Fast Additive Constant Approximation Algorithms for The Safe Deposit Boxes Problem with Two and Three Currencies

Boaz Ben-Moshe[1] Yefim Dinitz[2]

[1]Department of Computer Science, Ariel University Center of Samaria, Ariel 44837, Israel
[2]Department of Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel

Abstract. The following variant of the knapsack problem is considered; Suppose there are n safe deposit boxes, each containing known amounts of m currencies, and there is a certain need for each currency. The problem is to open the minimal number of boxes, in order to collect at least the prescribed amount of each currency. In a technical report, by Y. Dinitz and A. Karzanov, it is shown that this problem is NP-hard, and that its non-degenerate case can be solved within an absolute error of at most $m-1$ in $O(n^{m+1})$ time, assuming m is constant. They also suggested an $O(n^2)$ algorithm, with an absolute error of at most 1, for the general case of two currencies.
We suggest new combinatorial algorithms with significantly improved runtime for the two and three currency cases ($2D, 3D$). The $2D$ algorithm runs in $O(n \log^2 n)$ time, while the $3D$ algorithm runs in $O(n^2 \log^2 n)$ time. In addition to linear programming techniques, used in previous works, we also use the parametric search approach of N. Megiddo [10] for decreasing the running time. The degenerate case is formulated as an interesting problem, solved by combinatorial techniques.

1 Introduction

Let us consider the following integer optimization Safe Deposit Boxes (SDB) problem, suggested by A. S. Kronrod and described and solved in [3]. It is a special case of the multidimensional knapsack problem that has been extensively studied, particularly in the maximization version [1, 4, 5, 9].

Given n safe deposit boxes, where the i^{th} box contains a_i dollars and b_i euros. The problem is to choose the minimal number of boxes so that the total amount of dollars obtained is at least A and that of euros is at least B. It is assumed that all boxes together contain at least A dollars and B euros. The generalization to the case of m currencies is straightforward.

SDB can be represented as the following Boolean Programming problem:
$\min \sum_{i=1}^{n} x_i$, such that $\sum_{i=1}^{n} a_{ji} x_i \geq A_j$, where:
$j = 1, ..., m;\ i = 1, ..., n;\ x_i \in \{0, 1\};\ \forall_{i,j} : a_{ji} \geq 0.$

Previous work: SDB was introduced by Kronrod with the following preamble: "It is clear how to solve any safe deposit box problem with a single currency: the most valuable boxes should be selected, till the target amount is reached. But can it be generalized for two currencies?" This problem turned out to be NP-hard; however, a combinatorial algorithm that solves it with an absolute error of at most 1 was found. Such a quasi-optimal solution is formed by a set of boxes most valuable w.r.t. a certain exchange rate dollar:euro, and the algorithm finds such a rate. Moreover, for the non-degenerate case of m currencies, for an arbitrary m, it was observed that, for an instance of Safe Deposit Boxes, if an extreme-point fractional solution of its linear relaxation is given, the ceiling rounding of that solution is an $(m-1)$-absolute approximation for the original problem. The algorithm of [3] for two currencies works in time $O(n^2)$, including the degenerate case. For finding a fractional solution as above, an algorithm of $O(n^{m+1})$ runtime (assuming m is a constant) is

suggested there (see a similar result in [8]). Note that there are only few non-trivial approximation algorithms with a constant additive error, in the literature [6, 7, 12].

One can find several applications of the SDB problem, involving non-convertible resources ("currencies") related to any objects ("boxes"). For example, Moret [11] points out that in Operating Systems, the need to kill (or roll back) the smallest number of processes, in order to release essential resources (e.g. CPU and memory), may be formulated as such a problem.

Our contribution: This paper improves the $O(n^2)$ running time of Dinitz and Karzanov, by presenting an $O(n \log^2 n)$ runtime algorithm for the two currencies case with additive error one. Our algorithms make use of parametric search techniques [2, 10], while addressing some specific details of SDB. For the case of three currencies, we suggest an $O(n^2 \log^2 n)$ algorithm, improving the LP-rounding approach with running time $O(n^4)$ of Dinitz and Karzanov. Both algorithms are also extended to deal with the degenerate case using new combinatorial methods.

2 Duality-Based Bound for Two Currencies

In this section, following [3], we analyze in detail some aspects of the two currencies Safe Deposit Boxes problem, in order to acquire a base for combinatorial algorithms; most of the following observations can be straightforwardly generalized to the case of an arbitrary number of currencies. Let us introduce variables α and β (they are, in fact, the dual variables of the linear relaxation of SDB). We say that the **exchange rate $\boldsymbol{\alpha : \beta}$** is defined if values $\alpha \geq 0$ of one dollar and $\beta \geq 0$ of one euro in some third (imaginary) currency are fixed. Let

$$v_i = v_i(\alpha, \beta) = \alpha \cdot a_i + \beta \cdot b_i$$

be the **value** of box i, w.r.t. rate $\alpha : \beta$. We say the one box is "better" than another one, if it is more valuable, w.r.t. the considered rate.

A feasible solution is called **quasi-optimal** if the number of selected boxes is not greater than minimal possible plus one. The following theorem is proved in the next section.

Theorem 1. *For any Safe Deposit Box instance, there exists a (quasi-optimal) exchange rate, such that a certain quasi-optimal solution consists of boxes most valuable according to this rate. Moreover, such a rate exists in the set* $\{1 : 0;\ 0 : 1 = 1 : \infty;\ 1 : \frac{a_i - a_j}{b_j - b_i},\ 1 \leq i\ < j \leq n\}$.

