Probabilistic Analysis of Knapsack Core Algorithms

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

Berthold Vöcking†
voecking@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 *core algorithms*, the predominant algorithmic concept used in practice. These algorithms start with the computation of an optimal fractional solution that has only one fractional item and then they 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 a 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 practical experiments. We prove an upper bound of $O(n\,\mathrm{polylog}(n))$ on the expected running time of a core algorithm on instances with n items whose profits and weights are drawn independently, uniformly at random. A previous analysis on the average-case complexity of the knapsack problem proves a running time of $O(n^4)$, but for a different kind of algorithms. The previously best known upper bound on the running time of core algorithms is polynomial as well. The degree of this polynomial, however, is at least a large three digit number. In addition to uniformly random instances, we investigate harder instances in which profits and weights are pairwise correlated. For this kind of instances, we can prove a tradeoff describing how the degree of correlation influences the running time.

1 Introduction

This paper is concerned with a probabilistic analysis of the 0/1 knapsack problem. A subset of n given items has to be packed in a knapsack of *capacity* b. Each item has a *profit* p_i and a *weight* w_i, for $i \in [n] = \{1,2,\ldots,n\}$. The problem is to select a subset of the items whose total weight does not exceed the capacity bound b and whose total profit is as large as possible. In terms of an integer linear program (ILP), the problem is

$$\begin{aligned} \text{maximize} \quad & \sum_{i\in[n]} p_i x_i \\ \text{subject to} \quad & \sum_{i\in[n]} w_i x_i \le b \\ \text{and} \quad & x_i \in \{0,1\}, \text{ for } i \in [n]\,. \end{aligned}$$

We assume that weights and profits are drawn independently, uniformly at random from $[0,1]$. Following the conventions in previous analyses [11, 9], the value of n is assumed to be chosen according to the Poisson distribution with parameter N. Furthermore, $b = \beta N$, for some constant $\beta \in (0, \frac{1}{2})$.

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 [7, 8, 14]. This algorithmic concept was suggested by Balas and Zemel [1]. The idea is to start with the computation of an optimal fractional solution with at most one fractional item 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 for "typical instances" is relatively small. 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 [11]. Based on this result, Goldberg and Marchetti-Spaccamela [9] 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 large 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

*Supported by the graduated studies program on "Quality Guarantees for Computer Systems" funded by the German Science Foundation (DFG).
†Supported in part by DFG grant Vo889/1-1.

time. This work was later extended to the multidimensional knapsack problem by Dyer and Frieze [4].

Better bounds on the running time are only known for an algorithm by Nemhauser and Ullmann [13]. 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 $w(S) \leq w(T)$ and $p(S) \geq p(T)$. For simplicity, let us assume that all sets have different profits. The considered random instances satisfy this assumption with probability 1. Under this assumption, 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 other sets, the so-called *dominating sets*. In a recent study [2], we showed that, for uniformly random instances, the expected number of dominating sets is bounded by $O(n^3)$, even if the weights are chosen by an adversary. 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 [7, 8, 14]. Their running time on uniformly random instances is almost linear. Interestingly, the most successful implementations of core algorithms [7, 14] additionally exploit domination in order to decrease the number of subsets generated by the core.

1.1 New results

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 ([7],[15]), 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 [9] and combine it with the enumeration method for dominating sets by Nemhauser and Ullmann [13]. Somewhat surprisingly, this combination of theoretical concepts 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.

1.2 Outline

In Section 2 we start with a short overview of the techniques and results from previous work 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 sub-optimal 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 preliminary experimental results.

2 Tools and techniques

2.1 Core algorithms

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 $(w_i, p_i) \in \mathbb{R}^2$. 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 an appropriate subset of candidate items, called the core. Assume the core contains all items that needs to be exchanged to obtain an optimal solution, then items outside the core can be fixed to the value given by the break solution. There are various different ways to define the core.

[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.

