Probabilistic Analysis of Knapsack Core Algorithms

Rene Beier*
rbeier@mpi-sb.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Berthold Vöcking†
voecking@ls2.cs.uni-dortmund.de
Fachbereich Informatik
Universität Dortmund, Germany

Abstract

We study the average-case performance of algorithms for the binary knapsack problem. Our focus lies on the analysis of so-called *knapsack core algorithms*, the predominant algorithmic concept used in practice. Core algorithms start with the computation of an optimal fractional solution that contains at most one variable has a fractional value, and then exchange items until an optimal integral solution is found. The idea is that in many cases the optimal integral solution should be close to the fractional one such that only few items need to be exchanged. Despite the well known hardness of the knapsack problem on worst-case instances, practical studies show that knapsack core algorithms can solve large scale instances very efficiently. For example, they exhibit almost linear running time on purely random inputs.

In this paper, we present the first theoretical result on the running time of core algorithms that comes close to the results observed in practice. We prove an upper bound of $n\,\mathrm{polylog}(n)$ on the expected running time for random instances with n item and uniformly random profits and weights. This significantly improves a previous bound of $O(n^4)$ for such instances. This analysis is based on a different algorithmic concept, however. The previously best known upper bound for the running time of a core algorithm is polynomial as well, but the degree of the polynomial is at least a large three digit number. In addition to uniformly random instances, we investigate harder instances in which profits and weights are correlated. Here we can prove a tradeoff describing how the correlation between profits and weights influences the running time.

*Supported in part by the IST program of the EU under contract number IST-1999-14186 (ALCOM-FT).
†Supported in part by DFG grant Vo889/1-1.

1 Introduction

This paper is concerned with a probabilistic analysis of the 0/1 knapsack problem. Given n objects with weights $w_1,\ldots,w_n$ and profits $p_1,\ldots,p_n$ and a knapsack capacity B, the knapsack problem is to

$$\text{maximize } \sum_{j\in[n]} p_j x_j$$

subject to the constraints

$$\textstyle\sum_{j\in[n]} w_j x_j \le B, \text{ and } x_j \in \{0,1\}, \text{ for } j \in [n],$$

where $[n] = \{1,2,\ldots,n\}$. We assume that the p_i and w_i are drawn independently, uniformly at random from $[0,1]$. Following the conventions in previous analyses [13, 11], the value of n is assumed to be chosen according to the Poisson distribution with parameter N. As in the previous work, we assume $B = \beta N$, for some constant $\beta \in (0, \frac{1}{2})$. The motivation for these two assumptions is to simplify the presentation of the analysis by removing insignificant but quite disturbing dependencies as well as excluding degenerated cases like $B = \frac{N}{2}$ in which case either all or almost all items are contained in the knapsack.

Our focus lies on the analysis of so-called *core algorithms* that have been proven to be the most efficient algorithms in numerous practical studies [1, 9, 10, 16]. This concept was suggested by Balas and Zemel [1]. The idea is to start with the computation of an optimal fractional solution where at most one variable has a fractional value and then to exchange some of the items until an optimal integral solution is found. The set of items that are candidates to be exchanged, is called the *core*, and the hope is that the size of the core is relatively small for “typical instances”. As a first step towards analyzing core algorithms, Lueker proved an upper bound on the expected gap between the optimal integral and the optimal fractional solution [13]. Based on this result, Goldberg and Marchetti-Spaccamela [11] were able to prove structural properties of the core resulting in the following bound on the running time of a Las Vegas type core algorithm. For every fixed $k > 0$, with probability at least $1 - 1/k$, the running time of their algorithm does not exceed a specified upper bound that is polynomial in the number of items. However, the degree of this polynomial is quite large, the leading constant in the exponent is at least a three digit number. Even more dramatically, the degree of the polynomial grows with the reciprocal of the failure probability like $\sqrt{k}\log(k)$. Observe that this kind of tail bounds does not allow to conclude any sub-exponential upper bound on the expected running time. This work was later extended to the multidimensional knapsack problem by Dyer and Frieze [5].

Better bounds on the running time are only known for an algorithm by Nemhauser and Ullmann [15]. This algorithm, however, does not follow the core concept but instead applies a dominance criterion to reduce the search space. A subset $S \subseteq [n]$ with weight $w(S) = \sum_{i\in S} w_i$ and profit $p(S) = \sum_{i\in S} p_i$ *dominates* another subset $T \subseteq [n]$ if $w(S) \le w(T)$ and $p(S) \ge p(T)$. For simplicity, let us assume that all sets have different profits. Then no subset dominated by another subset can be an optimal solution to the knapsack problem, regardless of the knapsack capacity. Consequently, it suffices to consider only those sets that are not dominated by any other set, the so-called *dominating sets*. In a recent study by Beier and Vöcking [2], it was shown that, for uniformly random instances, the expected number of dominating sets is bounded by $O(n^3)$. This result implies an upper bound of $O(n^4)$ on the expected running time of the Nemhauser/Ullmann algorithm. In fact, experiments show that the running time of this algorithm behaves approximately like n^3. Core algorithms, however, show a much better performance in experiments [9, 10, 16]. Their running time on uniformly random instances is almost linear. Interestingly, the most successful implementations of core algorithms [9, 16] additionally exploit domination in order to decrease the number of subsets generated by the core.

Our contribution. The motivation for our study is to understand and explain the efficiency of knapsack core algorithms on random instances. We present the first theoretical results on the running time of a core algorithm coming close to the results observed in practice. In particular, we prove an upper bound of $n\,\mathrm{polylog}(n)$ on the expected running time of a core algorithm on uniformly random instances. In addition, following a recent trend in practical studies ([9],[17]), we investigate also harder instances in which profits and weights are correlated. Here we prove a tradeoff describing how the running time increases with the correlation.

As in the most efficient implementations, the algorithms underlying our analysis combine the core and the domination concept. However, certain details of this combination are quite different. In particular, we use Goldberg and Marchetti-Spaccamela's definition of the core [11] and combine it with the enumeration method for dominating sets by Nemhauser and Ullmann [15]. Somewhat surprisingly, this approach does not only enable us to do a rigorous mathematical analysis, but also yields a new implementation of a core algorithm that outperforms the best previous implementations by orders of magnitudes on well studied benchmark instances.

Outline. In Section 2 we start with a short overview of the techniques and results from previous studies that we apply in our analysis. In Section 3, we present an algorithm with almost linear expected running time on uniformly random instances. This algorithm, however, has some small failure probability, that is, it might compute a suboptimal solution with polynomially small probability. In Section 4, we show how failures can be handled without increasing the expected running by more than a constant factor. In Section 5, we describe an average-case model for so-called weakly correlated instances and generalize our analysis towards this model. Finally, we briefly present a few experimental results in order to document also the practical advantages of our algorithm. This part has been shifted to the Appendix.