Any subset of boxes as well as the set of corresponding numbers $I \in \overline{1, n}$ will be called a **plan**. Let us denote $v(I) = \sum_{i \in I} v_i$. A plan is called **feasible** w.r.t. the first constraint (by dollar) or w.r.t. the second constraint (by euro) if $\sum_{i \in I} a_i \geq A$ or, respectively, $\sum_{i \in I} b_i \geq B$. Further on, we rely on the following simple statement stemming directly from non-negativity of a_i, b_i, α, and β.

Lemma 1 (Monotonicity Lemma). *Let I and I' be two plans where $I' \supset I$. Then $v(I') \geq v(I)$. Moreover, if I is feasible w.r.t. some constraint, then I' is also feasible w.r.t. the same constraint.*

Optimality and quasi-optimality of plans constructed in the subsequent part of the paper are based on the following bounds.

Lemma 2 (Lower Bound Lemma). *Let there exist an exchange rate $\alpha : \beta$ and an integer k, such that some plan consisting of k boxes most valuable w.r.t. $\alpha : \beta$ is infeasible by both dollar and euro. Then the optimum is at least $k + 1$*

Proof. Infeasibility of such a plan, say I, by both dollar and euro implies that

$$v(I) = \sum_{i \in I} (\alpha \cdot a_i + \beta \cdot b_i) = \alpha \sum_{i \in I} a_i + \beta \sum_{i \in I} b_i < \alpha \cdot A + \beta \cdot B \ .$$

By the assumption of this Lemma and by Monotonicity Lemma, $v(I)$ is maximal possible for the exchange rate $\alpha : \beta$ over all the plans consisting of k or a smaller number of boxes. Therefore, for any plan I', $|I'| \leq k$,

$$v(I') \leq v(I) < \alpha \cdot A + \beta \cdot B$$

also holds. On the other hand, for any feasible plan I'' we have

$$v(I'') = \alpha \sum_{i \in I''} a_i + \beta \sum_{i \in I''} b_i \geq \alpha \cdot A + \beta \cdot B \ .$$

Hence, $|I''| \leq k$ holds for no feasible plan I'', as required.

Now, let us see how the non-increasing ordering of boxes by value w.r.t. the rate $1 : \beta$ (henceforth, "ordering at β") changes, when β changes from 0 to ∞. We call the value $\beta(i,j) = \frac{a_i - a_j}{b_j - b_i}$, if defined and positive, a **swap value**. It is easy to see that when β passes through this value, boxes i and j "swap" in the order: before it, the box with more dollars is better, at it, they are equal, while after it, the box with more euros is better.

Let us give a geometrical interpretation. In Figure 1(a), the straight lines represent graphically the dependency $v_i(\beta) = b_i \cdot \beta + a_i$. The abscissa of the intersection point of straight lines i and j, if positive, is the swap value of $\beta(i,j)$.

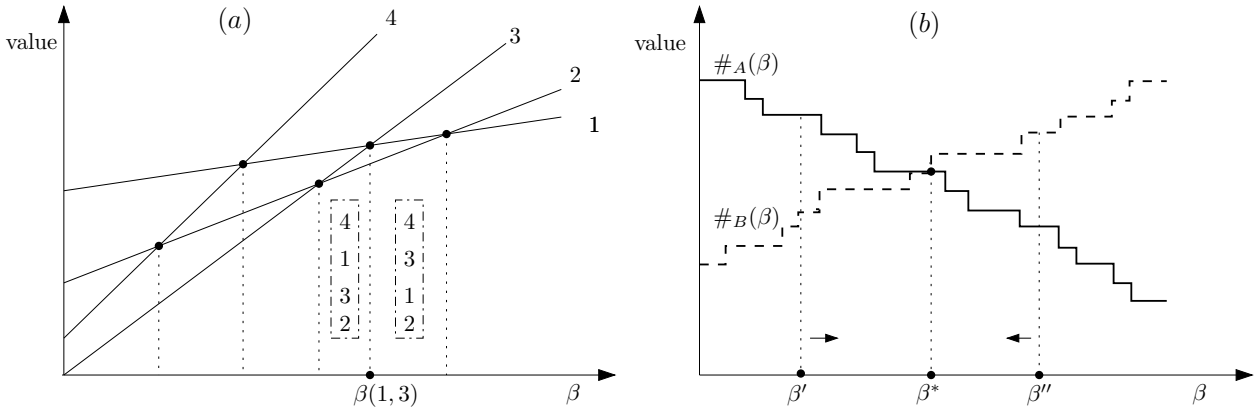

Fig. 1. (a): Geometric representation of box values as function of β. (b): Minimal number of best boxes satisfying constraints A and B, as a (monotone) function of β.

3 Non-Degenerate Two Currencies Case

3.1 Proof of theorem 1

Proof. Any value β_0 of β defines the (unique) non-increasing *preference* order of boxes $\preceq_{\beta_0}$ by the value of $v = v(\beta_0)$. Observe that if β_0 is not a swap value, the preference order is strictly decreasing

(this is up to transposition of identical boxes, which will not be mentioned further). For such values of β, let us define two functions: $\#_A = \#_A(\beta_0)$ and $\#_B = \#_B(\beta_0)$, as the minimal number of best boxes satisfying constraint A or B, respectively. Observe that, for any β, $\max\{\#_A(\beta_0), \#_B(\beta_0)\}$ best boxes form a feasible plan, i.e., $\max\{\#_A(\beta_0), \#_B(\beta_0)\}$ is an upper bound for the optimum, while by Lemma 2, $\min\{\#_A(\beta_0), \#_B(\beta_0)\}$ is a lower bound for it.