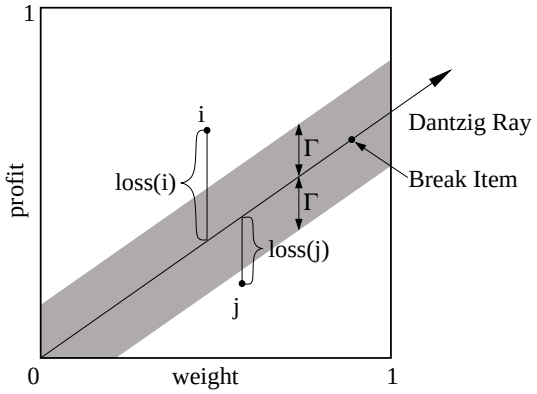

Figure 1: Dantzig ray and Break Item. The core stripe has vertical width 2Γ with Γ denoting the integrality gap. The loss of an item is the vertical distance to the Dantzig ray.

Goldberg and Marchetti-Spaccamela [9] define the core as follows. Assume the items are given in order of non-increasing profit-to-weight ratio and let κ denote the index of the break item. The solution vector for the fractional problem has the form $\bar{X} = (1,\ldots,1,f,0,\ldots,0)$ where $f \in (0,1]$ is the entry at position κ. For any feasible integer solution $X = (x_1,\ldots,x_n)$, define $ch(X) = \{i \in [n] : \bar{x} \neq x_i\}$, i.e., $ch(X)$ is the set of items on which $\bar{X}$ and X do not agree. When removing an item from the break solution the freed capacity can be used to include items that have profit-to-weight ratio at most $r := p_\kappa/w_\kappa$ 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 $r := p_\kappa/w_\kappa$ 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 $\bar{X}$ can be expressed as

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

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 $\Gamma = P(\bar{X}) - P(X^*)$ 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 item 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 [11] 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 [9] observe that this analysis can easily be generalized to obtain exponential tail bounds on Γ.

LEMMA 2.1. *There is constant c_0 such that for every $\alpha \leq \log^4 N$, $\mathbf{Pr}\left[\Gamma \geq c_0\alpha\frac{\log^2 N}{N}\right] \leq 2^{-\alpha}$.*

In words, the integrality gap does not exceed $O(\frac{\log^2 N}{N})$, with high probability. Let us remark that the value of c_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 [9] 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 Dantzig ray. Basically, the expected value of X corresponds to N times the area covered by the Γ-region around the Dantzig ray, 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 2.1 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 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. Observe that the considered random instances satisfy this assumption with probability 1. 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 other sets, 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 [13] 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 2.2. *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.

LEMMA 2.3. *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 $c_1 \in \mathbb{R}_{>0}$ such that the expected number of dominating sets is $\mathbf{E}[q] \leq c_1\phi\mu n^4 + 1$.*

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. The bound on the expected number of dominating sets increases linearly with the maximum density. The same holds for the expected running time. Saying it the other way around, the less randomness is available, the larger the expected running time.

3 Algorithm FastCore 1: filtering dominated solutions

Our first algorithm has almost linear running time but fails with a small probability. We use a static core consisting of all items with loss at most $d = c_d N^{-1}\log^3 N$, for a suitable constant c_d. In Figure 2a, core items correspond to those items falling into the regions A or B. We use the Nemhauser/Ullmann algorithm to generate all dominating sets over the core items. The items in region C, i.e. items outside the core and above the ray, are virtually added to these sets. Among all dominating sets satisfying the capacity bound 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 the algorithm outputs this set, otherwise the algorithm outputs *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 the algorithm might have nevertheless found the optimal solution but only the proof of optimality is missing. From Lemma 2.1, it follows that the failure probability is $\mathbf{Pr}[\Gamma > d] \leq 2^{-c_d\log N/c_0} = N^{-c_d/c_0}$.