2 Tools and Techniques

2.1 The core problem

Core algorithms start by computing an optimal solution for the *relaxed or fractional* knapsack problem. In this problem, the constraints $x_i \in \{0,1\}$ are replaced by $0 \leq x_i \leq 1$. An optimal solution to the fractional problem can be found by the following greedy algorithm [1]. Starting with the empty knapsack and we add items one by one in order of non-increasing profit-to-weight ratio[1]. The algorithm stops when coming to the first item that would exceed the capacity bound B. This item is called *break item* and the computed knapsack filling not including the *break item* is called *break solution*. It has been shown that the break solution, and hence also the fractional solution, can be found in linear time by solving a weighted median problem [1]. For a geometrical interpretation, let us map each item w_i, p_i to the point (w_i, p_i). Then the greedy algorithm described above can be pictured as rotating a ray clockwise around the origin starting from the vertical axis and inserting all swept items until the insertion of the current item would exceeds the capacity. The ray defined by the break item is called the *Dantzig ray* (see Figure 1).

There is a strong motivation to start with the computation of the break solution. It has been observed in practical studies that the break solution is quite similar to the optimal integral solution in the sense that the they differ in only a few variables. So only a few items need to be exchanged to transform the break solution to an optimal solution. The problem is, of course, that we do not know in advance which items to exchange. Therefore, one uses a appropriate subset of candidate items, called the core. Assuming that the core contains all items that needs to be exchanged to obtain an optimal solution, items outside the core can be fixed to the value given by the break solution. There are various different ways to define the core.

Goldberg and Marchetti-Spaccamela [11] define the core as follows. Assume the items are given in order of non-increasing profit-to-weight ratio and let b denote the index of the break item. The solution vector for the fractional problem has the form $\overline{X} = (1, \ldots, 1, f, 0, \ldots, 0)$ where $f \in (0,1]$ is the entry at position b. For any feasible integer solution $X = (x_1, \ldots, x_n)$, define $ch(X) = \{i \in [n] : \overline{x_i} \neq x_i\}$, i.e., $ch(X)$ is the set of items on which $\overline{X}$ and X do not agree. When removing an item from the break solution the freed capacity can be used to include

[1] In previous studies, the *profit-to-weight ratio* of an item is also called its *density*. In this paper, the term *density* solely refers to the density function describing the probability distributions of the profits.

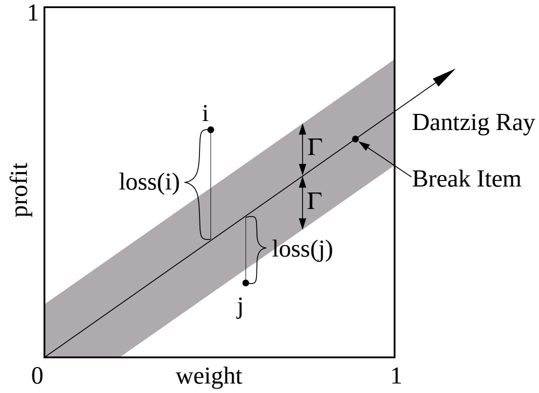

Figure 1: Dantzig ray and Break Item. The core stripe has vertical width 2Γ with Γ denoting the integrality gap. The loss of an item is the vertical distance to the Dantzig ray.

items that have profit-to-weight ratio at most $r := p_b/w_b$ corresponding to the slope of the Dantzig ray. Obviously, replacing items by other items with same weight but smaller profit-to-weight ratio leads to some loss in profit. This loss can only be compensated by filling the knapsack with more weight, that is, by decreasing the wasted capacity. One can quantify the loss that is incurred by replacing a “valuable” item ($p_i/w_i \geq r$) by a “cheap” item ($p_i/w_i \leq r$) as follows. Based on the slope $r := p_b/w_b$ of the Dantzig ray define $\text{loss}(i) = |p_i - rw_i|$, i.e., $\text{loss}(i)$ corresponds to the vertical distance of item i to the Dantzig ray. Goldberg and Marchetti-Spaccamela proved that the difference in profit between any feasible integer solution X and the optimal fractional solution $\overline{X}$ can be expressed as

$$\sum_{i\in[n]} \bar{x}_i p_i - \sum_{i\in[n]} x_i p_i = r\left(B - \sum_{i\in[n]} x_i w_i\right) + \sum_{i\in ch(X)} \text{loss}(i) \quad . \tag{1}$$

The first term on the right hand side corresponds to an unused capacity of the integer solution X whereas the second term is due to the accumulated loss of all items in $ch(X)$. Define $P(X) = \sum_{i\in[n]} x_i p_i$, and let X^* be an optimal integral solution. The integrality gap $\Gamma = P(\overline{X}) - P(X^*)$ gives an upper bound for the accumulated loss of X^* and therefore an upper bound for the individual loss of each item in $ch(X^*)$. Thus, all items in $ch(X^*)$ have vertical distance at most Γ from the Dantzig ray. Hence, the core can be defined to consist of all items with loss at most Γ. The value of Γ can be obtained, e.g., by guessing or by adding the items in order of increasing loss to the core until the loss of the next element is not smaller than the difference in profit between the optimal fractional solution and the best integer solution computed so far.

2.2 Properties of the core

Lueker’s analysis [13] bounds the expected integrality gap for uniformly random instances. In particular, he shows $\mathbf{E}[\Gamma] = O(\frac{\log^2 N}{N})$. Goldberg and Marchetti-Spaccamela [11] observe that this analysis can easily be generalized to obtain exponential tail bounds on Γ.

Lemma 1. *There is constant k_0 such that for every $\alpha \leq \log^4 N$,* $\mathbf{Pr}\left[\Gamma \geq k_0 \alpha \frac{\log^2 N}{N}\right] \leq 2^{-\alpha}$.

In words, the integrality gap does not exceed $O(\frac{\log^2 N}{N})$, with high probability. Let us remark that the value of k_0 depends on β, the constant determining the knapsack capacity. The constraint $\alpha \leq \log^4 N$ satisfies our requirements but, in general, the tail bound can be extended to hold for every $\alpha \leq N^\kappa$, for every fixed $\kappa < \frac{1}{2}$. We will use this bound to obtain a tail bound on the number of items in the core.

Goldberg and Marchetti-Spaccamela [11] use this tail bound for their probabilistic analysis of a core algorithm. Let us briefly sketch their analysis and explain why it fails to bound the expected running time. It is the basis for our modifications to the core algorithm. Let X denote the number of *core items*, i.e., items with vertical distance at most Γ from the Danzig ray. Basically, the expected value of X corresponds to the area covered by the Γ-region around the Danzig ray times N, that is, $\mathbf{E}[X] \approx 2\Gamma N$. (There are some slight dependencies that we neglect here. In fact, the Dantzig ray has some tendency to fall into dense regions.) If Γ is fixed then this number is sharply concentrated. Furthermore, because of Lemma 1 the random variable Γ is sharply concentrated around $O(\frac{\log^2 N}{N})$. Combining these results, it follows $X = O(\log^2 N)$, with high probability. Consequently, the number of sets generated by the core items is $2^X = n^{O(\log N)}$. Obviously, enumerating all these sets cannot be done in polynomial time. Therefore, Goldberg and Marchetti-Spaccamela use a further trick to significantly reduce the number of sets enumerated. They use a filtering mechanism that we call *loss filter*. This mechanism generates only those sets with loss at most Γ. For every fixed $\varepsilon > 0$, they show that the number of sets generated is $2^{O(\sqrt{X})} = n^{O(1)}$, with probability $1-\varepsilon$. Unfortunately, the degree of this polynomial grows rapidly with the reciprocal of the failure probability ε. Roughly speaking, this is because the random variable X has moved to the exponent so that small deviations in X might cause large deviations in the running time. For this reason, the analysis of Goldberg and Marchetti-Spaccamela fails to bound the expected running time. Moreover, constant factors lost in the analysis of X and Γ go directly to the exponent of the polynomial running time bound.