The box order, and thus each one of these functions is a constant on the interval between any two neighboring swap values of β; hence, it is possible to consider all of them as functions of these *β-intervals*. For any swap value β_0, let us define orders $\preceq_{\beta_0-}$ and $\preceq_{\beta_0+}$ as the orders at the intervals next to the left and next to the right from β_0; clearly, both of them are refinements of the preference order at β_0. Note that the order $\preceq_{\beta_0-}$ (resp., $\preceq_{\beta_0+}$) can be obtained from $\preceq_{\beta_0}$ by its refinement, where the boxes of the same $v(\beta_0)$-value are ordered by descending of their dollar (resp., euro) value. By the non-degeneracy assumption, exactly two boxes, say box i and box j, have equal values, at any swap value β_0; w.l.o.g., $a_i > a_j$ and $b_i < b_j$, and thus the single change from $\preceq_{\beta_0-}$ to $\preceq_{\beta_0+}$ is that box i was better than box j, and they swap.

The above inequalities imply an important observation:

Lemma 3. *Function $\#_A(\beta)$ is non-increasing, while $\#_B(\beta)$ is non-decreasing (as shown in Fig. 1(b)).*

Since the above box swap is the single change in the box order, values of $\#_A$ and $\#_B$ change by at most one, between two neighboring β-intervals (the "discrete continuity property"). Indeed, let S denote the best box set taken for satisfying a certain constraint X. If the swap is entirely inside or outside S, then for the new order, S satisfies constraint X as well. The only case when the value of $\#_X$, $X = A, B$, can change to a greater one is if exactly the *last* box taken for satisfying the constraint X and the next worse one are swapped, which causes unsatisfiability by the same number of boxes. Then, adding back the single swapped out box would satisfy X for sure. The case of changing to a lesser value is symmetric, where the critical swap is between the two last members of S.

By the above monotonicity and discrete continuity, the difference function $\#_A - \#_B$ is non-increasing and changes its values by at most 2 at each swap point. Therefore, there must exist a β-interval Q, where $|\#_A(Q) - \#_B(Q)|$ is at most 1. If $\#_A(Q)$ is equal to $\#_B(Q)$, the best $\#_A(Q)$ boxes form an *optimal solution*. Otherwise, the feasible plan consisting of the $\max\{\#_A(Q), \#_B(Q)\}$ best boxes is a *quasi-optimal solution*, since $\min\{\#_A(Q), \#_B(Q)\} = \max\{\#_A(Q), \#_B(Q)\} - 1$ is a lower bound for the optimum. Hence, item (i) of Theorem holds. Let us stress that all values in the *closed* interval Q give rise to this solution; we will call Q and all these values of β *quasi-optimal*. A swap value β_0 bordering a (quasi-)optimal interval may be considered as a (quasi-)optimal value too, since the preference order of boxes at this interval is one of the legal orders at β_0.

Based on theorem 1, a naive algorithm for the (non-degenerate) two currencies case can be designed, using the following two steps: (i) find all equilibrium values of β. (ii) test each β for a quasi-optimal solution. The initial idea of [3] was to compute and sort all swap values of β, along with its swapping pair of boxes. After that, beginning from the first β value, and subsequently update it for every β-interval, from the left to right, up to finding the β^* swap value. Then, a quasi-optimal solution is obtained as described above. The dominating time is $O(n^2 \log n)$ for sorting the $O(n^2)$ swap values of β; the time for other operations is $O(n^2)$. In [3], an improvement of the running time to $O(n^2)$ is presented. Using methods from computational geometry, all intersection points of n straight lines in the positive quadrant, in the non-decreasing order of β, can be found in time $O(n^2)$.

3.2 An improved algorithm for the two currencies case - Non-Degenerate

Our new algorithm also begins with analyzing the cases $\beta = 0, \infty$. After that, it uses binary search over the set of swap values of β, instead of its linear scanning (see Fig. 1(b)).

Lemma 4. *For any value of β, it is possible to answer the query whether the rate $1 : \beta$ is quasi-optimal, or on which side (lesser or greater) of this value such a rate exists, in $O(n \log n)$ time.*

Proof. For a given value of β_0. Sort the boxes by $v(\beta_0)$ (in decreasing order). Let $\#_A(\beta_0)$ be the smallest prefix of sorted boxes that satisfies the A-constraint. If it turns out that $|\#_A(\beta_0) - \#_B(\beta_0)| \leq 1$, a quasi-optimal solution would be easily found. Otherwise, if $\#_A(\beta_0)$ is greater than $\#_B(\beta_0) + 1$, a quasi-optimal value of β exists to the right of β_0, while in the other symmetric case, it exists to the left of it. It is easy to see that the time of sorting is dominating, so the time bound $O(n \log n)$ for answering the query holds, as required.