In the following probabilistic analysis, we will show that the expected running time of our algorithm is $O(N\operatorname{polylog}N)$. This analysis is quite tricky and complicated because of vast dependencies between different random variables. The central ideas, however, are rather simple and elegant. Thus, before going into the details, let us try to give some intuition about the main ideas behind the analysis. As the area covered by the core stripe is $O(N^{-1}\log^3 N)$, the

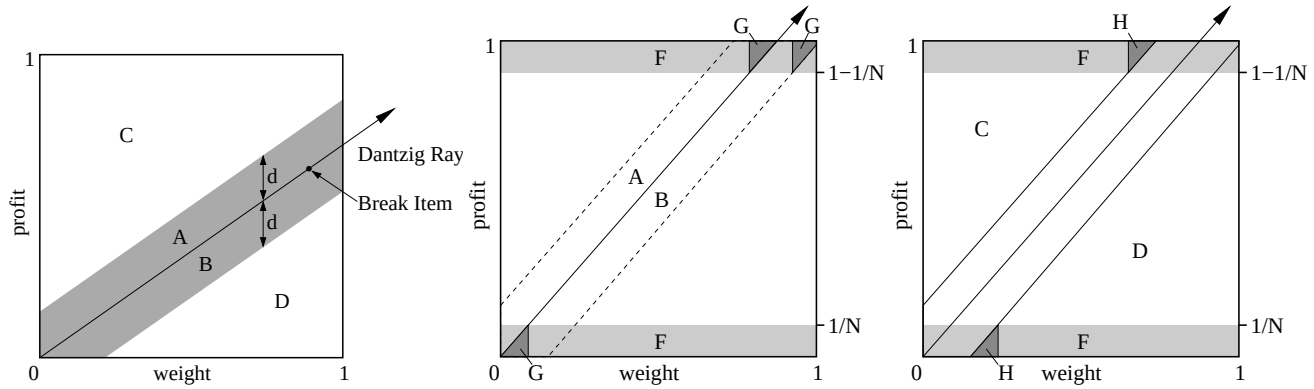

Figure 2: The core stripe consists of regions A and B which are defined by the Dantzig Ray. Items in region G have insufficient variation in profit.

number of core items is only $O(\log^3 N)$, with high probability. The running time of the Nemhauser/Ullmann 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 have to condition on the fact that these items have weights and profits that fall into the core stripe. The height of the core stripe is $\Theta(\frac{\log^3 N}{N})$. Thus, intuitively, the profit of a "typical" core item follows a uniform distribution over an interval of length $\Theta(\frac{\log^3 N}{N})$. More formally, we will show that the conditional probability distributions for the profits of almost all core items have density at most N. Hence, $\phi \leq N$. By Lemma 2.3, the expected number of dominating sets is at most $\phi \operatorname{polylog} N = N \operatorname{polylog} N$. Thus, by Lemma 2.2, the running time can be upper-bounded by $N \operatorname{polylog} N$ as well. Hence, interestingly, the linear term in the upper bound on the running time is not due the number of core items but due to the density of their profit distributions.

THEOREM 3.1. *For uniformly random instances, algorithm FastCore 1 computes an optimal solution with probability at least* $1 - N^{-c_d/c_0}$. *The expected running time of this algorithm is* $O(N \operatorname{polylog} N)$.

Proof. The bound on the failure probability follows from previous discussions. It 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 r with vertical distance at most $d = c_d N^{-1} \cdot \log^3 N$ from r, respectively. In the following, we identify the ray with its slope, i.e., $r = p_\kappa / w_\kappa$ with κ denoting the index of the break item. Define

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

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 depicts these regions. The motivation for these definitions will become clear soon.

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 item i arbitrarily on the line segment L_i does not affect the position of the Dantzig ray. Under these assumptions, the random variable p_i is chosen uniformly from the interval L_i. The same holds for the items in region B. Observe that the profit intervals for the items in $(A \cup B) \setminus G$ have length at least $1/N$, except for the break item that will be handled separately. As a consequence, the density of the profit distributions for these items is upper bounded by N. Hence, applying Lemma 2.3 yields the following bound on the expected number of dominating sets for these items. For any given region $R \subseteq \mathbb{R}^2$, let X_R denote the number of items in R and q_R the number of dominating sets over these items.

