Computing Lower Bounds on Basket Option Prices by Discretizing Semi-infinite Linear Programming

Hyunseok Cho[*], Kyoung-Kuk Kim[†]
Korea Advanced Institute of Science and Technology

Kyungsik Lee[‡]
Seoul National University

July 2015

Abstract

The problem of finding static-arbitrage bounds on basket option prices has received a growing attention in the literature. In this paper, we focus on the lower bound case and propose a novel efficient solution procedure that is based on the separation problem. The computational burden of the proposed method is polynomial in the input data size. We also discuss the case of possibly negative weight vectors which can be applied to spread options.

KEYWORDS: Basket option; Linear programming; Static-arbitrage bound; Separation problem

1 Introduction

Basket option is a type of multi-asset financial derivative whose payoff depends on a linearly weighted sum of multiple underlying asset prices. It has been popular in the derivatives markets as it provides investors with an exposure to, for instance, a basket of stocks or exchange rates so that it can serve as a method of managing portfolio risks in a cost-effective manner. In this paper, we are concerned with the problem of determining basket option prices, in particular, European style basket call option. Although it has a simple payoff function, i.e., $p(S) = (\sum_{i=1}^{n} w_i S_i(T) - K)^+$ where the w_i are fixed weights and the $S_i(T)$ stand for the asset prices at the option maturity T with the option strike K, the pricing of such an option poses theoretical and computational challenges. The primary reasons are that it is difficult to obtain the complete information about the stochastic variables $S_i(T)$'s and, even if we do, it may require a considerable amount of computational resources to get reliable price estimates.

Starting in the mid-2000's, a stream of literature on *model free* bounds for basket option prices has drawn a growing attention from the optimization research community as it was shown that the problem of finding

[*]Industrial and Systems Engineering, E-mail: hyunseokcho@kaist.ac.kr
[†]Corresponding author, Industrial and Systems Engineering, E-mail: kkim3128@kaist.ac.kr, Tel: +82-42-350-3128
[‡]Industrial Engineering, E-mail:optima@snu.ac.kr

upper or lower bounds on basket option prices can be transformed into linear programming problems. More specifically, such an optimization problem is a dual program of $\inf(\sup)_\pi \mathsf{E}_\pi [p(S)]$ where π ranges over the space of multivariate distributions of $(S_1(T), \ldots, S_n(T))$ subject to some available market data such as asset prices $S_i(0)$'s and call option prices for each asset. Finding model free bounds has important practical implications because, first, one can avoid model risk arising from assuming specific dynamics of underlying assets, and second, upper or lower bounds provide us with a method of statically hedging basket options. Furthermore, finding an instance that violates the resulting bounds is directly related to any mispricing in the markets hence for possible arbitrage opportunities. Lastly, since static-arbitrage bounds hold for any parametric model, they can be utilized to check the validity of the model implemented.

In a series of papers [5, 6, 7], the authors study static-arbitrage lower and upper bounds for basket options. Laurence and Wang [7] provide closed-form upper bounds by solving a linear program when forward and single call prices per asset are given. They accomplish the same task for lower bounds but only for $n = 2$. The upper bound is generalized in [6] to the cases where there are infinitely or finitely many available call prices per asset. The lower bound for the case where call prices with a continuum of strikes per asset are known is studied in [5], again for $n = 2$. d'Aspremont and El Ghaoui [2] study Lagrangean relaxations of the problem. And in some particular instances, e.g. when forward and single call per asset are given, the authors' method gives static-arbitrage bounds via linear programming formulations. We note, however, that these approaches described so far are restrictive in terms of the number of assets or constraints.

This naturally leads to the search for more general treatments as proposed in the works of [9, 10, 11]. Peña et al. [11] contribute to the existing literature by considering bid-ask prices of call options and further strengthening the upper bound case. The problem size of their linear programming is linear in the input data size, hence efficient computations are possible. On the other hand, the lower bound case is dealt in [10] where the authors introduce additional variables to transform the original problem into a new linear program. However, as the authors put it as being "fundamentally different from that of the super-replicating portfolios" [11], the lower bound case raises substantially difficult computational challenges because of the exponentially increasing problem size. Peña et al. [9] suggest a nonlinear programming approach via Dantzig-Wolfe decomposition to treat the upper and lower cases together. The approach, however, is based on the assumption that the support of the asset distributions is an n-dimensional compact box.

In this paper, we propose a method of finding static-arbitrage lower bounds on basket options prices when arbitrary number of calls or forward per asset, say m, are given. Especially, we aim to improve upon [10] so that efficient computations of larger problem instances are feasible. In spirit, our approach is close to that of [3] where the authors propose a feasible discretization method for the problem of finding an upper bound. In what follows, we show the semi-infinite linear programming for the lower bound problem with possibly negative weights is also equivalent to a linear programming problem, albeit with an exponential number of constraints (and polynomially many variables). More importantly, we design an algorithm to efficiently solve such a linear program by considering the associated separation problem, which can be solved in polynomially many steps. This result has a practical importance because the lower bound problem has been confined to linear programs of manageable size only, due to computational difficulties.

The linear programming formulation for the lower bound case with positive weight vectors is presented in Section 2.1, and Section 2.2 describes our suggested algorithm. The more general cases with possibly negative weight vectors are studied in Section 2.3. We check the performance of the method with extensive

numerical tests and the results are reported in Section 3. We conclude the paper with some final remarks in Section 4.