For each analyzed value of β, the algorithm sorts boxes w.r.t. it, processes them, and defines how to continue the binary search, according to Lemma 4. It is known that, given q straight lines in the plane, and an integer $r : 1 \leq r \leq q(q-1)/2$, it is possible to find their r^{th} pairwise intersection in the increasing order along any axis in time $O(q \log q)$ [2]. Using this method, the total time for any iteration of the binary search is $O(n \log n)$, so the total running time of the algorithm is $O(n \log^2 n)$.

Theorem 2. *For the case of two currencies, an SDB problem can be solved, with an additive error at most 1, in $O(n \log^2 n)$ time.*

4 Generalization to Degenerate Case, for Two Currencies

Let us generalize the algorithm of Section 3.2 to the case when, for some value of β, more than one pair of boxes have the same value. Now, at any swap value β_0, there may be several groups of boxes, which have the same value. The order in each group turns over, between the two intervals separated by β_0, from that by non-increasing value of the a-component, to that by non-increasing value of the b-one. For $X = A, B$, function $\#_X$ may be affected only if the prefix of $\#_X$ best boxes ends inside some equality group; in what follows, we say "$\#_X$ falls into that group". Monotonicity of functions $\#_A$ and $\#_B$ remains, but the discrete continuity may not hold. The definition of β^* remains the same: $\#_B(\beta^*+) \leq \#_A(\beta^*+)$, while $\#_A(\beta^*-) < \#_B(\beta^*-)$. It can be easily shown that, w.r.t. both $\preceq_{\beta^*+}$ and $\preceq_{\beta^*-}$, $\#_A$ and $\#_B$ fall into the same group.

Let us denote this group by S', and the set of boxes better than those in that group by S. The plan S satisfies none of the constraints, while $S \cup S'$ is a feasible plan. Let us reduce the original problem $\mathcal{P}$ to problem $\mathcal{P}'$ as follows: retain the boxes in S' only, and define $A' = A - \sum_{i \in S} a_i$, $B' = B - \sum_{i \in S} b_i$. We will find a (+1)-approximation to the optimum of $\mathcal{P}'$, and its union with S will be such an approximation for $\mathcal{P}$.

In what follows, we consider $\mathcal{P}'$. Notice that, now, all boxes have the same value, V. Let us define $k' = \lceil (A' + B')/V \rceil$; note that k' is a lower bound for the optimum. Observe also that (i) any plan of size k' satisfies at least one of the constraints, and (ii) any feasible plane of size $k' + 1$ is quasi-optimal.

Algorithm G2: Compute S, S', A', B', and k'. Consider the array of boxes in S' sorted by the a value. Let us move the window of size k' from the top of it to its bottom, box by box. By

the definition of β^*, at the beginning, the A' constraint is satisfied by the window contents, while at the end, the B' constraint is satisfied by it. Let us stop at the first time when the B' constraint becomes satisfied. If this happens at the very beginning, the window contains an optimum solution. Otherwise, notice that the previous window satisfied the A' constraint (since it did not satisfy the B' one). We add to the current window the box removed from it at the last step, arriving at a feasible plan of size $k' + 1$. We add it to S and return this solution.

5 Non-Degenerate Three Currencies Case

Let us analyze the non-degenerate case of the SDB problem with three currencies, denoted by $\mathcal{P}$. Our goal is to find a quasi-optimal solution, which in this case is allowed to have an additive error at most 2. In this section, we assume the general position case, that is, when no two objects (e.g., intersection lines or points) coincide, if there exists an infinitesimal modification of the data making them distinct.

Let us consider the 3D space with coordinates β, γ, and v. For any box, i, its value, $v_i = v_i(\beta, \gamma)$, is the linear function $a_i + \beta \cdot b_i + \gamma \cdot c_i$, which defines the box i plane in 3D, P_i. For any two boxes, if their planes intersect, then the projection of their intersection is the border straight line between the two areas of equal order of these two boxes. Let us denote it by $l_{ij} = l_{ji}$, for boxes i and j. Let us partition the positive quadrant of the (β, γ)-plane into the regions of equal box value order. The total division of the plane into regions is formed by cutting it by all those straight lines. Observe that the intersection point of three planes P_i, P_j, and P_r is seen in the projection picture as the common intersection point of the *three* lines l_{ij}, l_{jr}, l_{ri}. General position implies that no fourth line goes through such a triple intersection point, and that no other triple line intersection is at this point. On the other hand, the intersection of some two lines, l_{ij}, l_{rs}, is just the common projection of two crossing, but not intersecting in 3D. Let us project all these line intersection points to the γ axis, resulting in the set of *swap values* of γ, dividing the γ axis into the γ-intervals. Note that there is $O((n^2)^2) = O(n^4)$ line intersection points (only $O(n^3)$ of them are three planes intersections). By [2], for any r, the r^{th} one of them, in the increasing order of γ, can be found in $O(n^2 \log n)$ time.

5.1 Auxiliary two currencies problem

Let us analyze the situation by the sweeping line method. Let us consider the straight line in the (β, γ)-plane defined by fixing $\gamma = \gamma_0$. We define the *auxiliary γ_0-two currencies SDB problem* as follows. The currencies are named AC and B, the values of box i are $a_i + \gamma_0 \cdot c_i$ and b_i, respectively, and the constraint right hand sides are $A + \gamma_0 \cdot C$ and B, respectively. Let us denote the leftmost equilibrium value of β, for the γ_0-problem, by $\beta^*(\gamma_0)$ (see the definition in Section 3.1). It is easy to see that any feasible solution of the original problem is a feasible solution for γ_0-problem. The inverse is not guaranteed, but it can be easily shown that any solution feasible for γ_0-problem must satisfy the constraint B and at least one of constraints A and C. The direction of the following analysis is to build a binary search algorithm to find an equilibrium value γ^*, such that the preference order $\preceq_{\beta^*(\gamma^*)}$ will satisfy all the three constraints almost simultaneously, resulting in a quasi-optimal solution.