Instead, we will replace the loss filter by a better filtering mechanism reducing the number of enumerated sets from $2^{O(\sqrt{X})}$ to $nX^{O(1)}$. This way, we will be able to bound the expected running time by $n \operatorname{polylog} n$. Our filtering mechanism is based on the following dominance criterion.

2.3 Dominating sets and the Nemhauser/Ullmann algorithm

Fix a knapsack instance with n items. Recall, that a subset $S \subseteq [n]$ with weight $w(S) = \sum_{i \in S} w_i$ and profit $p(S) = \sum_{i \in S} p_i$ *dominates* another subset $T \subseteq [n]$ if $w(S) \leq w(T)$ and $p(S) \geq p(T)$. For simplicity, let us assume that all sets have different profits[2]. Sets that are dominated by other sets cannot be optimal solutions to the knapsack problem, regardless of the specified knapsack capacity. Consequently, it suffices to consider those sets that are not dominated by any other set, the so-called *dominating sets*. In other terminology, dominating sets are *Pareto-optimal solutions*, i.e., solutions that cannot be improved in weight and profit simultaneously by other solutions.

Nemhauser and Ullman [15] introduced the following elegant algorithm that computes the sequence of all dominating sets in a very efficient way. For $i \in [n]$, let $\mathcal{S}(i)$ be the sequence of dominating subsets over the items $1, \ldots, i$. The sets in $\mathcal{S}(i)$ are assumed to be listed in increasing order of their weights. Given $\mathcal{S}(i)$, the sequence $\mathcal{S}(i+1)$ can be computed from $\mathcal{S}(i)$ as follows. First duplicate all subsets in $\mathcal{S}(i)$ and then add item $i+1$ to each of the duplicated sets. In this way we obtain two ordered sequences of sets. Now we merge the two sequences by removing those subsets that are dominated by any other set in the union of the two sequences. The result is the ordered sequence $\mathcal{S}(i+1)$ of dominating sets over the items $1, \ldots, i+1$.

The sequence $\mathcal{S}_{i+1}$ can be calculated from the sequence $\mathcal{S}_i$ in time linear in the length of $\mathcal{S}_i$, that is, linear in the number of dominating subsets over the items $1, \ldots, i$. Since the optimal knapsack filling is described by one of the subsets in the list $\mathcal{S}_n$, namely the subset with largest weight not exceeding the capacity B, generating $\mathcal{S}_n$ solves the knapsack problem. This yields the following lemma.

Lemma 2. *For every $i \in [n]$, let $q(i)$ denote the number of dominating sets over items $1, \ldots, i$ and assume $\mathbf{E}[q(i+1)] \geq \mathbf{E}[q(i)]$. The Nemhauser/Ullman algorithm computes an optimal knapsack filling in expected time $O(\sum_{i=1}^{n-1} \mathbf{E}[q(i)]) = O(n \cdot \mathbf{E}[q(n)])$.* □

In [2] it is shown that $\mathbf{E}[q(n)] = O(n^3)$ for uniformly random instances. Hence, the expected running time for these instances is $O(n^4)$. Furthermore, an analysis is presented which allows adversarial weights and randomly drawn profits that possibly follow different probability distributions.

---

[2] The probability for the existence of two sets with same profit under the uniform distribution is zero anyway.

Lemma 3. *Suppose profit p_i is drawn according to a continuous, nonnegative probability distribution with density function $f_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Suppose $\mu \geq \max_{i\in[n]} (\mathbf{E}[p_i])$ and $\phi \geq \max_{i\in[n]} \left(\sup_{x\in\mathbb{R}_{\geq 0}} (f_i(x))\right)$. Then there is a constant $c \in \mathbb{R}_{>0}$ such that the expected number of dominating sets is $\mathbf{E}[q] \leq c\phi\mu n^4 + 1$.*

Combining the two lemmas, it follows that the expected running time of the Nemhauser/Ullmann algorithm is $O(\phi\mu n^5)$. Observe that profits can be rescaled such that $\mu = 1$, unless there is an item whose expected profit is unbounded. The interesting parameter is the maximum density ϕ. Our bound on the expected number of dominating sets and, hence, on the expected running time increases linearly with the maximum density. Saying it the other way around, the less randomness is available, the larger is our estimate on the expected running time.

3 Algorithm FastCore 1: Filtering Dominated Solutions

Our first algorithm has almost linear running time but fails with some small probability. We use a static core consisting of all items with loss at most $d = c_0 \log_3 N$ for some constant c_0. Thus, core items correspond to points in region A and B (see Figure 2a). The Nemhauser/Ullmann algorithm is used to enumerate all dominating sets over the core items. Items outside the core which are included in the break solution, corresponding to points in region C, are (virtually) added to each dominating set. Then among all dominating sets with weight at most B the most profitable one is selected. If the profit of this set differs from the profit of the optimal fractional solution by at most d then we have a proof of optimality and output this set, otherwise we output *failure*.

The correctness of this algorithm follows from the discussions in the previous sections. The algorithm fails only if the chosen height of the core stripe is smaller than the integrality gap, that is, if $d < \Gamma$. Notice, that we might have found the optimal solution but we have no proof of optimality. From Lemma 1, it follows that $\mathbf{Pr}[\Gamma > d] \leq 2^{-c_0 \log N/k_0} = N^{-c_0/k_0}$.

In the following analysis, we will show that the expected running time of this algorithm is $O(N \operatorname{polylog} N)$. Unfortunately, the analysis is quite technical due to vast dependencies between different random variables. The main proof idea, however, is rather simple and elegant. Thus, before going to the details, let us present this idea and sketch the analysis.

Using tail bounds, we prove that the number of core items is $O(\log^3 N)$, with high probability. Hence, roughly speaking, the algorithm of Nemhauser and Ullmann has to deal only with $O(\log^3 N)$ items. The running time of this algorithm depends polynomially on the number of core items and is linear in ϕ, i.e., the maximum density of the probability distributions for the profits of the core items. When calculating ϕ, we condition on the fact that these items are core items, and, hence, correspond to points in the core stripes A or B, depending on whether the item is included or excluded in the break solution. The break item is handled separately.