2 Solution Approach

2.1 Discretizing semi-infinite linear programming

Recall that the payoff function of European basket call option is given by $(w \cdot S(T) - K)^+$ where $w = (w_1, \ldots, w_n)$ is a weight vector, $S(T) = (S_1(T), \ldots, S_n(T))$ is a random vector representing underlying asset prices at maturity T, and K is an option strike pre-specified in a contract. Throughout this paper, we impose the following assumptions.

Assumption 1. There are m number of European call options with maturity T and strikes $K_{i1} \leq K_{i2} \leq \cdots \leq K_{im}$ for each stock S_i whose market prices are given as c_{ij} where $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

Assumption 2. The weights w_i are positive integers and strikes K_{ij} and K are nonnegative integers. In addition, their values increase at most polynomially as n, m increase.

Assumption 3. All contracts are tradable without any friction. Also, the risk-free rate is zero.

In the first assumption, we can actually utilize put option prices or forward prices instead of call options; in the former, we use the put-call parity and in the latter, we set $K_{ij} = 0$. We also see that the number of available call prices could be different across stocks, but we can repeat the same strikes multiple times so that **Assumption 1** still holds. As for the second assumption, we notice that typical calls or puts in the market have integer strikes. If weights or strikes are rational numbers, then we can scale those values so that **Assumption 2** holds. The risk-free rate is assumed to be zero without any loss of generality as, e.g., we can work under the forward measure. Market frictions in **Assumption 3** refer to any restriction on option trading such as transaction costs or short-sale constraint for instance. We shall discuss the relaxation of this latter assumption in a later part of the paper.

Under these assumptions, the model-free lower bound of the basket call is obtained by solving the following optimization problem [2]:

$$\begin{aligned} \inf_{\pi \in \Pi} \quad & \mathsf{E}_\pi \left[(w \cdot S(T) - K)^+ \right] \\ \text{s.t.} \quad & \mathsf{E}_\pi \left[(S_i(T) - K_{ij})^+ \right] = c_{ij}, \; i = 1, ..., n, \quad j = 1, ...m. \end{aligned} \tag{1}$$

Here, Π is the set of all distributions on $\mathbb{R}^n_+$ and E_π is the expectation where $S(T)$ has the probability distribution π. The dual problem of (1) is given in [10] as follows:

$$\begin{aligned} \sup_{z \in \mathbb{R}, y \in \mathbb{R}^{n \times m}} \quad & z + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} y_{ij} \\ \text{s.t.} \quad & z + \sum_{i=1}^{n} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij} \leq (w \cdot s - K)^+, \quad \forall s = (s_1, \ldots, s_n) \in \mathbb{R}^n_+. \end{aligned} \tag{2}$$

Peña et al. [10] discuss sufficient conditions for strong duality between (1) and (2) to hold. One condition that guarantees the strong duality is that there is at least one call option price for each stock. This problem

can be interpreted as finding the best possible sub-replicating portfolio using available call options. The authors further transform the above semi-infinite linear program into a linear program. However, they note that the linear program is of an enormous size in full generality, and they instead focus on special cases in which linear programs of solvable sizes are treated.

We, alternatively, take a different approach to (2) by noting that it is enough to check the constraints for each $s \in \mathfrak{C}_0$ not for all $s \in \mathbb{R}^n_+$ where

$$\mathfrak{C}_0 = \mathfrak{C}_1 \cup \mathfrak{C}_2,$$
$$\mathfrak{C}_1 = \left\{ s \in \mathbb{R}^n_+ \middle| \text{ for each } i, s_i = 0 \text{ or } K_{ij} \text{ for some } j \right\}, \quad \mathfrak{C}_2 = \bigcup_{k=1}^{n} \mathfrak{B}_k,$$
$$\mathfrak{B}_k = \left\{ s \in \mathbb{R}^n_+ \middle| \text{ for each } i \neq k, s_i = 0 \text{ or } K_{ij} \text{ for some } j, s_k = w_k^{-1} \left(K - \sum_{i=1, i\neq k}^{n} w_i s_i \right) \geq 0 \right\}.$$

Here, the sets $\mathfrak{B}_k$ represent the points at which the hyperplane defined by the basket option payoff intersects with the rectangular grid given by the vanilla option strikes. From now on, we define $K_{i0} = 0$ for each i for notational convenience. Then, the resulting linear program can be written as follows:

$$\begin{aligned} \max_{z \in \mathbb{R}, y \in \mathbb{R}^{n \times m}} \quad & z + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} y_{ij} \\ \text{s.t.} \quad & z + \sum_{i=1}^{n} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij} \leq (w \cdot s - K)^+, \quad \forall s \in \mathfrak{C}_0, \\ & \sum_{j=1}^{m} y_{ij} \leq w_i, \quad \forall i = 1, \ldots, n. \end{aligned} \tag{3}$$

The next proposition shows that the above two formulations are indeed equivalent.