LEMMA 3.1. $\mathbf{E}\left[q_{(A \cup B) \setminus G} \,|\, X_{(A \cup B) \setminus G} = j\right] \leq c_1 N j^4 + 1$, *for any* $j \in \mathbb{N}$. □

Every additional item can increase the number of dominating sets at most by a factor of two. For this reason, we can assume that the break item is covered by the bound in Lemma 3.1 as well. Furthermore, can apply this fact to the items in G and obtain that the number of dominating sets over the core items is $q_{A \cup B} \leq 2^{X_G} \cdot q_{(A \cup B) \setminus G}$. The running time of the Nemhauser/Ullmann algorithm is roughly $\mathbf{E}[q_{A \cup B}]$ times the number of core items. 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.

We assume that the algorithm first computes the dominating sets over the items in region $(A\cup B)\setminus G$ adding the items in some order that is independent of the profits, e.g., in order of increasing weight. Afterwards, the items from region G are added. Let T denote the running time of this algorithm. Lemma 2.2 combined with Lemma 3.1 yields

$$\begin{aligned}&\mathbf{E}[T \,|\, X_{A\cup B} = k \wedge X_G = g]\\ &\quad\leq\ c_2[(N(k-g)^4+1)(k-g)+((N(k-g)^4+1)2^g)]\ ,\end{aligned}$$

for every $k \geq g \geq 0$ and a suitable constant c_2. Define $f(k,g) = c_2N(k-g+7)^5 2^g$. Notice that f is defined in a way such that it is monotonically increasing in g, for every $0 \leq g \leq k$. Rewriting the above bound in terms of this function, we obtain the following slightly weaker bound.

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

Applying first $G \subseteq A\cup B$ and then $G \subseteq (A\cup B)\cap F$ combined with the monotonicity property of f yields

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

Next replacing $f(k,g)$ by $c_2N(k-g+7)^5 2^g$ and rearranging the sums yields

$$\begin{aligned}\mathbf{E}[T] \ &\leq\ c_2N\sum_{g=0}^{\infty}\mathbf{Pr}\left[X_{(A\cup B)\cap F} = g\right]2^g\\ &\qquad\sum_{k=g}^{\infty}\mathbf{Pr}\left[X_{A\cup B} = k \,|\, X_{(A\cup B)\cap F} = g\right](k-g+7)^5\\ &=\ c_2N\sum_{g=0}^{\infty}\mathbf{Pr}\left[X_{(A\cup B)\cap F} = g\right]2^g\\ &\qquad\sum_{k=0}^{\infty}\mathbf{Pr}\left[X_{(A\cup B)\setminus F} = k \,|\, X_{(A\cup B)\cap F} = g\right](k+7)^5.\end{aligned}$$

Let us switch to a more compact notation and define $X = X_{(A\cup B)\setminus F}$ and $Y = X_{(A\cup B)\cap F}$. This way,

(3.1)
$$\mathbf{E}[T] \leq c_2N\sum_{g=0}^{\infty}\mathbf{Pr}[Y=g]\,2^g\sum_{k=0}^{\infty}\mathbf{Pr}[X=k|Y=g]\,(k+7)^5.$$

In the following, we upper-bound the two probability terms occuring in this bound one after the other. We start by proving $\sum_{k=0}^{\infty}\mathbf{Pr}[X=k\,|\,Y=g]\,(k+7)^5 = O(\text{polylog}N)$, for any choice of $g \geq 0$. Afterwards, we show that $\sum_{g=0}^{\infty}\mathbf{Pr}[Y=g]\,2^g = O(1)$. Hence, $\mathbf{E}[T] = O(\text{polylog}N)$ so that the theorem follows.

First, let us study the term $\mathbf{Pr}[X=k\,|\,Y=g]$. Observe, the variables X and Y describe numbers of points in the disjoint regions $(A\cup B)\setminus F$ and $(A\cup B)\cap F$, respectively. If these regions would be fixed, then these two variables would be independent as points are generated by the Poisson distribution. Unfortunately, however, both regions are defined by the Dantzig ray r and this ray somehow depends on the points found in these regions. In fact, both of the considered regions are adjacent to r, and r has some tendency to fall into a dense region. Therefore, we have to take into account the dependency of the variables X and Y on the random variable r. We deal with this dependency by placing worst-case assumptions on r. In particular, we assume an adversary knowing all points in the unit square chooses the ray r to be any ray through the origin instead of assuming that r is the ray through the break item. The regions $(A\cup B)\setminus F$ and $(A\cup B)\cap F$ are now defined with respect to this adversarial ray r. Let $\mu = 2dN$. Observe that the value of $\mathbf{E}[X]$ is roughly equal to μ.