The height of each stripe is $d = \Theta(\frac{\log^3 N}{N})$. Thus, intuitively, the profit of a "typical" core item follows a uniform distribution over an interval of width $\Theta(\frac{\log^3 N}{N})$. More formally, we will show that the conditional probability distributions for the profits of almost all core items have density at most N. Thus, for these items $\phi \leq N$. Consequently, by Lemma 3, the expected number of dominating sets is at most $\phi \operatorname{polylog} N = N \operatorname{polylog} N$, and so is the running time. Thus, interestingly, the linear term in the running time formula is not due the number of core items but due to the density of their profit distributions.

Theorem 4. *For uniformly random instances, algorithm FastCore1 computes an optimal solution with probability at least $1 - N^{-c_0/k_0}$. The expected running time of this algorithm is $O(N \mathit{polylog} N)$.*

Proof. It only remains to prove the upper bound on the expected running time. Let A and B denote the set of points above and below the Dantzig ray $\mathcal{R}$ with vertical distance at most $d = c_0/N \cdot \log^3 N$ from $\mathcal{R}$, respectively. Define

$$G = \{(w,p) \in A \,|\, rw > 1 - \tfrac{1}{N}\} \cup \{(w,p) \in B \,|\, rw < \tfrac{1}{N}\} \cup \{(w,p) \in B \,|\, rw - d > 1 - \tfrac{1}{N}\} \ ,$$

where $r = p_b/w_b$ denotes the slope of the Dantzig ray. Furthermore, define $F = \{(x,y) \in \mathbb{R}^2 : x \in [0,1] \wedge y \in [0, \frac{1}{N}] \cup [1 - \frac{1}{N}, 1]\}$. Figure 2b explains the situation. The motivation for these definitions will become clear soon.

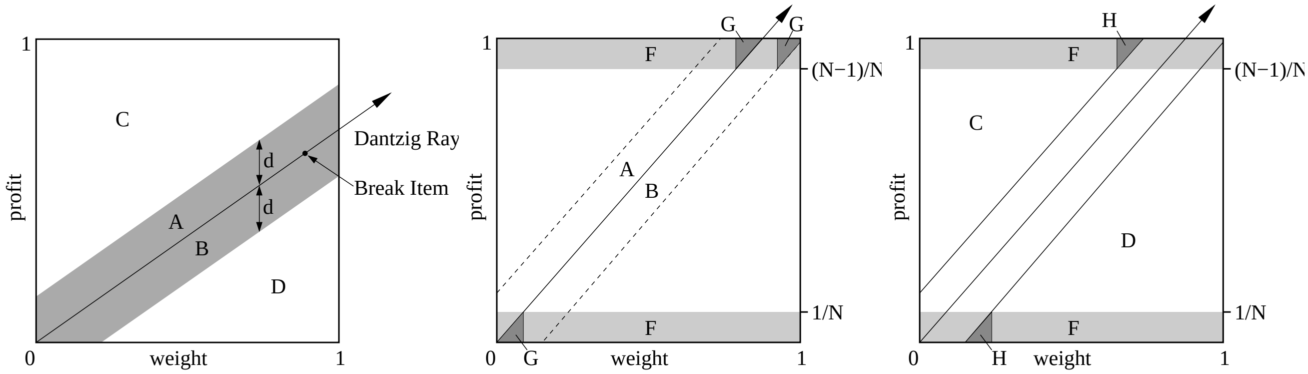

Figure 2: The core stripe consists of regions A and B which are defined by the Dantzig Ray. Items in region G have insufficient variation in profit.

For a moment, let us assume that r and the number of items in A and B as well as their individual weights are fixed arbitrarily. Consider an item i with weight w_i in region A. The profit of this item corresponds to a point on the line segment $L_i = \{p \in [0,1] | (p, w_i) \in A\}$. Observe that the Dantzig ray depends on the weight w_i of this item but not on its profit. In particular, moving the point corresponding to i arbitrarily on the line segment L_i does not affect the position of the Dantzig ray. Hence, p_i is a random number chosen uniformly from an interval whose length corresponds to the length of the line segment L_i. The same holds for the points in B. Observe that for items in $(A \cup B) \setminus G$ these profit intervals have length at least $1/N$. As a consequence, the density of the profit distributions for these items is upper bounded by N. Hence, applying Lemma 3 yields the following claim. For a any region R, let X_R denote the number of items in R and q_R the number of dominating sets over these items.

Claim 5. *For any* $k \in \mathbb{Z}_{\geq 0}$, $\mathbf{E}\left[q_{(A\cup B)\setminus G} \mid X_{(A\cup B)\setminus G} = k\right] \leq cNk^4 + 1$.

Next we come to the items in G. Adding a single item can increase the number of dominating sets at most by a factor two. This yields $q_{A\cup B} \leq 2^{X_G} \cdot q_{(A\cup B)\setminus G}$. Observe that $\mathbf{E}[q_{A\cup B}]$ is the dominant term in the estimation of the expected running time of the Nemhauser/Ullmann algorithm given by Lemma 2. In the following, we will (implicitly) show that $\mathbf{E}\left[q_{(A\cup B)\setminus G}\right] = N \operatorname{polylog} N$ and $\mathbf{E}\left[2^{X_G}\right] = O(1)$. Unfortunately, however, the random variables X_G and $q_{(A\cup B)\setminus G}$ are not completely independent as both of them depend on the position of the Dantzig ray. The major difficulty in the remaining analysis is to formally prove that this kind of dependence is insignificant.

Suppose the algorithm first computes the dominating sets over items in $(A\cup B)\setminus G$ adding items in some order independent of the profits, e.g., in order of increasing weight. Afterwards, the items from G are added. Let T denote the running time of this algorithm. Lemma 2 combined with Claim 5 yields

$$\mathbf{E}[T \mid X_{A\cup B} = k \wedge X_G = g] \;\leq\; c[(N(k-g)^4+1)(k-g) + ((N(k-g)^4+1)2^g)] \;,$$

for every $k \geq g \geq 0$. Define $f(k,g) = cN(k-g+7)^5 2^g$. This way,

$$\mathbf{E}[T \mid X_{A\cup B} = k \wedge X_G = g] \;\leq\; f(k,g)$$

with $f(k,g)$ being a function that is monotonically increasing in g, for every $0 \leq g \leq k$. Because $G \subseteq A\cup B$, we obtain

$$\mathbf{E}[T] \;\leq\; \sum_{k=0}^{\infty}\sum_{g=0}^{k} \mathbf{Pr}[X_{A\cup B} = k \wedge X_G = g]\, f(k,g).$$

Combining $G \subset (A\cup B)\cap F$ with the monotonicity property of the function f gives

$$\mathbf{E}[T] \;\leq\; \sum_{k=0}^{\infty}\sum_{g=0}^{k} \mathbf{Pr}\left[X_{A\cup B} = k \wedge X_{(A\cup B)\cap F} = g\right] f(k,g).$$