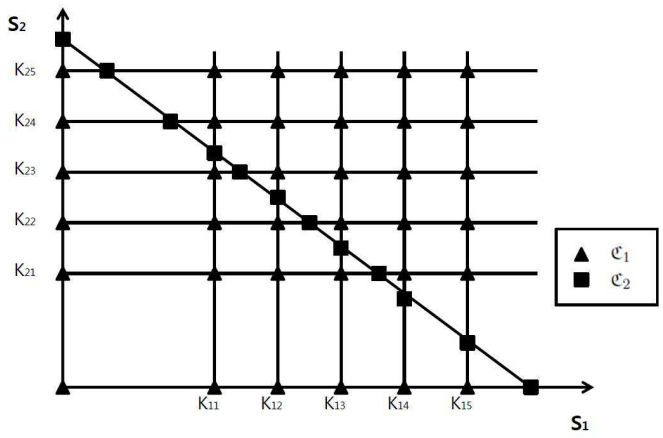


Figure 1: Dividing the first quadrant by hyperplanes $\{s \in \mathbb{R}^2_+ | s_i = K_{ij}\}$ and $\{s \in \mathbb{R}^2_+ | w \cdot s = K\}$.

Proposition 1 *The semi-infinite program* (2) *and the linear program* (3) *are equivalent.*

Proof: Suppose (z, y) satisfies the constraints in (2) for all s. Thus, trivially (z, y) satisfies the constraints in (3) for all $s \in \mathfrak{C}_1 \cup \mathfrak{C}_2$. Also, by sending s_i to infinity and fixing all others, we easily see that $\sum_{j=1}^{m} y_{ij} \leq w_i$.

To prove the converse, let us start from the observation that $\mathbb{R}^n_+$ can be divided into bounded and unbounded regions whose interiors are mutually exclusive. In particular, we divide $\mathbb{R}^n_+$ using hyperplanes of form $\{s \in \mathbb{R}^n_+ | s_i = K_{ij}\}$ and $\{s \in \mathbb{R}^n_+ | w \cdot s = K\}$. See Figure 1 for an illustration. Then, the function $(w \cdot s - K)^+ - z - \sum_{i,j}(s_i - K_{ij})^+ y_{ij}$ becomes linear in each of the bounded regions. Since a linear function assumes its maximum at one of extreme points, it is enough to check the nonnegativity of this function at extreme points of each bounded region. However, it is easy to see that such extreme points are elements of $\mathfrak{C}_1 \cup \mathfrak{C}_2$.

We next turn our attention to the unbounded regions. Any such region can be expressed as

$$\mathfrak{D}_{\mathcal{I}} = \left\{ s \in \mathbb{R}^n_+ | w \cdot s \geq K, \;\; s_i \geq \max_{j=1,\ldots,m} K_{ij} \;\; \forall i \in \mathcal{I} \text{ and } s_i \in [\underline{K}_i, \overline{K}_i] \;\; \forall i \in \mathcal{I}^c \right\}$$

where $\mathcal{I}$ is some nonempty subset of $\{1, \ldots, n\}$ and $\overline{K}_i \geq \underline{K}_i$ are two of the strike values for asset i. We need to show that $(w \cdot s - K)^+ - z - \sum_{i,j}(s_i - K_{ij})^+ y_{ij}$ is nonnegative for all $s \in \mathfrak{D}_{\mathcal{I}}$ given that the constraints in (3) are satisfied. Indeed, we observe that, for $s \in \mathfrak{D}_{\mathcal{I}}$,

$$\begin{aligned}
&(w \cdot s - K)^+ - z - \sum_{i,j}(s_i - K_{ij})^+ y_{ij} \\
&= \; w \cdot s - K - z - \sum_{i \in \mathcal{I}} \sum_j (s_i - K_{ij}) y_{ij} - \sum_{i \in \mathcal{I}^c} \sum_j (s_i - K_{ij})^+ y_{ij} \\
&= \; \sum_{i \in \mathcal{I}} s_i \left\{ w_i - \sum_j y_{ij} \right\} + (\text{remaining terms}).
\end{aligned}$$

Then, the last expression is minimized when $s_i = \max_j K_{ij}$ for each $i \in \mathcal{I}$ because of the constraint $\sum_j y_{ij} \leq w_i$. The remaining terms are linear in variables $s_i \in \mathcal{I}^c$ which are in bounded sets. Consequently, the minimum is obtained at an extreme point, therefore, the feasibility of (3) implies that of (2). ■

It should be noted that the linear program of [10] has variables and constraints of exponential size in n whereas (3) has $nm + 1$ variables. However, the still large size of the set of constraints, $\mathcal{O}((m+1)^{n-1}(m+n))$, puts a heavy burden on the solvability of the original problem. This is a stark contrast with Theorem 3.3 of [3] where the number of constraints in the linear program is $O(nm)$, the reason being that the idea behind the authors' result does not apply for the inequality of (3). In the next section, we instead tackle the associated *separation problem* first and devise an algorithm for the original problem that utilizes our solution to the separation problem.

2.2 Solving the separation problem

The separation problem is the problem of checking if for given (z, y) all the constraints are satisfied or, if not, finding a violated constraint. We present the separation problem associated with (3) as, for given (z, y),

$$\text{check if} \qquad z + \sum_{i=1}^{n} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij} \leq (w \cdot s - K)^+, \quad \forall s \in \mathfrak{C}_0, \tag{4}$$

or find $s^* \in \mathfrak{C}_0$ such that the inequality is violated.

Note that $\sum_j y_{ij} \leq w_i$ does not appear explicitly above. However, we can easily check these n constraints for (z, y) before we solve the separation problem.

The key step in our solution approach is to compute the maximum of the left-hand side of (4) for $s \in \mathfrak{C}_0$. Such a maximum turns out to depend on $x \in w \cdot s$ and we construct auxiliary functions $f_i(x)$ and $g_i(x)$ as we run through $i = 1, \ldots, n$. For the description of the algorithm, let us denote a lower bound of the left hand side of (4) by $l(z, y)$. For example, one can take simply $l(z, y) = -\infty$ or

$$l(z, y) = z + \sum_{(i,j):y_{ij}<0} (\bar{s}_i - K_{ij})\, y_{ij} - 1$$

where $\bar{s}_i = \max\{s_i | s \in \mathfrak{C}_0\}$.