LEMMA 3.2. *For every adversarial choice of the ray r, $\mathbf{Pr}[X \geq 2\alpha\mu] \leq N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$, for every $\alpha \geq 1$*

Proof. The idea is to consider only a few essential positions of the ray r and to sum the probability over all these positions. For this purpose, we define a set of overlapping parallelograms R_i having the property that any given core stripe $A\cup B$ is completely covered by two of these parallelograms. The first $l = \lceil 1/(2d)\rceil$ parallelograms cover the right lower triangle of the unit square $\mathcal{U}$. Parallelogram R_i $(i = 1,\ldots,l)$ 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 $\mathcal{U}$. Parallelogram R_{l+i} $(i = 1,\ldots,l)$ 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 it holds for all $\alpha \geq 1$ and $i \in [2l]$:

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

Hence, $\mathbf{Pr}[\exists i : X_i \geq \alpha\mu] \leq \sum_{i=1}^{2l}\mathbf{Pr}[X_i \geq \alpha\mu] = 2l\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$ and so $\mathbf{Pr}[X \geq 2\alpha\mu] \leq 2l\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu \leq N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$. □

The above tail bound on X holds for any adversarial choice of a ray r. Consequently, when letting r denote the Dantzig

ray, the bound holds for any outcome of this ray as well. Furthermore, it holds for any choice of the variable Y, too, as the dependence between the variables X and Y is only via the position of the Dantzig ray. As a consequence, for every $g \geq 0$,

$$\begin{aligned}\sum_{k=0}^{\infty} \mathbf{Pr}[X=k \,|\, Y=g]\,(k+7)^5 \\ \leq\ & \mathbf{Pr}[X \leq 4\mu \,|\, Y=g]\,(4\mu+7)^5 \\ & + \sum_{\alpha=2}^{\infty} \mathbf{Pr}[X \geq 2\alpha\mu \,|\, Y=g]\,(2(\alpha+1)\mu+7)^5 \\ \leq\ & (4\mu+7)^5 + \sum_{\alpha=2}^{\infty} N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^{\mu} (2(\alpha+1)\mu+7)^5 .\end{aligned}$$

Using $\mu = 2dN = 2c_d \log^3 N$ yields

$$\begin{aligned}\sum_{\alpha=2}^{\infty} N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^{\mu} (2(\alpha+1)\mu+7)^5 \\ =\ \mu^5 \sum_{\alpha=2}^{\infty} N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^{2c_d \log^3 N} \left(2\alpha+2+\frac{7}{\mu}\right)^5 .\end{aligned}$$

For any fixed $c_d > 0$, this term is bounded by $O(\mu^5)$. Consequently, there exists a constant $c_3 > 0$, such that

$$\sum_{k=0}^{\infty} \mathbf{Pr}[X=k \,|\, Y=g]\,(k+7)^5 \ \leq\ c_3 \log^{15} N .$$

Applying this upper bound to Equation 3.1 gives

$$\mathbf{E}[T] \ \leq\ c_2 c_3 N \log^{15} N \cdot \sum_{g=0}^{\infty} \mathbf{Pr}[Y=g]\, 2^g . \tag{3.2}$$

We further simplify.

$$\sum_{g=0}^{\infty} \mathbf{Pr}[Y=g]\, 2^g = \mathbf{E}\left[2^Y\right] = \mathbf{E}\left[2^{X_{(A\cup B)\cap F}}\right] \leq \mathbf{E}\left[2^{X_F}\right] . \tag{3.3}$$

The latter term can be bounded as follows.

LEMMA 3.3. $\mathbf{E}\left[2^{X_F}\right] = e^2$.