Let us see how the structure of the auxiliary two currencies γ_0-problem is related to the structure of the original three currencies problem. Fixing $\gamma = \gamma_0$ defines the (β, v) plane $P(\gamma_0)$. Its intersection with plane P_i is the straight line $l_i(\gamma_0)$, which is the graph of the value of box i, defined as $v(\beta, \gamma_0) = (a_i + \gamma_0 \cdot c_i) + \beta \cdot b_i$. The intersections of $P(\gamma_0)$ with the lines l_{ij} define the swap

values of β, at the γ_0-problem, where the order of boxes i and j swaps. These points together break the β axis into the intervals of constant box order. In fact, these intervals are the intersections of $P(\gamma_0)$ with the aforementioned regions of constant box order in the (β, γ)-plane. As a consequence, moving the sweep line in the interval between the neighboring swap values of γ does not change the combinatorial structure of the parameterized auxiliary problem: the sequence of β-intervals, together with the box orders in them, remains the same, while the swap values of β move without interchanging, in their order. We denote the values related to the γ_0-problem by $\cdot(\gamma_0)$, and the point $(\beta^*(\gamma_0), \gamma_0)$, in the β, γ-plane, by $M^*(\gamma_0)$.

5.2 Binary Search

Let us define our binary search rule as follows. Similarly to the two currencies case, we first analyze values 0 and ∞ of γ, resulting either in a quasi-optimal solution, or in the $[0, \infty]$ search area, at the γ axis, where $\#_A(M^*(0)) < \#_C(M^*(0))$ and $\#_C(M^*(\infty)) < \#_A(M^*(\infty))$; in what follows, the former (resp., latter) inequality, will be called *A/C- (resp., C/A-)situation*, for some value of γ. For $\gamma = 0$, if $\#_A(M^*(0)) \geq \#_C(M^*(0))$, we can output the quasi-optimal solution of the 0-problem. For $\gamma = \infty$, if $\#_C(M^*(\infty)) \geq \#_A(M^*(\infty))$, we can output the quasi-optimal solution of the ∞-problem. This solution is found as follows: we divide the boxes into groups with the same c value; we take boxes, group by group, by decreasing c, until both constraints A and B are satisfied; if this happens in the same group, we solve the two currencies problem for this group separately, otherwise, we remain with just the one currency problem, for this group.

The opposite A/C- and C/A-situations at the ends of the search area are maintained during the binary search, as follows. At any iteration, after finding the r^{th} swap value γ^r, we define the (γ^r-)- and (γ^r+)-auxiliary problems. If there is the C/A-situation at the (γ^r-)-problem, we continue with the first half of the search area, before the r^{th} swap value; if there is the A/C-situation at the (γ^r+)-problem, we continue with the second half, after this value; if there is the switch from A/C to C/A around γ^r, we define $\gamma^* = \gamma^r$, and find a quasi-optimal solution by means of Algorithm *Local*, given below. When the search area shrinks to a single $[\gamma', \gamma'']$ interval between two neighboring swap values of γ, we find a quasi-optimal solution by means of Algorithm *Sandwich*, given below. For an illustration of these two cases, see Figure 2.

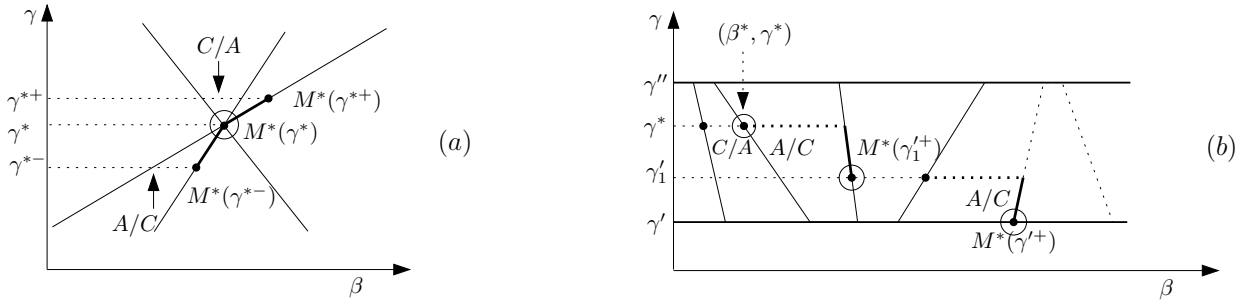

Fig. 2. Final phase: (a) 'local' case. (b) 'sandwich' case.

The (γ^r-)-auxiliary problem is, in fact, the $(\gamma^r - \epsilon)$-problem, for an infinitesimal ϵ. Its definition is combinatorially simulated as follows. Consider the γ^r-problem. The three boxes corresponding to the straight lines intersecting at $M^*(\gamma^r)$ have some order at the β-interval before $\beta^*(\gamma^r)$, which

flips to the inverse one at the β-interval after it. At the new problem, the former order undergoes swapping, step by step, at the three straight lines "below" $M^*(\gamma^r)$, in their order w.r.t. the β direction. All the other components of the γ^r-problem do not change, by the general position assumption. The (γ^r+)-problem is defined similarly.

