

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

Boaz Ben-Moshe\*

Yefim Dinitz†

## 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  $i^{\text{th}}$  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:  
 $j = 1, \dots, m$ ;  $i = 1, \dots, n$ ;  $x_i \in \{0, 1\}$ ;  $\forall_{i,j} : a_{ji} \geq 0$ .

**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 a 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.

\*Department of Computer Science, College of Judea and Samaria, Ariel 44837, Israel, benmo@yosh.ac.il

†Department of Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel. dinitz@cs.bgu.ac.il

## 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  $\alpha$  and  $\beta$  (they are, in fact, the dual variables of the linear relaxation of *SDB*). We say that the **exchange rate**  $\alpha : \beta$  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  $\{1 : 0; 0 : 1 = 1 : \infty; 1 : \frac{a_i - a_j}{b_j - b_i}, 1 \leq i < j \leq n\}$ . (see full version for detailed proof).*

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  $a_i, b_i, \alpha$ , and  $\beta$ .

**Lemma 2 (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 3 (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  $k + 1$  (see full version for the proof).*

Now, let us see how the non-increasing ordering of boxes by value w.r.t. the rate  $1 : \beta$  (henceforth, “ordering at  $\beta$ ”) changes, when  $\beta$  changes from 0 to  $\infty$ . 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  $\beta$  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  $\beta(i, j)$ .

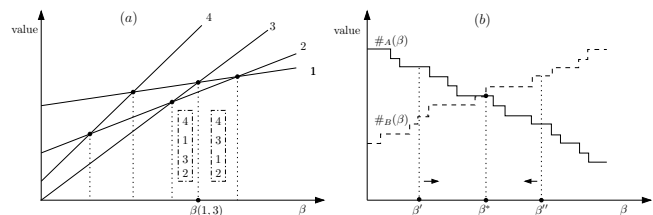


Figure 1: (a): Geometric representation of box values as function of  $\beta$ . (b): Minimal number of best boxes satisfying constraints  $A$  and  $B$ , as a (monotone) function of  $\beta$ .

## 3 Non-Degenerate Two Currencies Case

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  $\beta$ . (ii) test each  $\beta$  for a quasi-optimal solution. The initial idea of [3] was to compute and sort all swap values of  $\beta$ , along with its swapping pair of boxes. After that, beginning from the first  $\beta$  value, and subsequently update it for every  $\beta$ -interval, from the left to right, up to finding the  $\beta^*$  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  $\beta$ ; 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  $\beta$ , can be found in time  $O(n^2)$ .

### 3.1 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  $\beta$ , instead of its linear scanning (see Fig. 1(b)).

**Lemma 4** *For any value of  $\beta$ , 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  $\beta_0$ . 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  $\beta$  exists to the right of  $\beta_0$ , 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.  $\square$

For each analyzed value of  $\beta$ , 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  $r^{\text{th}}$  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 5** *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. (The 2D degenerate case is discussed in the full version).*

#### 4 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  $\beta$ ,  $\gamma$ , 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  $(\beta, \gamma)$ -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  $\gamma$  axis, resulting in the set of *swap values* of  $\gamma$ , dividing the  $\gamma$  axis into the  $\gamma$ -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  $r^{\text{th}}$  one of them, in the increasing order of  $\gamma$ , can be found in  $O(n^2 \log n)$  time.

#### 4.1 Auxiliary two currencies problem

Let us analyze the situation by the sweeping line method. Let us consider the straight line in the  $(\beta, \gamma)$ -plane defined by fixing  $\gamma = \gamma_0$ . We define the *auxiliary  $\gamma_0$ -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  $\beta$ , for the  $\gamma_0$ -problem, by  $\beta^*(\gamma_0)$  (see the definition in Section 3). It is easy to see that any feasible solution of the original problem is a feasible solution for  $\gamma_0$ -problem. The inverse is not guaranteed, but it can be easily shown that any solution feasible for  $\gamma_0$ -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  $\gamma^*$ , 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  $\gamma_0$ -problem is related to the structure of the original three currencies problem. Fixing  $\gamma = \gamma_0$  defines the  $(\beta, v)$  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  $\beta$ , at the  $\gamma_0$ -problem, where the order of boxes  $i$  and  $j$  swaps. These points together break the  $\beta$  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  $(\beta, \gamma)$ -plane. As a consequence, moving the sweep line in the interval between the neighboring swap values of  $\gamma$  does not change the combinatorial structure of the parametrized auxiliary problem: the sequence of  $\beta$ -intervals, together with the box orders in them, remains the same, while the swap values of  $\beta$  move without interchanging, in their order. We denote the values related to the  $\gamma_0$ -problem by  $\cdot(\gamma_0)$ , and the point  $(\beta^*(\gamma_0), \gamma_0)$ , in the  $\beta, \gamma$ -plane, by  $M^*(\gamma_0)$ .

#### 4.2 Binary Search

Let us define our binary search rule as follows. Similarly to the two currencies case, we first analyze values 0 and  $\infty$  of  $\gamma$ , resulting either in a quasi-optimal

solution, or in the  $[0, \infty]$  search area, at the  $\gamma$  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  $\gamma$ . 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  $\infty$ -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  $r^{\text{th}}$  swap value  $\gamma^r$ , we define the  $(\gamma^r -)$ - and  $(\gamma^r +)$ -auxiliary problems. If there is the C/A-situation at the  $(\gamma^r -)$ -problem, we continue with the first half of the search area, before the  $r^{\text{th}}$  swap value; if there is the A/C-situation at the  $(\gamma^r +)$ -problem, we continue with the second half, after this value; if there is the switch from A/C to C/A around  $\gamma^r$ , 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  $\gamma$ , we find a quasi-optimal solution by means of Algorithm *Sandwich*, given below. For an illustration of these two cases, see Figure 2.

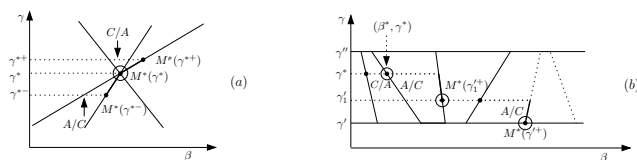


Figure 2: Final phase: (a) 'local' case. (b) 'sandwich' case.

The  $(\gamma^r -)$ -auxiliary problem is, in fact, the  $(\gamma^r - \epsilon)$ -problem, for an infinitesimal  $\epsilon$ . Its definition is combinatorially simulated as follows. Consider the  $\gamma^r$ -problem. The three boxes corresponding to the straight lines intersecting at  $M^*(\gamma^r)$  have some order at the  $\beta$ -interval before  $\beta^*(\gamma^r)$ , which flips to the inverse one at the  $\beta$ -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  $\beta$  direction. All the other components of the  $\gamma^r$ -problem do not change, by the general position assumption. The  $(\gamma^r +)$ -problem is defined similarly.

**Theorem 6** *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.*

## 5 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.