Proposition 2 *Suppose that Assumptions 1 to 3 hold. Then, the separation problem* (4) *can be solved in* $O(nm^2(M + nK))$ *operations where* $M = \sum_{i=1}^n w_i K_{im}$.

Proof: Let $l(z, y)$ be a sufficiently small or a large negative number. In the first step, we maximize the left hand side of (4) over $\mathfrak{C}_1$.

Define functions f_1, g_1 on $\mathcal{D} := \{0, 1, \ldots, M\}$ by

$$f_1(x) = \begin{cases} z + \sum_{j=1}^{m} (K_{1h} - K_{1j})^+ y_{1j}; \text{ if } x = w_1 K_{1h}, \\ l(z, y); \text{ otherwise,} \end{cases} \quad g_1(x) = \begin{cases} h; \text{ if } x = w_1 K_{1h}, \\ 0; \text{ otherwise,} \end{cases}$$

where h ranges over $\{0, 1, \ldots, m\}$. The total number of these operations is $O(m^2 M)$. Next, we define f_2 on $\mathcal{D}$ by

$$f_2(x) = \begin{cases} \max_{h \in \mathcal{D}_2(x)} \left[f_1(x - w_2 K_{2h}) + \sum_{j=1}^{m} (K_{2h} - K_{2j})^+ y_{2j} \right]; \text{ if } \mathcal{D}_2(x) \neq \emptyset, \\ l(z, y); \text{ otherwise.} \end{cases}$$

Here, the set $\mathcal{D}_2(x)$ is nothing but the set of indices j_2 such that x can be represented by $w_1 K_{1j_1} + w_2 K_{2j_2} = x$ for some j_1 and j_2. Or equivalently, $\mathcal{D}_2(x) = \{h \in \{0, \ldots, m\} | x' = x - w_2 K_{2h} \geq 0, f_1(x') > l(z, y)\}$. The function g_2 is then defined as one of the maximizers in the definition of f_2. These computations can be done in a sequential manner with $O(m^2 M)$ operations. See `step (2)` in `Separation_Problem`. Consequently, the image $\{f_2(x)\}$ is the set of vectors which record the maximum values of

$$z + \sum_{j=1}^{m} (s_1 - K_{1j})^+ y_{1j} + \sum_{j=1}^{m} (s_2 - K_{2j})^+ y_{2j}$$

among all (s_1, s_2) such that $x = w_1 s_1 + w_2 s_2$ for some strike values s_1, s_2 in the input data.

By repeating the same procedure until we arrive at $f_n : \mathcal{D} \to \mathbb{R}$, we observe that $f_n(x)$ for $x \in \mathcal{D}$ is equal to the maximum of the left hand side of (4) among all $s \in \mathfrak{C}_1$ such that $x = w \cdot s$; if there is no such s, then $f_n(x) = l(z, y)$. Also, $(g_1(x), \ldots, g_n(x))$ gives us the corresponding indices of strikes in the input

data. Then, by comparing $\{f_n(x)\}$ with $\{(x-K)^+\}$, we can find (a subset of) violating constraints over $\mathfrak{C}_1$ in $O(nm^2M)$ operations.

In the second step, we consider the set $\mathfrak{C}_2$. The procedure is essentially the same except that now we have a fixed budget $w \cdot s = K$. To be more specific, let us consider $\mathfrak{B}_n$ in which $s_1, \ldots, s_{n-1}$ take on strike values whereas s_n is determined by $w_n s_n = K - \sum_{j=1}^{n-1} w_i s_i$. We set $\mathcal{D} = \{0, 1, \ldots, K\}$ and proceed as above until we arrive at $f_{n-1} : \mathcal{D} \to \mathbb{R}$. Then, each $f_{n-1}(x)$ records the maximum value of

$$z + \sum_{i=1}^{n-1} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij}$$

whenever $x = \sum_{i=1}^{n-1} w_i s_i \leq K$. As in the first step, $(g_1(x), \ldots, g_{n-1}(x))$ yields the corresponding indices. Lastly, we compute

$$\tilde{f}(x) = f_{n-1}(x) + \sum_{j=1}^{m} \left(\frac{K-x}{w_n} - K_{nj} \right)^+ y_{nj}, \quad x \in \mathcal{D}.$$

By comparing $\{\tilde{f}(x)\}$ with 0, we can find (a subset of) violating constraints over $\mathfrak{B}_n$. We note that this takes $O(nm^2K)$ operations.

To complete the second step, we repeat the same procedure for $\mathfrak{B}_1, \ldots, \mathfrak{B}_{n-1}$. Therefore, we conclude that the separation problem can be solved in total $O(nm^2(M + nK))$ operations. ■

One can reduce the complexity of the above procedure at the expense of computational resources to get extra information. Let us denote the g.c.d. of $\{w_i s_i \forall i \text{ and } \forall s \in \mathfrak{C}_1\}$ by d. Then, every computation only needs to be done in the lattice $\mathcal{J} := \{0, d, 2d, \ldots, M\}$, reducing the total operations by a factor of d. This is practically relevant in a situation where strikes are given by multiples of a certain constant.