Now substituting $f(k,g)$ and rearranging the sums yields

$$\begin{aligned}
\mathbf{E}[T] &\le \sum_{g=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\cap F}=g\right] \sum_{k=g}^{\infty} \mathbf{Pr}\left[X_{A\cup B}=k \mid X_{(A\cup B)\cap F}=g\right] cN(k-g+7)^5 2^g \\
&= cN\sum_{g=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\cap F}=g\right] 2^g \sum_{k=g}^{\infty} \mathbf{Pr}\left[X_{A\cup B}=k \mid X_{(A\cup B)\cap F}=g\right] (k-g+7)^5 \\
&= cN\sum_{g=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\cap F}=g\right] 2^g \sum_{k=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\setminus F}=k \mid X_{(A\cup B)\cap F}=g\right] (k+7)^5 \ .
\end{aligned}$$

Next we show that the variable $X_{(A\cup B)\setminus F}$ under the condition $X_{(A\cup B)\cap F}=g$ is sharply concentrated. Observe that these two variables describe numbers of points in disjoint regions of the unit square and, hence, depend on each other only due to the position of the Danzig ray r. (Recall that we assume a Poisson distribution.) Unfortunately, these regions are adjacent to r, and r has some tendency to fall into a dense region. At this point, however, we can deal with this kind of dependencies by enumerating over all basic positions of the ray r. In this way, we obtain a tail bound on $X_{(A\cup B)\setminus F}$ that implies the following lemma. The proof of this lemma including the concentration result for $X_{(A\cup B)\setminus F}$ is shifted to the appendix.

Lemma 6. *There exist constants $c_3, c_4 > 0$ such that $\forall g \ge 0$: $\sum_{k=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\setminus F}=k \mid X_{(A\cup B)\cap F}=g\right](k+7)^5 \le c_3 \log^{c_4} N$.*

From this lemma, we immediately obtain

$$\mathbf{E}[T] \ \le\ cNc_3\log^{c_4} N\cdot \sum_{g=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\cap F}=g\right]2^g \ \le\ cc_3N\log^{c_4} N\cdot \mathbf{E}\left[2^{X_{(A\cup B)\cap F}}\right] \ . \tag{2}$$

We further simplify.

Lemma 7. $\mathbf{E}\left[2^{X_{(A\cup B)\cap F}}\right] \le \mathbf{E}\left[2^{X_F}\right] = e^2$.

Proof. The number of points in F is a Poisson random variable whose mean corresponds to the area covered by F times N because the number of points in the unit square is a Poisson variable with mean N and points are placed uniformly at random. Obviously, the area of F is $2/N$. Hence, the number of points in F follows the Poisson distribution with parameter 2. Consequently,

$$\mathbf{E}\left[2^{X_F}\right] \ =\ \sum_{f\ge 0} \mathbf{Pr}[X_f=f]\,2^f \ =\ \sum_{f\ge 0}\frac{e^{-2}\cdot 2^f}{f!}\cdot 2^f \ =\ e^{-2}\sum_{f\ge 0}\frac{4^f}{f!} \ =\ e^{-2}e^4 \ =\ e^2 \ .$$

□

Finally, applying Lemma 7 on Equation 2 yields $\mathbf{E}[T] \le cc_3e^2N\log^{c_4} N$. Thus Theorem 4 is shown. □

4 Algorithm FastCore 2: Computing Two Lists of Dominating Sets

In order to obtain an algorithm that always computes an optimal solution, one can dynamically expand the core until algorithm FastCore 1 is successful. A somewhat extreme variant of this approach is to start with a core stripe that immediately gives a high success probability, say $1-n^{-3}$, and to use a single backup routine that computes all dominating sets using the Nemhauser/Ullmann algorithm if the core algorithm fails. On a first view this approach looks quite promising as an analysis in [2] shows that the expected running time for computing all dominating sets under the uniform distribution is only $O(n^4)$. Therefore, let us have a closer look at this approach. The running time is mainly determined by the number of computed states (i.e. dominating sets). Let q_{all}, q_{in}, and q_{out} denote the number of dominating sets over all items, over items inside and over items outside the core stripe, respectively. Let $\mathcal{E}$ denote the event that the core computation is successful. Then the expected number of dominating sets is $\mathbf{E}[q_{\text{in}}] + \mathbf{E}[q_{\text{all}}|\neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}]$. The difficulty lies in estimating $\mathbf{E}[q_{\text{all}}|\neg\mathcal{E}]$. Observe that the event $\neg\mathcal{E}$, by definition, is very unlikely. Thus the value of q_{all} might be extremely biased by $\neg\mathcal{E}$.

In order to bypass the difficulties caused by dependencies, we use a different approach utilizing two lists of dominating sets. First, we compute a list with all dominating sets over the items in the core. If this fails to find an optimal solution, we compute a second list with all dominating sets over the items outside the core. Observe that combining these lists to obtain all dominating sets yields a combined list of maximum length $q_{\text{in}} \cdot q_{\text{out}}$. This way the expected number of dominating sets is upper-bounded by $\mathbf{E}[q_{\text{in}}|\mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[q_{\text{in}} \cdot q_{\text{out}}|\neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}]$. In the following analysis, we will be able to give a good estimate for $\mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$ as the random variable q_{out} is only slightly biased by $\neg\mathcal{E}$. Still this does not immediately solve the problem since q_{in} heavily depends on $\neg\mathcal{E}$ so that we are not able to upper-bound $\mathbf{E}[q_{\text{in}}|\neg\mathcal{E}]$ appropriately.

Now the key observation is that we do not need to compute all dominating sets but we only need to find the set among them that is maximal with respect to the given weight threshold B. This, however, given the two sorted lists of dominating sets can be done in time $O(q_{\text{in}} + q_{\text{out}})$ instead of $O(q_{\text{in}} \cdot q_{\text{out}})$ by scanning the two sorted lists in parallel. This scanning technique is presented, e.g., in [12], where it is used to reduce the worst case running time of the Nemhauser/Ullmann algorithm from $O(2^n)$ to $O(2^{n/2})$. In our case this trick does not only reduce the running time but also allows to deal with the dependencies. Using this technique, the expected running time of algorithm FastCore 2 can be (very roughly) estimated by $\mathbf{E}[q_{\text{in}}] + \mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$. We have estimated $\mathbf{E}[q_{\text{in}}] = N\text{polylog}\,N$ already in the proof of Theorem 4. The following analysis mainly deals with bounding $\mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$.

Theorem 8. *Algorithm FastCore 2 always computes an optimal solution. Its expected running time is $O(N\,polylog\,N)$.*

Proof. We begin the proof by specifying some details of the algorithm that are missing in the description above. The algorithm uses a fixed core stripe of height $d = 5c_0(\log N)^3/N$. Let us adopt the notation from the previous section for the regions defined by the core stripe as depicted in Figure 2. Let C and D denote the regions above and below the core stripe, respectively. We add the region H (see Figure 2c) to the core stripe. The analysis of the running time of algorithm FastCore 1 for the items inside the core is not affected by this change as we already upper-bounded X_G by X_F and $F \supset G \cup H$. This technical assumption will significantly simplify the presentation of the analysis of the Nemhauser/Ullmann algorithm applied to the elements outside the core as it ensures that all items in $C \cup D$ follow a distribution with density at most N.