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,

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

□

Finally, combining Lemma 3.3 with the Equations 3.2 and 3.3 yields

$$\mathbf{E}[T] \ \leq\ c_2 c_3 N \log^{15} N\, \mathbf{E}\left[2^{X_F}\right] \ \leq\ c_2 c_3 e^2 N \log^{15} N .$$

Thus Theorem 3.1 is shown. □

4 Algorithm FastCore 2: 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. The analysis in [2] shows that the expected running time for computing all dominating sets under the uniform distribution is only $O(N^4)$. Thus, neglecting dependencies, this approach promises an expected running time of $N^{-3}O(N^4) = O(N)$ for the backup routine. Let us have a closer look at this approach. The running time is mainly determined by the number of dominating sets. Let q_{all}, q_{in}, and q_{out} denote the number of dominating sets over all items, over the items inside and over the 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 $\neg\mathcal{E}$, and, hence, the running time of the Nemhauser/Ullmann algorithm conditioned on $\neg\mathcal{E}$ might be much larger than $O(N^4)$.

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 computation does not 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 the variable q_{in} heavily depends on the event $\neg\mathcal{E}$ 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 capacity bound 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 as described by Horowitz and Sahni in [10]. They use this technique to re-

duce the worst case running time of the Nemhauser/Ullmann algorithm from $O(2^n)$ to $O(2^{n/2})$. We use their idea to deal with the dependencies in our average-case analysis. Using this technique, the expected number of dominating sets generated by our algorithm can be estimated by $\mathbf{E}[q_{\text{in}}] + \mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$. We have shown $\mathbf{E}[q_{\text{in}}] = O(N\text{polylog}N)$ in the proof of Theorem 3.1. The following analysis mainly deals with bounding $\mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$.

THEOREM 4.1. *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 with $d = 5c_0(\log N)^3/N$, where c_0 is the constant given in Lemma 2.1. 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. For the purpose of a simple analysis, 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 upper-bounded X_G by X_F and $F \supseteq G \cup H$. Including H into the core stripe 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 items in $C \cup D$, respectively. Furthermore, let $\mathcal{E}$ be the event that the core stripe was chosen sufficiently large, 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. This way,

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

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

$$\mathbf{E}[T_{C\cup D} \mid \neg\mathcal{E}] \;\leq\; \frac{N\text{polylog}N}{\mathbf{Pr}[\neg\mathcal{E}]} \;\;,$$

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 in region 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 item follows a uniform distribution with density at most N. Thus for all $k \in \mathbb{N}$,

$$\mathbf{E}[T_{C\cup D} \mid \neg\mathcal{E} \wedge X_{C\cup D} = k] = O(k^5 N) \;\;.$$

Let us switch to a more compact notation and define $X = X_{C\cup D}$. It remains to show

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

Observe that Lemma 2.1 with $d = 5c_0(\log N)^3/N$ yields $\mathbf{Pr}[\neg\mathcal{E}] \leq 2^{-5\log N} = N^{-5}$. Hence, it suffices to show that $\mathbf{E}\left[X^5 \mid \neg\mathcal{E}\right] = O(\max\{N^5, \mathbf{Pr}[\neg\mathcal{E}]^{-1}\})$.

The idea is that the random variable X is very sharply concentrated around its mean $\mathbf{E}[X] < N$ so that conditioning on $\neg\mathcal{E}$ does not significantly change the expected value of X. For every $\tau \geq 1$,

$$\begin{aligned}\mathbf{E}\left[X^5 \mid \neg\mathcal{E}\right] \;&=\; \sum_{i=1}^{\infty} \mathbf{Pr}\left[X^5 \geq i \mid \neg\mathcal{E}\right] \\ &\leq\; \tau + \sum_{i=\tau}^{\infty} \mathbf{Pr}\left[X^5 \geq i \mid \neg\mathcal{E}\right] \;\leq\; \tau + \sum_{i=\tau}^{\infty} \frac{\mathbf{Pr}\left[X^5 \geq i\right]}{\mathbf{Pr}[\neg\mathcal{E}]} \;\;.\end{aligned}$$