We write down one version of the algorithm that solves the separation problem below. The pseudocode returns s that violates (4) the most over each of the sets $\mathfrak{C}_1, \mathfrak{B}_1, \ldots, \mathfrak{B}_n$. For notational simplicity, we use $\mathrm{ind}(s_i) = j$ if $s_i = K_{ij}$.

Algorithm Separation_Problem

(1) for each $x \in \mathcal{J}$ {
  if $x = w_1 K_{1h}$ for some $h \in \{0, \ldots, m\}$,
  then set $f_1(x) = z + \sum_{j=1}^{m} (K_{1h} - K_{1j})^+ y_{1j}$ and $g_1(x) = h$;
  for all others, set $f_1(x) = l(z, y)$ and $g_1(x) = 0$;
}
(2) for $i = 2$ to n {
  (2-1) initialize $f_i(x) = l(z, y)$ and $g_i(x) = 0$ for all $x \in \mathcal{J}$;
  (2-2) for each $x \in \mathcal{J}$ and for $h = 0$ to m {
    if $f_{i-1}(x - w_i K_{ih}) > l(z, y)$ & $\xi := f_{i-1}(x - w_i K_{ih}) + \sum_{j=1}^{m} (K_{ih} - K_{ij})^+ y_{ij} \geq f_i(x)$,
    then set $f_i(x) = \xi$ and $g_i(x) = h$;
  }
}
(3) if $x^* = \mathrm{argmax}\{f_n(x) - (x-K)^+\}$ and $f_n(x^*) > (x^* - K)^+$,

then return *violated* with s such that $\mathrm{ind}(s_i) = g_i(x^*)$ for $i = 1, \ldots, n$;
(4) for $p = n$ to 1 {
do steps (1) and (2) with $i \in \mathcal{I} = \{1, 2, \ldots, p-1, p+1, \ldots, n\}$ for $x \leq K$;
compute $\tilde{f}(x) = f_{\max \mathcal{I}}(x) + \sum_{j=1}^{m} ((K - x)/w_p - K_{pj})^+ y_{pj}$ for each x;
if $x^* = \mathrm{argmax}\{\tilde{f}(x)\}$ and $\tilde{f}(x^*) > 0$,
then return *violated* with s such that $\mathrm{ind}(s_i) = g_i(x^*), i \in \mathcal{I}$ & $s_p = (K - x^*)/w_p$;
}

The proposed algorithm above has an implication for the lower bound problem thanks to the polynomial equivalence between a linear program and the associated separation problem. Such equivalence is well known to hold for (possibly unbounded) rational polyhedra as long as the coefficients of constraints do not grow excessively large in their magnitudes. We refer the interested reader to, e.g., Theorem 14.1 and its corollaries of [12]. Such a moderate growth condition for (3) can be shown to hold due to **Assumption 2** and the definitions of $\mathfrak{C}_1$ and $\mathfrak{C}_2$. As a consequence, we are led to the following conclusion.

Corollary 3 *Suppose that Assumptions 1 to 3 hold. Then, the linear program* (3) *can be solved in the number of operations that is polynomial in* n *and* m.

Although such polynomially many operations can be implemented by ellipsoid methods as explained in [12], it was realized by the research community that "theoretically efficient" ellipsoid algorithms are not necessarily empirically efficient, as pointed out in Chapter I.6 of [8] and Chapter 3 of [4]. In order to make the newly developed algorithm practically useful, we adopt one widely used approach as in [3]. That is, we start the algorithm with some initial set of constraints, say $\mathfrak{C}$, to obtain an optimal solution, say $(z^{(0)}, y^{(0)})$, and then solve the separation problem in order to add violated constraints to $\mathfrak{C}$. Then, we re-solve the modified linear program to obtain the next solution $(z^{(1)}, y^{(1)})$, and so on. See the algorithm `Lower_Bound` below. The computational advantage we take from this approach may depend on $\mathfrak{C}$. In Section 3, we present some numerical results as we vary the initial set of constraints.

Algorithm Lower_Bound

initialize $\mathfrak{C}$ and set *flag*$= 0$;
do {
solve (3) with $\mathfrak{C}$ to obtain (z^*, y^*) and the optimum c^*;
solve *separation problem* for (z^*, y^*) and find violated constraints $\mathfrak{C}'$;
if $\mathfrak{C}' = \emptyset$, then *flag*$= 1$ and return c^*;
else update $\mathfrak{C}$ by $\mathfrak{C} \cup \mathfrak{C}'$;
} while (*flag*$= 0$)

We note that, slightly different from the discretization method in [3], our algorithm adds a set of constraints $\mathfrak{C}'$ because `Separation_Problem` finds the most violating constraints from each of $\mathfrak{C}_1$, $\mathfrak{B}_1$, ..., $\mathfrak{B}_n$. We also note that the algorithm runs until there is no violating constraint. This implies that, in theory, the algorithm might run exponentially many iterations. It can be avoided, to some extent, by setting up

some bounds on the number of iterations or taking an ϵ-optimality criterion. However, for simplicity, we implement the algorithm as described above and report numerical outcomes later. It is quite interesting to see that the resulting number of iterations is reasonably small even in the case of large baskets such as 8 or 10 assets. The numerical tests below thus support the practical efficiency of `Lower_Bound`.