Let T, $T_{A\cup B\cup H}$, and $T_{C\cup D}$ denote the running time of algorithm FastCore 2, algorithm FastCore 1 applied to the items in $A \cup B \cup H$, and the Nemhauser/Ullmann algorithm applied to the elements items in $C \cup D$, respectively. Furthermore, let $\mathcal{E}$ be the event that the height $2d$ of the core was sufficiently large to prove optimality, i.e. the integrality gap is at most d. We account the time for combining the two lists to the time needed for their computation. Then

$$\mathbf{E}[T] = \mathbf{E}[T_{A\cup B\cup H}] + \mathbf{Pr}[\neg\mathcal{E}] \cdot \mathbf{E}[T_{C\cup D} \mid \neg\mathcal{E}]$$

The analysis of algorithm FastCore 1 yields $\mathbf{E}[T_{A\cup B\cup H}] = O(N\text{polylog}\,N)$. In the following, we will show

$$\mathbf{E}[T_{C\cup D} \mid \neg\mathcal{E}] \;=\; O\left(\frac{N\text{polylog}\,N}{\mathbf{Pr}[\neg\mathcal{E}]}\right) \;,$$

which yields the theorem.

First, let us verify that every item in $C \cup D$ follows a distribution with density at most N. For a moment assume that the Dantzig ray is fixed. Suppose an adversary specifies the weight w_i of an item i in region C (or analogously for D). Then the item can be moved up and down on the line segment $L_i = \{p \in [0,1] : (p, w_i) \in C\}$ without affecting the event $\neg\mathcal{E}$. The length of this line segment is at least $\frac{1}{N}$ as we added the region H to the core. Consequently, independent of the outcome of $\neg\mathcal{E}$, the profit of each element follows a uniform distribution with density at most N. Let k denote a random variable describing the number of elements in the region $C \cup D$. The running time of the Nemhauser/Ullman algorithm applied to the elements in $C \cup D$ is

$$\mathbf{E}\left[T_{C\cup D} \mid \neg\mathcal{E}\right] = O(k^5 N)$$

Thus it remains to show

$$\mathbf{E}\left[k^5 \mid \neg\mathcal{E}\right] = O\left(\frac{\operatorname{polylog} N}{\mathbf{Pr}\left[\neg\mathcal{E}\right]}\right) \quad .$$

Observe that Lemma 1 with $d = 5c_0(\log N)^3/N$ yields $\mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1} \geq 2^{5\log N} = N^5$.

Now the idea is that the random variable k is very sharply concentrated around its mean, which is slightly smaller than N, so that conditioning on $\neg\mathcal{E}$ does not significantly change the mean value of k. For every $\kappa \geq 1$,

$$\mathbf{E}\left[k^5 \mid \neg\mathcal{E}\right] \;=\; \sum_{i=1}^{\infty} \mathbf{Pr}\left[k^5 \geq i \mid \neg\mathcal{E}\right] \;\leq\; \kappa + \sum_{i=\kappa}^{\infty} \mathbf{Pr}\left[k^5 \geq i \mid \neg\mathcal{E}\right] \;\leq\; \kappa + \sum_{i=\kappa}^{\infty} \mathbf{Pr}\left[k^5 \geq i\right] \cdot \mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1} \quad .$$

As k is Poisson distributed, $\mathbf{Pr}\left[k^5 \geq (2\alpha N)^5\right] \leq \exp(-\frac{1}{2}\alpha N)$, for every $\alpha \geq 1$. Thus, for $i \geq (2N)^5$, $\mathbf{Pr}\left[k^5 \geq i\right] \leq \exp\left(-\frac{1}{4} i^{1/5}\right)$. Now set $\kappa = (2N)^5$. Then

$$\mathbf{E}\left[k^5 \mid \neg\mathcal{E}\right] \;\leq\; (2N)^5 + \sum_{i=\kappa}^{\infty} \exp\left(-\frac{1}{4} i^{1/5}\right) \cdot \mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1} \;=\; O(N^5 + \mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1}) \quad .$$

Finally, observe that the latter term is bounded from above by $O\left(\frac{\operatorname{polylog} N}{\mathbf{Pr}\left[\neg\mathcal{E}\right]}\right)$, for $\mathbf{Pr}\left[\neg\mathcal{E}\right] \leq N^{-5}$. This completes the proof of Theorem 8. □

5 δ-Correlated Random Instances

Our analysis shows that instances under the uniform model are quite easy to solve on average. However, these instance are not typical in real world applications. Therefore, we follow [9],[10],[16] and [17] and investigate the so-called weakly correlated instances. In particular, we define δ-correlated instances as follows. Weights are randomly drawn from $[0,1]$. Profits are set to a random perturbation of the corresponding weight value, i.e., for all $i \in [n]$, $p_i := w_i + r_i$, where r_i is a random variable uniformly distributed in $[-\delta/2, \delta/2]$. This can be seen as choosing n points independently at random from the quadrangle defined by the points $(0,\delta/2),(0,-\delta/2),(1,1-\delta/2),(1,1+\delta/2)$ (see Figure 3). For our analysis we again assume that the value of n is chosen according to the Poisson distribution with parameter N, and $B = \beta N$, for some constant $\beta \in (0, \frac{1}{2})$.

5.1 Integrality gap for δ-Correlated Instances

In this section we prove the following upper bound on the integrality gap Γ for δ-correlated instances.

Lemma 9. *There is constant k_0 such that for every $\alpha \leq \log^4 N$,* $\mathbf{Pr}\left[\Gamma \geq k_0 \alpha \frac{\delta}{N} \log^2 \frac{N}{\delta}\right] \leq 2^{-\alpha}$.

Proof. (sketch). We adapt the proof from Lueker [13] for uniform instances. Please refer to [13] for details. In particular we compare the solution of an approximation algorithm with the optimal fractional solution. The difference of the objective values is an upper bound on the integrality gap. The approximation works as follows. Let $k := \log_4(N/\delta)$ and $\varepsilon = O(k4^{-k})$ nd assume that items are ordered according to non-decreasing profit-to-weight ratio. Starting with the empty knapsack we insert items in this order until the remaining knapsack capacity is smaller than vk where v is the expected weight of the next item (the average weight of points in a region swept by the density ray when slightly rotating it further). Let *cap* be the remaining capacity of the knapsack. We then repeatedly consider successive sets of about $2k$ items trying to find a subset of k items with weight in $[cap - 2\varepsilon, cap]$. The probability that we find such a set is at least $\frac{1}{2}$ for every set of $2k$ items. This can be shown with the help of the following lemma, which essentially shows that subsets of random numbers lay exponentially dense.