5.3 Final phase: local case

Let us analyze the 'local' case of the switch from A/C to C/A around γ^r. By the definition of $M^*(\gamma^r)$, the following change occurs at it, in the γ^r-problem: at the right close β-interval, $\#_B(\gamma^r) \leq \#_{AC}(\gamma^r)$, while at the left close β-interval, $\#_B(\gamma^r) > \#_{AC}(\gamma^r)$. Let us denote by S' the set of three boxes whose order changes at $M^*(\gamma^r)$, and by S the set of boxes better than those three boxes, at the $M^*(\gamma^r)$-problem. Algorithm *Local* returns $S \cup S'$ as the quasi-optimal solution. Let us show its correctness.

Lemma 5. *The set S satisfies none of constraints A, B, C. The set $S \cup S'$ satisfies all of them.*

Proof. For the (γ^r-)-problem, the same change as above is made by three pairwise box order changes. There exists some of them, where the similar flip between "$\leq$" to "$>$" is caused by the interchange between some two boxes in S'. ¿From this, it is easy to see that S does not satisfy neither constraint B nor AC. Since there is the A/C-situation, constraint C is not satisfied by S as well. In a similar way, we see that $S \cup S'$ satisfies both constraints B and A. Similar considerations for the (γ^r+)-problem show that constraint A is not satisfied by S, while constraint C is satisfied by $S \cup S'$. All this implies both statements of Lemma.

By the first statement of Lemma, $|S|+1$ is a lower bound for the optimum of $\mathcal{P}$. By the second one, $S \cup S'$ is a feasible solution, and since $|S \cup S'| = |S| + 3$, $S \cup S'$ is quasi-optimal, as required.

5.4 Final phase: sandwich case

Let us now turn to the "sandwich" case of the switch from the A/C- to C/A-situation between the neighboring $(\gamma'+)$- and $(\gamma''-)$-problems, $\gamma' < \gamma''$. Consider the straight line L, passing through $M^*(\gamma')$, which defines the value $\beta^*(\gamma'+)$. At this value, a single interchange of two boxes causes the flip between $\#_B(\gamma'+) \leq \#_{AC}(\gamma'+)$ and $\#_B(\gamma'-) > \#_{AC}(\gamma'-)$. The naive point of view sees a contradiction: since γ' and γ'' are neighbors, the same regions, with the same box orders, define the $(\gamma'+)$- and $(\gamma''-)$-problems, the same straight line L must define the value $\beta^*(\gamma''-)$, which contradicts the assumption of the change from the A/C-situation at the $(\gamma'+)$-problem to the C/A-situation at the $(\gamma'-)$-one. However, this does not cover all circumstances.

A delicate observation is that the move from $\gamma'+$ to $\gamma''-$ may change the satisfaction of constraint AC. Indeed, by the A/C-situation assumption, the best $\#_{AC}(\gamma'-)$ boxes, satisfying constraint AC, may not satisfy constraint C. Hence the increase of their total AC-value, when γ grows, is slower than the increase of $A+\gamma \cdot C$. As a consequence, there may be some value γ'_1, $\gamma' < \gamma'_1 < \gamma''$, such that beginning from γ'_1+, constraint AC would not be satisfied (see Figure 2); this is checked at the beginning of Algorithm *Sandwich*. In this case, the situation $\#_B \leq \#_{AC}$ does not flip to the opposite one at L, since $\#_B(\gamma'_1+)$ becomes *equal* to $\#_{AC}(\gamma^1+)$. In this case, Algorithm *Sandwich* will check the next to the left swap values of β, at the (γ'_1+)-problem, one by one, to the direction of the flip from $\#_B = \#_{AC}$ to $\#_B \leq \#_{AC}$, resulting in another straight line defining the value $\beta^*(\gamma'_1+)$. Note that Algorithm *Sandwich* checks whether the situation flips from A/C

to C/A, whenever two boxes swap. In this case, thanks to $\#_B = \#_{AC}$, taking *both* the swapping boxes, together with better ones, results in a quasi-optimal solution; we denote the current point by (β^*, γ^*). If this does not happen, during finding $\beta^*(\gamma'_1+)$, we arrive at a situation similar to that at $\beta^*(\gamma'+)$, and Algorithm *Sandwich* continues in the same way, finding $\beta^*(\gamma'_r+)$, up to the flip of A/C to C/A. If for some $\beta^*(\gamma'_r+)$, Algorithm *Sandwich* stops to advance, since constraint AC is satisfied throughout to γ'', then $\beta^*(\gamma'_r+)$ is defined by the same straight line as $\beta^*(\gamma''-)$. Then, the situation must be already C/A, as at the naive point of view, described above.

Observe that the entire geometry of the strip between $\gamma = \gamma'$ to $\gamma = \gamma''$, in the (β, γ)-plane, is defined by $O(n^2)$ *non-intersecting* straight lines, dividing it into regions of equal box order. Note that the order of these straight lines is computed just once, when processing (γ'_1+)-problem, if any.