2.3 Extension to negative weights

The idea implemented in the previous subsections can be extended to the cases where the basket weights are possibly negative. Since $w < 0$ makes the problem trivial, we assume that there are two disjoint and nonempty sets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $w_{i_1} > 0$ for all $i_1 \in \mathcal{I}_1$, $w_{i_2} < 0$ for all $i_2 \in \mathcal{I}_2$, and $\mathcal{I}_1 \cup \mathcal{I}_2 = \{1, 2, \ldots, n\}$.

The most prominent example of such contracts is a spread option, which typically is defined as an option on the difference between two asset prices. Other interesting examples are introduced in [1]. For instance, a 3:2:1 crack spread option involves asset prices associated with the prices of unleaded gasoline, heating oil, and crude oil with weights $(\frac{2}{3}, \frac{1}{3}, -1)$. As done in Section 2.1, we re-formulate the semi-infinite linear program (2). One naturally arising set of constraints is $\sum_{j=1}^{m} y_{ij} \le w_i^+$ for all $i = 1, 2, \ldots, n$ (see also [3]). However, unlike the upper bound counterpart, those are not enough to guarantee the equivalence between (2) and (3) due to the peculiarity of the lower bound problem.

Proposition 4 *Suppose that $w_{\mathcal{I}_1} > 0$ and $w_{\mathcal{I}_2} < 0$ for two disjoint sets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $\mathcal{I}_1 \cup \mathcal{I}_2 = \{1, 2, \ldots, n\}$. Then, the semi-infinite program* (2) *is equivalent to the following linear program:*

$$\begin{aligned} \max_{z \in \mathbb{R}, y \in \mathbb{R}^{n \times m}} \quad & z + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} y_{ij} \\ \text{s.t.} \quad & z + \sum_{i=1}^{n} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij} \le (w \cdot s - K)^+, \quad \forall s \in \mathfrak{C}_0, \\ & \sum_{j=1}^{m} y_{ij} \le w_i^+, \quad \forall i = 1, \ldots, n, \\ & w_{i_1} \sum_{j=1}^{m} y_{i_2 j} \le w_{i_2} \sum_{j=1}^{m} y_{i_1 j}, \quad \forall i_1 \in \mathcal{I}_1, \ \forall i_2 \in \mathcal{I}_2. \end{aligned} \tag{5}$$

Proof: We first argue that a given vector (z, y) satisfies the constraint

$$z + \sum_{i=1}^{n} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij} \le (w \cdot s - K)^+, \quad \forall s = (s_1, \ldots, s_n) \in \mathbb{R}_+^n$$

if and only if (z, y) satisfies the following set of constraints

$$\begin{aligned} & z + \sum_{i=1}^{n} \sum_{j=1}^{m} (s_i - K_{ij})^+ y_{ij} \le (w \cdot s - K)^+, \quad \forall s \in \mathfrak{C}_0, \\ & \lambda \cdot \eta \le (\lambda \cdot w)^+, \quad \forall \lambda \in \mathbb{R}_+^n \end{aligned}$$

where $\eta_i = \sum_{j=1}^m y_{ij}$ for $i = 1, 2, \ldots, n$.

For the necessity, fix $\lambda \in \mathbb{R}^n_+$ and set $s = p\lambda$. By dividing the inequality by p and sending p to infinity, we obtain $\lambda \cdot \eta \leq (\lambda \cdot w)^+$.

For the sufficiency, consider an unbounded region, one of the regions that can be constructed as in Figure 1. Suppose that $\tilde{\mathcal{I}}$ is the set of indices such that s_i is unbounded for each $i \in \tilde{\mathcal{I}}$. Then, we observe that

$$\begin{aligned} & z + \sum_{i=1}^n \sum_{j=1}^m (s_i - K_{ij})^+ y_{ij} - (w \cdot s - K)^+ \\ = \quad & (\text{ linear and bounded }) + \sum_{i \in \tilde{\mathcal{I}}} s_i \eta_i - (w \cdot s - K)^+. \end{aligned}$$

In such a region, $(w \cdot s - K)^+$ is also linear and thus the above function either obtains its maximum at its extreme points or is unbounded. However, the second constraint of the sufficient conditions implies

$$\sum_{i \in \tilde{\mathcal{I}}} s_i \eta_i - (w \cdot s - K)^+ \leq (s_{\tilde{\mathcal{I}}} \cdot w_{\tilde{\mathcal{I}}})^+ - (s_{\tilde{\mathcal{I}}} \cdot w_{\tilde{\mathcal{I}}} + s_{\tilde{\mathcal{I}}^c} \cdot w_{\tilde{\mathcal{I}}^c} - K)^+,$$

which is clearly bounded. Here, $s_{\tilde{\mathcal{I}}}$ denotes the vector of s_i's for $i \in \tilde{\mathcal{I}}$. Other symbols are similarly defined. Therefore, the sufficiency follows.