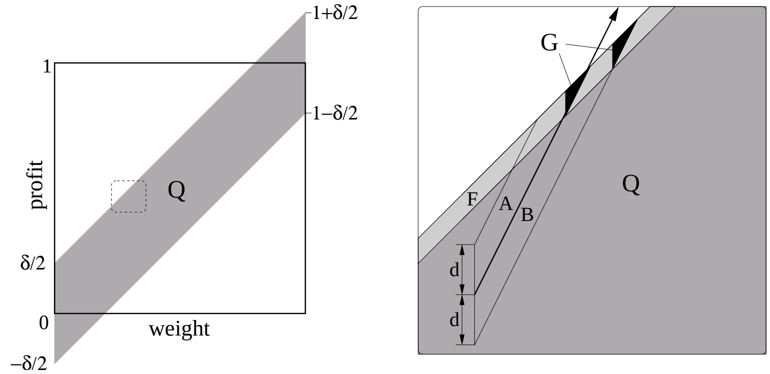

Figure 3: For δ-correlated instances items are uniformly sampled from region Q. The right figure shows an example for core stripes A and B with height d. Items in region $G \subseteq F$ have insufficient randomness.

Lemma 10. *(Lueker [13]) Let f be a piecewise continuous density function with domain $[-a,a]$. Suppose f is bounded and has mean 0 and variance 1. Let x_k be a real sequence with $x_k = o(\sqrt{k})$. Suppose we draw $2k$ variables $X_1,\ldots,X_{2k}$ according to f. Then for large enough k the probability that there exists some k-element subset $S \in [2k]$ with $\sum_{i\in S} X_i \in [x_k - \varepsilon, x_k + \varepsilon]$ is at least $\frac{1}{2}$, provided $\varepsilon = 7k4^{-k}$.*

The gap in profit of the approximate and optimal fractional solution can be attributed to two causes (see Equation 1). First, we might not fully use the capacity of the knapsack and secondly, we use items with small profit-to-weight ratio, i.e., items i with $p_i/w_i < p_b/w_b = r$. Applying Lemma 10, the residual capacity is at most 2ε. Consequently, the loss in profit is at most $r2\varepsilon = O(\frac{r\delta}{N} \cdot \log \frac{N}{\delta})$, where $r = p_b/w_b$ is the slope of the Dantzig ray, which is upper bounded by a constant with high probability. The second part can be estimated by $cap(r - r_k) \leq k(r_0 - r_k)$ where r_0 and r_k are the slope of the density ray before the first and after the last iteration, respectively. In each iteration the density ray advances $O(\delta k/N)$ with high probability. If the algorithm performs k iterations, then the accumulated loss of the k items is $O(\delta k^2/N)$. □

5.2 Running Time

Using algorithm FastCore 2 with core height $d = c_o \frac{\delta}{N} \log^3 \frac{N}{\delta}$ we obtain the following bound on the running time.

Theorem 11. *The expected running time of FastCore 2 on δ-correlated instances is $O(\frac{N}{\delta}\, polylog\, \frac{N}{\delta})$.*

Proof. (sketch). Let Q be the region from where we sample the items (see Figure 3). The area of Q has size δ. Compared to the uniform model, the concentration of items is larger by factor $1/\delta$. Choosing core height $d = c_o \frac{\delta}{N} \log^3 \frac{N}{\delta}$ we expect about $2c_0 \log^3 \frac{N}{\delta}$ core items in contrast to $2c_0 \log^3 N$ for the uniform case. Let A and B denote the core regions above and below the Dantzig ray, respectively. Consider the case when the slope of the Dantzig ray is larger than 1 (the other case is similar). Define regions F and G with $G \subset F$ (see Figure 3).

$$
\begin{aligned}
F &= \{(x,y) \in Q : y - x \in [d - \delta/N, d]\} \\
G &= \{(w,p) \in A \,|\, (w, rw) \in F\} \cup \{(w,p) \in B \,|\, (w, rw - d) \in F\} \ .
\end{aligned}
$$

For items in $(A \cup B) \setminus G$ the maximum density of the profit distribution is N/δ. Therefore, for any $k \in \mathbb{Z}_{\geq 0}$, $\mathbf{E}\left[q_{(A\cup B)\setminus G} \mid X_{(A\cup B)\setminus G} = k\right] \leq c_0(N/\delta)k^4 + 1$. Items in G have larger densities, so we again pessimistically assume that each item doubles the number of dominating sets. We have chosen the size of F so that on average there is only 1 item in F. This way our analysis which accounts also for items in G applies here as well.

To summarize, the parameter δ has an logarithmic influence on the expected number of core items. Furthermore, the smaller δ the smaller the deviation of the profits from weights and the larger the maximum density of the profit distribution, resulting in a larger number of dominating sets over the core items. Consequently, the running time increases in δ^{-1} essentially like $\delta^{-1} \log \delta^{-1}$. □

6 Acknowledgments

We want to thank David Pisinger for providing us with the code of the combo knapsack implementation.

References

[1] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research,* Vol. 28, pp. 1130-1154, 1980.

[2] R. Beier and B. Voecking. Random Knapsack in Expected Polynomial Time. *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, 2003.

[3] G. B. Dantzig. Discrete Variable Extremum Problems. *Operations Research*, Vol 5, pp. 266-277, 1957.

[4] G. D'Atri and C. Puech. Probabilistic Analysis of the Subset-Sum Problem. *Discrete Applied Math.*, Vol 4, 329–334, 1982.

[5] M. E. Dyer and A. M. Frieze. Probabilistic Analysis of the Multidimensional Knapsack Problem. *Mathematics of Operations Research 14(1)*, 162-176, 1989.

[6] D. Fayard and G. Plateau. An algorithm for the solution of the 0-1 knapsack problem. *Computing*, Vol. 28, pp. 269-287, 1982.

[7] A. M. Frieze. On the Lagarias-Odlyzko algorithm for the Subset-Sum Problem. *SIAM J. Comput. 15(2)*, 536-539, 1986.

[8] M. R. Garey and D. S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, 1979.

[9] S. Martello, D. Pisinger and P. Toth. New Trends in Exact Algorithms for the 0/1 Knapsack Problem. *European Journal of Operational Research*, Vol. 123, pp. 325-332, 2000.

[10] S. Martello and P. Toth. *Knapsack Problems – Algorithms and Computer Implementations*. Wiley, 1990.

[11] A. Goldberg and A. Marchetti-Spaccamela. On Finding the Exact Solution to a Zero-One Knapsack Problem. *Proceedings of the 16th ACM Symposium on Theory of Computing (STOC)*, 359–368, 1984.

[12] E. Horowitz and S. Sahni. Computing Partitions with Applications to the Knapsack Problem, *J. of the ACM*, Vol. 21, 277–292, 1974.

[13] G. S. Lueker. On the Average Difference Between the Solutions to Linear and Integer Knapsack Problems. *Applied Probability - Computer Science, the Interface*, 1, Birkhauser 1982.

[14] G. S. Lueker. Exponentially Small Bounds on the Expected Optimum of the Partition and Subset Sum Problems. *Random Structures and Algorithms*, 12, 51-62, 1998.

[15] G. Nemhauser and Z. Ullmann. Discrete Dynamic Programming and Capital Allocation. *Management Science*, 15(9), 494–505, 1969.

[16] D. Pisinger. *Algorithms for Knapsack Problems.* Ph.D. thesis, DIKU, University of Copenhagen, 1995.