Algorithm *Sandwich* is finite, since $M(\cdot)$ moves to the left at each step, bounded by the γ axis and by the C/A-situation at the $(\gamma''-)$-problem. The running time is $O(1)$ for each one of the $O(n^2)$ elementary steps, and $O(n^2 \log n)$ for the region sorting, which is carried out at most once for one value of β in the search. The total time, for solving $\mathcal{P}$, is dominated by sorting the boxes at all $O(\log n)$ steps of binary search, which results in the following result.

Theorem 3. *For the case of three currencies, an SDB problem can be solved, with an additive error at most 2, in $O(n^2 \log^2 n)$ time.*

6 Generalization to Degenerate Case, for Three Currencies

Let us extend the approximation algorithm suggested in Section 5 to degenerate cases (proofs are omitted due to space limitations). It may happen that, in the "local" case (when there is A/C-situation at γ^*- and C/A-situation at γ^*+), more than three boxes may have equal values w.r.t. the order $\preceq_{M^*}$, or in the "sandwich" case, the point (β^*, γ^*) (where A/C- flips to C/A-situation) may lie on the straight line which is the projection of a few plane intersections, so that more than one pair of boxes have equal values and swap at (β^*, γ^*). We unify them, based on the analysis in Sections 3.1 and 5: In both cases, a point (β^*, γ^*) and a lower bound k are given, such that there are three box orders not contradicting $\preceq_{\beta^*,\gamma^*}$ which have the following three properties, respectively: (i) $\#_B(\beta^*, \gamma^*) \le k$; (ii) $\#_{AC}(\beta^*, \gamma^*) \le k$ in the A/C-situation; (iii) $\#_{AC}(\beta^*, \gamma^*) \le k$ in the C/A-situation. In what follows, assuming these properties, we show that there exists a (+2)-approximated solution. Statement proofs are omitted.

Lemma 6. *For pairs of constraints {A,B}, {A,C}, {B,C}, there exist plans P_{AB}, P_{AC}, P_{BC} of size $k+1$, satisfying them, respectively.*

Similarly to Section 4, all $\#_A$, $\#_B$, $\#_C$ fall into the same group of boxes S' of the same value; we denote by S the set of boxes better than those in S'. Notice that Lemma 5, which originally related to the case $|S'| = 3$, remains valid. As in Section 3.1, we move to the reduced problem $\mathcal{P}'$, on the set of boxes S', s.t. there is a one-to-one correspondence between the feasible plans of $\mathcal{P}$ and of $\mathcal{P}'$. For simplicity, we scale $b_i \to \beta^* \cdot b_i$ and $c_i \to \gamma^* \cdot c_i$, arriving at the equal sum $a_i + b_i + c_i = D$, for all boxes in S'. Denote by A', B', C' the scaled reduced constraint bounds, and by P'_{AB}, P'_{AC}, P'_{BC} the reduced plans. Let k' be $\lceil (A' + B' + C')/D \rceil$; clearly, any k' boxes satisfy at least one of the constraints. Analyzing the definition of k value, it can be shown that $k' \ge k - |S|$, and that analogs of all above statements concerning $\mathcal{P}$ and k are valid for $\mathcal{P}'$ and k'.

In what follows, we work with box sets of size $k'+2$ only. Some of them are ordered; for an ordered set Z and $X = A, B, C$, function $\#_X(Z)$ is defined as the size of shortest prefix of Z

satisfying constraint X'. Two sets are said to differ by a *swap*, if each one is obtained from the other by a single box removal and a single box insertion.

Algorithm G3 (subroutines are given below):

1. Transform $\mathcal{P}$ to $\mathcal{P}'$; compute S, S', k', and the ordered sets $P'_{AB}, P'_{AC}, P'_{BC}$.
2. Using *Algorithm Chain1*, find either a quasi-optimal solution, or an ordered set T_{AB} which satisfies both currencies A' and B', but does not satisfy B' by its first $k'+1$ boxes. Similarly, find T_{BC}.
3. Using *Algorithm Chain2*, construct a chain of sets, from T_{AB} to T_{BC}, such that any two neighboring sets differ in a swap, and for any set Z, $k'+1 \le \#_B(Z) \le k'+2$. Note that, for any set in this chain, its first k' boxes satisfy at least one of constraints A', C'.
4. Along this chain, either find a set satisfying all three constraints (a quasi-optimal solution), or, if there is none, find two consecutive sets Z_1, Z_2, such that Z_1 does not satisfy constraint C', and Z_2 does not satisfy constraint A'.
5. Using *Algorithm Compose*, combine from Z_1 and Z_2 a (+2)-approximation plan P' for $\mathcal{P}'$.
6. Return plan $P' \cup S$.

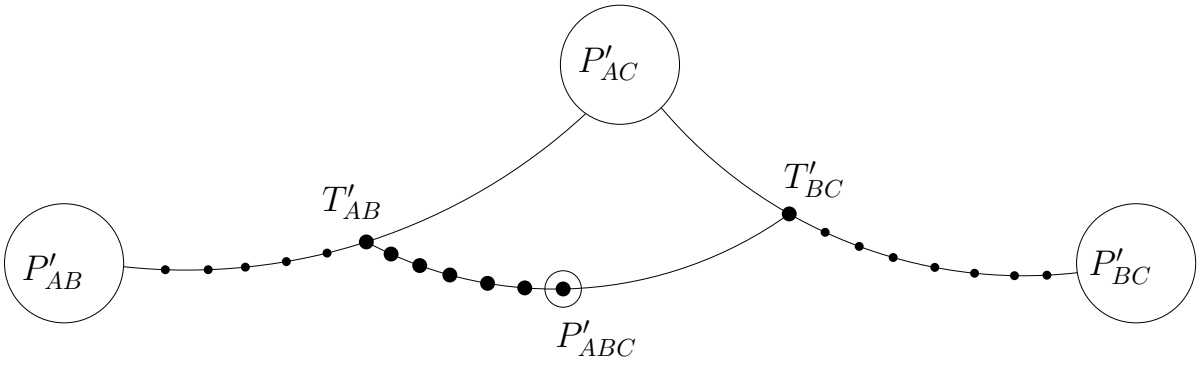

Fig. 3. Scheme of Algorithm G3.