As X follows a Poisson distribution with mean at most N, $\mathbf{Pr}\left[X^5 \geq (2\alpha N)^5\right] \leq \exp(-\frac{1}{2}\alpha N)$, for every $\alpha \geq 1$. Hence, for $i \geq (2N)^5$, $\mathbf{Pr}\left[X^5 \geq i\right] \leq \exp\left(-\frac{1}{4}i^{1/5}\right)$. Now setting $\tau = (2N)^5$ gives

$$\begin{aligned}\mathbf{E}\left[X^5 \mid \neg\mathcal{E}\right] \;&\leq\; (2N)^5 + \mathbf{Pr}[\neg\mathcal{E}]^{-1} \sum_{i=\tau}^{\infty} \exp\left(-\tfrac{1}{4}i^{1/5}\right) \\ &=\; (2N)^5 + \mathbf{Pr}[\neg\mathcal{E}]^{-1} O(1) \;.\end{aligned}$$

This completes the proof of Theorem 4.1. □

5 Correlated instances

Several experimental studies [7, 8, 14, 15] do not only investigate uniformly random instances but also some other, harder classes of random inputs, e.g., so-called "weakly correlated" instances. We define δ-*correlated instances*, $0 < \delta \leq 1$, a parameterized version of *Weakly correlated instances* (the latter correspond to δ-correlated instances with $\delta = 10\%$) as follows. All 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 an upper bound on the integrality gap Γ for δ-correlated instances.

LEMMA 5.1. *There is a constant c_0 such that for every $1 \leq \alpha \leq \log^4 N$, $\mathbf{Pr}\left[\Gamma \geq c_0\alpha\frac{\delta}{N}\log^2\frac{N}{\delta}\right] \leq 2^{-\alpha}$.*

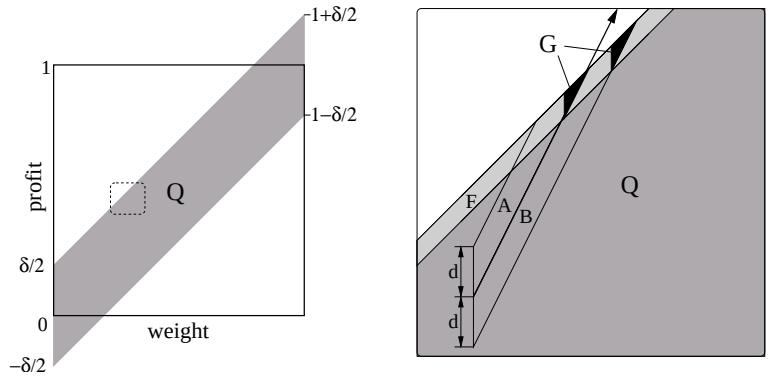

Figure 3: For δ-correlated instances items are uniformly sampled from region Q. The right figure (magnified rectangle from the left figure) shows an example for core stripes A and B with height d. Items in region $G \subseteq F$ have insufficient randomness.

Proof. (sketch). We adapt the proof from Lueker [11] for uniform instances. Please refer to [11] 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. Define $k := \log_4(N/\delta)$ and $\varepsilon = O(k4^{-k})$. 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 about 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 $S_1, S_2, \ldots$ of about $2k$ items trying to find a subset $S \subset S_i$ with weight in $[cap - 2\varepsilon, cap]$. The probability that we find such a subset is at least $\frac{1}{2}$ for every S_i. This can be shown with the help of the following lemma, which essentially shows that subsets of random numbers lay exponentially dense.