[17] D. Pisinger and P. Toth. *Knapsack Problems.* In D-Z. Du, P. Pardalos (ed.), Handbook of Combinatorial Optimization, vol. 1, Kluwer Academic Publishers, 299-428, 1998.

#states (in Thousand) for uniform instances				
N	Dom	DomF LossF	DomF LossF 2-lists	combo
2000	71	22	1	5
4000	175	48	2	11
8000	428	109	4	22
16000	1,045	230	6	45
32000	2,587	547	12	96
64000	5,887	1,110	20	173
128000	14,020	2,486	34	329
256000	32,327	5,235	54	639
512000	76,700	11,442	87	1,337
1024000	175,451	24,868	148	2,760

Running time for δ-correlated instances				
$1/\delta$	Dom	DomF LossF	DomF LossF 2-lists	combo
2	0.34	0.07	–	0.02
4	1.01	0.16	–	0.04
8	2.88	0.32	0.01	0.07
16	7.47	0.77	0.01	0.13
32	17.83	2.54	0.02	0.27
64	41.28	6.83	0.02	0.54
128	109.31	16.92	0.04	1.23
256	–	36.13	0.06	3.05
512	–	86.51	0.11	8.42
1024	–	202.01	0.18	21.47

Table 1: **a**) Number of states (in Thousand) for uniform instances (average over 1000 instances). DomF = dominating-set filter; LossF = Loss-filter; combo = knapsack solver by Pisinger. **b**) Running time for δ-correlated instances without preprocessing (average over 1000 instances, $N = 10000$). We fixed $\beta = 0.4$ for all experiments.

A Experimental Results

The ideas and concepts presented in this paper allow not only to derive theoretical results but also convince in experimental studies. We implemented a core algorithm that combines the ideas presented in this and other theoretical studies. Our implementation uses the core concept from Goldberg and Spaccamela considering only items in a stripe around the Dantzig ray. The core size increases dynamically. We combine their loss filter with the dominance filter based on the Nemhauser/Ullmann algorithm. In addition, we use two lists, which are finally combined by the linear time algorithm of Horowitz and Sahni. All these ingredients are not exactly implemented as in our theoretical study but the ideas are the same. The major difference is that, in our implementation, elements are added to the core in order of increasing vertical distance to the Dantzig ray while our analysis assumes some order independent of their profit.

It turns out that our implementation beats the best previous implementation by Pisinger (combo code with 64-bits integer arithmetic [9]) by several orders of magnitudes. Table 1 presents the number of computed subsets (states) for uniformly random instances. As predicted the dominance filter produces a slightly super-linear number of states. Putting all the ideas together (dominance filter + loss filter + 2 lists) the number states appears to be even sub-linear in the number of items. In fact for uniform instances the running time is dominated by the pre-computation time, i.e., the time for computing the optimal fractional solution. Studying the running time becomes interesting for δ-correlated instances. Table 1b shows that the running time for δ-correlated instances increases slightly super-linear in $1/\delta$ (except for the variant with two lists, which shows a sub-linear behavior). Also this matches our theoretical results.

B Proof of Lemma 6

We prove a tail bound on $X_{(A\cup B)\setminus F}$ conditioned on $X_{(A\cup B)\cap F} = g$ for arbitrary g, from which we can derive the lemma. These variables depend on each other only due to the position of the Danzig ray r. Unfortunately, this ray has some tendency to fall into a dense region. The following claim, however, shows that these dependencies are negligible.

Claim 12. *Let $\mu = 2dN$ which is roughly the expected number of core items. Suppose an adversary sees all points in the unit square and then this adversary chooses the position of a ray r through the origin and this way determines the regions $(A\cup B)\setminus F$. Then for all $\alpha \geq 1$, $\mathbf{Pr}[X_{A\cup B} \geq \alpha 2\mu] \leq N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$.*

Proof. We define a set of overlapping parallelograms R_i having the property that any given core stripe $A\cup B$ is completely covered by at most two parallelograms. The first $l = \lceil 1/(2d) \rceil$ parallelograms cover the right lower triangle of the unit square U. Parallelogram R_i $(i = 1,\dots,l)$ is a quadrangle with corner coordinates $(0,d),(0,-d),(1,i2d-2d)$ and $(1,i2d)$. Another l parallelograms cover the upper left triangle of the unit square U. Parallelogram R_{l+i} $(i = 1,\dots,l)$ is a quadrangle with corner coordinates $(-d,0),(d,0),(i2d,1)$ and $(i2d-2d,1)$. Every parallelogram covers an area of size $2d$. It is easy to verify that this set of regions has the required property. Let X_i denote the number of items in region R_i. Then $\mathbf{E}[X_i] = \text{area}(\mathcal{U}\cap R_i)\cdot N < 2dN = \mu$. Using Chernoff bounds we have for all $\alpha \geq 1$ and $i \in [2l]$:

$$\mathbf{Pr}[X_i \geq \alpha\mu] \leq \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu.$$

Thus $\mathbf{Pr}[\exists i : X_i \geq \alpha\mu] \leq \sum_{i=1}^{2l} \mathbf{Pr}[X_i \geq \alpha\mu] = 2l\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$ and $\mathbf{Pr}\left[X_{(A\cup B)\setminus F} \geq 2\alpha\mu\right] \leq 2l\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu \leq N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$. $\square$

Now let $\mathcal{E}$ be the event $X_{(A\cup B)\cap F} = g$. Then

$$\begin{aligned}&\sum_{k=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\setminus F} = k \mid \mathcal{E}\right](k+7)^5 \\ &\quad\leq \mathbf{Pr}\left[X_{(A\cup B)\setminus F} \leq 4\mu \mid \mathcal{E}\right](4\mu+7)^5 + \sum_{\alpha=2}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\setminus F} \geq 2\alpha\mu \mid \mathcal{E}\right](2(\alpha+1)\mu+7)^5.\end{aligned}$$

To estimate $\mathbf{Pr}\left[X_{(A\cup B)\setminus F} \geq \alpha 2\mu \mid \mathcal{E}\right]$ we use Claim 12. Observe that this claim is not affected by the condition $\mathcal{E}$ as the only dependence between $\mathcal{E}$ and $X_{(A\cup B)\setminus F}$ is via the position of the Dantzig ray and the claim assumes an adversarial choice for this ray.

Therefore $\mathbf{Pr}\left[X_{(A\cup B)\setminus F} \geq 2\alpha\mu \mid \mathcal{E}\right] \leq N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$. Using the equality $\mu = 2dN = 2c_0\cdot\log^3 N$ yields

$$\sum_{\alpha=2}^{\infty} N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu \cdot (2(\alpha+1)\mu+7)^5 \leq \mu^5 \sum_{\alpha=2}^{\infty} N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^{2c_0\cdot\log^3 N} \cdot (2\alpha+2+7/\mu)^5 = O(\mu^5).$$

Finally, $\mathbf{Pr}\left[X_{(A\cup B)\setminus F} \leq 4\mu \mid \mathcal{E}\right](4\mu+7)^5 = O(\mu^5) = O((\log N)^{15})$, which completes the proof of Lemma 6.