Routine Chain1: Build a chain, i.e., a set of swaps, from P'_{AB} to P'_{AC}, such that constraint A' is satisfied everywhere along it. This can be done by matching arbitrarily $P'_{AB} \setminus P'_{AC}$ with $P'_{AC} \setminus P'_{AB}$, and performing first the swaps increasing A and after that the rest of them. If constraint B is satisfied everywhere at this chain, then P'_{AC} is a quasi-optimal solution. Otherwise, let us detect the first swap, say, from Z_1 to Z_2, such that constraint B is not satisfied by Z_2. We define T_{AB} as the set Z_1 ordered somehow, so that the single box in $Z_1 \setminus Z_2$ is at its last position. Note that $\#_B(T_{AB})$ is equal to $k'+2$.

Routine Chain2: We build a swap chain of ordered sets, from T_{AB} to T_{BC}. Each set will be ordered in the non-decreasing order of currency B amount. Observe that such an (initial) ordering of T_{AB} and T_{BC} cannot spoil their property $\#_B(\cdot) = k'+2$. For any set, Z, in the chain, we will maintain the property $k'+1 \le \#_B(Z) \le k'+2$. Suppose, a prefix of the chain is built, up to set Z, and we need to choose a swap to the next set in the chain. *Rule of swap choice*: If $\#_B(Z) = k+1$ use a B-decreasing swap, which always exist. Else (if $\#_B(Z) = k+2$), if there is a B-increasing

swap, use it. Otherwise, i.e., there are only B-non-increasing swaps, any chain of remaining swaps is suitable.

Routine Compose: We are given Z_1, Z_2 as mentioned in the main algorithm. Suppose that $\#_B(Z_1) = \#_B(Z_2) = k' + 1$. Return the union the first $k' + 1$ boxes of Z_1 and that of Z_2. Let us see that this is the only case. Indeed, otherwise, w.l.o.g., $\#_B(Z_1) = k' + 2$. Notice that Z_2 contains some k' boxes among the first $k' + 1$ first boxes of Z_1. These k' boxes must satisfy constraint A', since in Z_1, they do not satisfy none of constraints B', C'. That is, Z_2 satisfies all the constraints, a contradiction to our assumption.

7 Conclusion and Future work

We suggested new combinatorial algorithms, for the $2D$ and $3D$ cases of the SDB problem. The suggested algorithms improved the best known runtime significantly. Moreover a new combinatorial method was suggested to deal with degenerate input. This method eliminates the need for numerical methods which are hard to use in practice.

Several aspects of the SDB problem remain unsolved; One question would be whether the suggested algorithms may be improved or generalized to higher dimensions. Another open problem has to do with the lower bound. While there is some evidence that the $3D$ case of the SDB problem might be $3Sum - Hard$, the true time complexity of this problem remains unclear.

Acknowledgments. The authors are grateful to R. Ravi and E. Omri for useful comments.

References

1. D. Bertsimas and R. Demir. An approximate dynamic programming approach to multidimensional knapsack problems. *Management Sci* 48(4), 550-565 (2002).
2. R. Cole, J. Salowe, W. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM J. Comput.*, **18**, no. 4 (1989), 792–810.
3. E. A. Dinitz and A. V. Karzanov. Boolean Optimization Problems with Uniform Constraints. Reprint, Institute of Control Sci., Moscow, 1978, 42 p. (in Russian).
4. M. Dyer and A. Frieze. Probabilistic analysis of the multi-dimensional knapsack problem. *Math. of OR* 14, 162-176 (1989).
5. A. Freville and G. Plateau. An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. *Disc. Appl. Math.* 48, 189-212 (1994).
6. M. Fürer and B. Raghavachari. Approximating the Minimum-Degree Steiner Tree to Within One of Optimal. *J. of Algorithms* **17** (1994), 409–423.
7. T. Jordan. On the Optimal Vertex Connectivity Augmentation, *J. of Combinatorial Theory, ser. B*, **63** (1995), 8–20.
8. H. W. Lenstra, Jr. Integer Programming with a fixed number of variables, *Math. of Oper. Res.* **8**, no. 4 (1983), 538–548.
9. S. Martello, P. Toth. Knapsack Problems – Algorithms and Computer Implementations, John Wiley and Sons, Chichester et al., 1990.
10. N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. of ACM.*, **30**, no. 4 (1983), 852–865.
11. B. M. E. Moret. *The Theory of Computation.* Addison-Wesley Publ. Co, Reading, Mass., 1997.
12. V. G. Vizing. On an Estimate of the Chromatic Class of a p-Graph. *Diskret. Analiz.* **3** (1964), 25-30.