential).

4.2 The Ellipsoid Method & The Separation Oracle

An LP with a polynomial number of variables and an exponential number of constraints can be solved in polynomial time using the **Ellipsoid Method**, provided one can supply a **Separation Oracle**. The oracle is given a potential solution \mathbf{y} (a vector of m dual variables) and must either certify that \mathbf{y} is feasible (satisfies all N constraints) or return a constraint p that is violated.

4.3 The "Aha!" Moment: The Oracle is the Knapsack Problem

The dual constraints are of the form $A_p^T \mathbf{y} \leq 1$ for each pattern p . The oracle's job is to find a pattern p that *maximizes* $A_p^T \mathbf{y}$.

If we interpret the dual variables y_i as the "**profit**" or "value" of packing an item of type i , and the item size s_i as its "**weight**", then $\max_p(A_p^T \mathbf{y})$ is the task of finding the **maximum-profit** set of items that can fit into a single bin (i.e., total weight ≤ 1). This is the exact definition of the **0/1 Knapsack Problem**.

4.4 The Full Stack

The Knapsack Problem is NP-hard. However, it admits a Fully Polynomial-Time Approximation Scheme (FPTAS). The Ellipsoid Method is robust enough to work with an *approximate* separation oracle. Therefore, the LP is solved by "stacking" these algorithms: the Ellipsoid Method is called, and at each of its steps, it calls the Knapsack FPTAS to act as its oracle.

5 The KK Mechanism (Part 3): Iterative Rounding & The $O(\log^2 OPT)$ Guarantee

Solving the LP gives a fractional solution \mathbf{x} . The final step is to round this \mathbf{x} to an integer packing. KK82 does this using an iterative process that runs in a loop for $t = O(\log n)$ iterations.

5.1 The Iterative Loop

In each iteration i :

1. **Start with Residual Instance:** Begin with the set of items I_i remaining from the last iteration.
2. **Group:** Apply the Geometric Grouping to I_i , creating instance J_i and a "discard" instance J'_i . The items in J'_i are packed into new bins, creating the first source of additive error, Y_i .
3. **Solve LP:** Solve the Configuration LP for instance J_i to get a fractional solution \mathbf{x} .
4. **Partial Rounding:** "Buy" the integer part of the solution. For every pattern p where $x_p \geq 1$, pack $\lfloor x_p \rfloor$ bins with that pattern. These items are permanently removed.
5. **Define New Residual:** The *new* residual instance I_{i+1} consists of all items that were part of the fractional remainders, $\mathbf{x} - \lfloor \mathbf{x} \rfloor$.
6. **Repeat:** The key insight is that the *total size* of the residual problem decreases geometrically at each step, guaranteeing the loop terminates in $O(\log n)$ iterations.

5.2 Source of the $O(\log^2 OPT)$ Guarantee

The final solution cost is the sum of the fractional optimum ($\text{lin}(I) \leq OPT(I)$) plus all the bins added during the iterative process.

- **Number of Iterations:** $t = O(\log n)$.
- **Error per Iteration:** In each iteration i , the algorithm adds Y_i bins from packing the "grouped-out" items. This error is bounded by $Y_i \leq O(\log(1/g))$, which is $O(\log n)$.
- **Total Additive Error:**

Since n is bounded by $OPT(I)$ (as $s_i > g$), $\log n$ is $O(\log OPT(I))$. This gives the final $OPT(I) + O(\log^2 OPT(I))$ guarantee.

6 Conclusion: The KK82 Legacy

The Karmarkar-Karp algorithm provides a canonical example of the trade-off between complexity and solution quality.

- **FFD (Phase 1):** $O(n \log n)$ time, but a *multiplicative* error ($\frac{11}{9}OPT$).
- **KK82 (Phase 2):** A high-order polynomial time (e.g., $O(n^8)$) but achieves a "near-perfect" *additive* error ($OPT + O(\log^2 OPT)$).

The Karmarkar-Karp algorithm is not a single function but a complex, multi-stage "stack" of advanced techniques. Its $O(\log^2 n)$ bound on the integrality gap, derived from its specific iterative rounding technique, created a theoretical barrier that stood for 30 years until the recent introduction of discrepancy theory methods by Rothvoß.

7 Comparative Summary Table

Algorithm	Type	Time Complexity	Performance Guarantee
FFD	Offline Heuristic	$O(n \log n)$	$A(I) \leq \frac{11}{9}OPT(I) + 4$
KK82 (Additive)	AFPTAS	High-order poly ($O(n^8)$)	$A(I) \leq OPT(I) + O(\log^2 OPT(I))$
Rothvoß (2013)	AFPTAS	Polynomial	$A(I) \leq OPT + O(\log OPT \cdot \log \log OPT)$
Hoberg & Rothvoß	AFPTAS	Polynomial	$A(I) \leq OPT(I) + O(\log OPT(I))$

Table 1: Comparison of Bin Packing Approximation Algorithms