The rest of the proof becomes complete thanks to the lemma below. ■

Lemma 5 *Suppose that $w_{\mathcal{I}_1} > 0$ and $w_{\mathcal{I}_2} < 0$ for two disjoint sets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $\mathcal{I}_1 \cup \mathcal{I}_2 = \{1, 2, \ldots, n\}$. Then, an n-dimensional vector η satisfies*

$$\lambda \cdot \eta \leq (\lambda \cdot w)^+, \quad \forall \lambda \in \mathbb{R}^n_+$$

if and only if $\eta \leq w^+$ and

$$w_{i_1} \eta_{i_2} \leq w_{i_2} \eta_{i_1}, \quad \forall i_1 \in \mathcal{I}_1, \ \forall i_2 \in \mathcal{I}_2.$$

Proof: ($\Rightarrow$) The first part is clear by considering the canonical basis vectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ in $\mathbb{R}^n_+$. More specifically, note that, from the given condition, we have for $a, b > 0$

$$\lambda \cdot \eta = a\eta_{i_1} + b\eta_{i_2} \leq (aw_{i_1} + bw_{i_2})^+$$

where $\lambda = a\mathbf{e}_{i_1} + b\mathbf{e}_{i_2}$. With $a = -bw_{i_2}/w_{i_1}$, it is easy to see that

$$w_{i_2} \eta_{i_1} \geq w_{i_1} \eta_{i_2}.$$

We note that this holds for every pair (i_1, i_2) with $w_{i_1} > 0$ and $w_{i_2} < 0$.

($\Leftarrow$) If $\lambda_{\mathcal{I}_1} = 0$ for $\lambda \in \mathbb{R}^n_+$, then the desired inequality reduces to $\lambda_{\mathcal{I}_2} \cdot \eta_{\mathcal{I}_2} \leq 0$ which is implied by $\eta_{\mathcal{I}_2} \leq 0$; this follows from $\eta_{\mathcal{I}_2} \leq w^+_{\mathcal{I}_2} = 0$. If $\lambda_{\mathcal{I}_2} = 0$ for $\lambda \in \mathbb{R}^n_+$, then we need to show $\lambda_{\mathcal{I}_1} \cdot \eta_{\mathcal{I}_1} \leq \lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1}$, but again this follows from $\eta_{\mathcal{I}_1} \leq w^+_{\mathcal{I}_1} = w_{\mathcal{I}_1}$.

Hence, for the rest of the proof, we assume $\lambda_{\mathcal{I}_1} \neq 0$ and $\lambda_{\mathcal{I}_2} \neq 0$. Fix such $\lambda \in \mathbb{R}^n_+$. From the given conditions, we see that

$$\lambda_{i_1} \eta_{i_1} w_{i_2} \lambda_{i_2} \geq \lambda_{i_1} w_{i_1} \eta_{i_2} \lambda_{i_2}$$

for all pairs (i_1, i_2) with $w_{i_1} > 0$ and $w_{i_2} < 0$. By summing over $\mathcal{I}_1$ and $\mathcal{I}_2$, we obtain

$$(\lambda_{\mathcal{I}_1} \cdot \eta_{\mathcal{I}_1})(\lambda_{\mathcal{I}_2} \cdot w_{\mathcal{I}_2}) \geq (\lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1})(\lambda_{\mathcal{I}_2} \cdot \eta_{\mathcal{I}_2}). \tag{6}$$

If we write $\lambda_{\mathcal{I}_2} \cdot w_{\mathcal{I}_2} = \lambda \cdot w - \lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1}$, then the above inequality leads us to

$$(\lambda_{\mathcal{I}_1} \cdot \eta_{\mathcal{I}_1})(\lambda \cdot w) \geq (\lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1})(\lambda \cdot \eta); \tag{7}$$

whereas with the expression $\lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1} = \lambda \cdot w - \lambda_{\mathcal{I}_2} \cdot w_{\mathcal{I}_2}$ applied to the right hand side of (6) we obtain

$$(\lambda \cdot \eta)(\lambda_{\mathcal{I}_2} \cdot w_{\mathcal{I}_2}) \geq (\lambda_{\mathcal{I}_2} \cdot \eta_{\mathcal{I}_2})(\lambda \cdot w). \tag{8}$$

Suppose $\lambda \cdot w \geq 0$. Then, Eq. (7) together with $\lambda_{\mathcal{I}_1} \cdot \eta_{\mathcal{I}_1} \leq \lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1}$ and $0 < \lambda_{\mathcal{I}_1} \cdot w_{\mathcal{I}_1}$ leads to $\lambda \cdot \eta \leq \lambda \cdot w$. Now suppose $\lambda \cdot w \leq 0$. We note that

$$\begin{aligned} \lambda_{\mathcal{I}_2} \cdot w_{\mathcal{I}_2} &< 0, \\ \lambda_{\mathcal{I}_2} \cdot \eta_{\mathcal{I}_2} &\leq 0. \end{aligned}$$

Consequently from Eq. (8), we get $\lambda \cdot \eta \leq 0$. ■

It is not difficult to modify the proposed `Separation_Problem` to the case of general weight vectors. We need to change $\mathcal{D}$ in the proof of Proposition 2 to the set $\{-M_2, -M_2 + 1, \ldots, M_1\}$ where $M_1 = \sum_{i \in \mathcal{I}_1} w_i K_{im}$ and $M_2 = -\sum_{i \in \mathcal{I}_2} w_i K_{im}$; however, the cental idea of the algorithm is unchanged.

Before we move onto numerical experiments, other possible extensions merit some comments. As noted at the beginning of this section, there are possible market frictions that may become of practical concerns in implementing (sub-)replicating strategies. In the case of bid-ask spreads, we can adopt the approach of [11] by introducing additional variables such that $y_{ij} = y_{ij}^+ - y_{ij}^-$ and $y_{ij}^{\pm} \in \mathbb{R}_+$. Those variables are associated with ask prices p_{ij}^+ and bid prices p_{ij}^-, respectively, instead of a single p_{ij}. If there is a no-short restriction on trading, then it can be easily incorporated by adding the constraints $y_{ij} \geq 0$. Lastly, we notice that there are other instruments that contain extra distributional and correlation information; power options on a single asset as dealt in [3] or call-on-max options on multiple assets in [13]. This interesting extension, however, remains a topic to be explored in the future.