LEMMA 5.2. *(Lueker [11]) 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-item subset $S \subset [2k]$ 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 profit-gap between the approximate and the optimal fractional solution has two contributions: residual capacity and cumulated loss of packed items. When the approximation algorithm find a suitable subset S, the residual capacity of the knapsack is at most 2ε causing a loss in profit of at most $r2\varepsilon = O(\frac{r\delta}{N} \cdot \log\frac{N}{\delta})$, where $r = p_\kappa/w_\kappa$ is the slope of the Dantzig ray. With high probability the slope r is upper-bounded by a constant term depending on β. The cumulated loss can be estimated by $cap(r - r_l) \le k(r_0 - r_l)$ where l denotes the number of iterations performed by the algorithm and r_0 and r_l are the slopes 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. The accumulated loss of items in $S \subset S_l$ is $O(l\delta k^2/N)$. In each iteration the success probability is at least $\frac{1}{2}$, therefore $\mathbf{Pr}[l > \alpha] \le 2^{-\alpha}$, for all $\alpha \in \mathbb{N}$. □

5.2 Running time

We adapt algorithm FastCore 2 to the new situation by using a smaller core stripe, that is, we set $d = c_d \frac{\delta}{N} \log^3 \frac{N}{\delta}$ instead of $d = c_d N^{-1} \log^3 N$. This way, we obtain the following result.

THEOREM 5.1. *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 $d = c_d \frac{\delta}{N} \log^3 \frac{N}{\delta}$ we expect about $2c_d \log^3 \frac{N}{\delta}$ core items in contrast to $2c_d \log^3 N$ 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).

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

For items in $(A \cup B) \setminus G$ the maximum density of the profit distribution is N/δ. Therefore, for any $j \in \mathbb{N}$, $\mathbf{E}\left[q_{(A\cup B)\setminus G} \,|\, X_{(A\cup B)\setminus G} = j\right] \le c_1(N/\delta)j^4 + 1$. Items in G have larger densities, so we again pessimistically assume that each of these items doubles the number of dominating sets. We have chosen the size of F so that on average there is only one item in F. This way our analysis, which accounts also for items in G, applies here as well. □

$1/\delta$	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: Average running time (in seconds) for δ-correlated instances of size $n = 10000$ without preprocessing. Each entry gives the average over 1000 test instances. Columns 2–4 give the numbers for three variants of our algorithm implementing a specifi ed subset of features: DomF = dominating-set fi lter; LossF = Loss-fi lter; combo = knapsack solver by Pisinger. We fi xed $\beta = 0.4$ for all experiments.

6 A few notes on experimental results

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 Marchetti-Spaccamela considering only items in a stripe around the Dantzig ray. In contrast to our theoretical analysis, our experimental study uses a dynamically growing core. This way the algorithm automatically finds the optimal core size. The reason why we immediately switch to a dynamically growing core is that any algorithm with a static core cannot seriously compete with this dynamic approach. In the theoretical analysis we only assumed a static, fixed size core because a dynamically growing core introduces additional dependencies that we cannot analyze.

Our experimental study shows, if we combine all the concepts presented in the preceding sections, then we obtain an implementation that can outperform the best known previous implementations. In particular, we apply the loss filter from Goldberg and Marchetti-Spaccamela as well as the dominance filter based on the Nemhauser/Ullmann algorithm. In addition, we use two lists, which are finally merged using the linear time algorithm of Horowitz and Sahni. Our implementation beats the best previous implementation by Pisinger (combo code with 64-bits integer arithmetic [7]) on δ-correlated instances by several orders of magnitudes, but only if we add all the features listed above. In fact, for uniformly random instances the running time is dominated by the time to find the optimal fractional solution and there are no significant differences between different implementations. The experimental results for δ-correlated instances are much more interesting. Table 1 presents first measurements obtained on a Sun Fire^{TM}15K. As our theoretical results suggest, the running time increases slightly super-linear in $1/\delta$ for most of the implementations. The implementation using all the features shows even a slightly sub-linear behavior. We want to point out that these experimental result are only preliminary, and we plan to continue with a more thorough study.

7 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. V¨ocking. Random Knapsack in Expected Polynomial Time. *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, 232–241, 2003.

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

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

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

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

[7] 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.

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

[9] 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.

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

[11] G. S. Lueker. On the Average Difference Between the Solutions to Linear and Integer Knapsack Problems. *Applied Probability - Computer Science, The Interface*, Vol. 1, Birkh¨auser 1982.

[12] 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.

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

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

[15] 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.