3 Numerical Results

Experiments. To test the performance of the proposed algorithm, we conduct numerical experiments by varying the sizes of the basket and available call prices. In more detail, we consider baskets consisting of 3, 4, 5, 6, 8, and 10 assets with the number of calls per asset up to as many as hundreds. The market prices of those calls, in our experiments, are calculated based on the Black-Scholes formula. For this, initial stock prices are set equal to 100 in all of the test cases, and stock volatilities are as in Table 1. The table also records the weight vectors for the baskets considered. For option strikes of the calls for each asset, we start from 100 and add more calls with step size ± 1 as m increases.

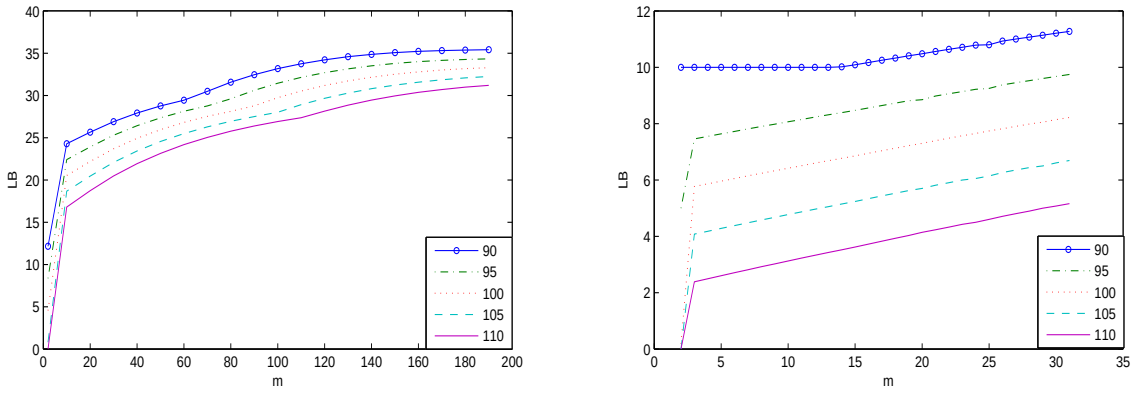
The algorithm is dependent on the user's choice of the initial set of constraints, $\mathfrak{C}$. We consider $\mathfrak{C}_0$, $\mathfrak{C}_1$, and $\mathfrak{C}_2$ from the previous section. In addition, we set $\mathfrak{C}_3$ by

$$\mathfrak{C}_3 = \Big\{ (s_1, \ldots, s_n) \Big| \text{ for some } i, s_i = K_{i0} \text{ or } K_{im}, \text{ and for some } p, s_j = K_{jp} \; \forall \, j \neq i \Big\}.$$

Table 1: Volatilities and weight vectors in the numerical tests.

n	values									
3	σ	1.0	1.6	2.0						
	w	0.3	0.35	0.35						
4	σ	0.3	0.3	1.8	1.2					
	w	0.1	0.2	0.3	0.4					
5	σ	0.3	0.4	0.8	1.8	1.9				
	w	0.2	0.2	0.2	0.2	0.2				
6	σ	0.3	0.5	1.3	1.5	1.9	2.1			
	w	0.1	0.1	0.1	0.1	0.3	0.3			
8	σ	0.1	0.2	1.5	1.6	1.7	1.8	1.9	2.0	
	w	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.3	
10	σ	0.02	0.05	0.1	0.13	0.15	0.2	0.23	0.25	0.29	0.35
	w	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

The last set is particularly useful because its size is $O(mn)$ whereas all other sets are exponential in n. It is seen from our numerical tests that this significantly increases the problem size that is solvable via linear programming approach. Lastly, the sets of constraints $\mathfrak{C}_i$ are augmented by n constraints $\sum_{j=1}^{m} y_{ij} \leq w_i$ for all i as given in (3). The numerical test is implemented on a personal computer with Intel i3-3220 CPU @ 3.30GHz and 4GB RAM, and the CPLEX solver is used.



(a) 3 assets (b) 5 assets

Figure 2: Lower bounds for varying m with option strikes $\{90, 95, 100, 105, 110\}$.

Results. Figure 2 illustrates how lower bounds improve as the number of calls per asset m increases. Although we show only two cases of $n = 3$ and $n = 5$, similar behaviors are observed for other instances. One can see that overall there is an increasing tendency of the lower bounds for larger m values; however, there are also regions of m in which some curves are flat. For example, see the case of $K = 90$ and $n = 5$ where a larger m is needed to get a nontrivial bound other than $w \cdot S(0) - K$ (cf. [7]). Notice that the method solves much larger problems if n is smaller.

Table 2: Computational times (sec.) for finding lower bounds.

n	$\mathfrak{C}$	m				n	$\mathfrak{C}$	m			
		10	20	30	40			2	6	10	14
3	$\mathfrak{C}_0$	0.02	0.20	0.93	3.13	6	$\mathfrak{C}_0$	0.09	1.09	33.01	
	$\mathfrak{C}_1$	0.10	0.23	0.92	3.05		$\mathfrak{C}_1$	1.58	2.20	95.15	
	$\mathfrak{C}_2$	1.31	0.29	0.85	1.94		$\mathfrak{C}_2$	0.29	1.96	47.03	
	$\mathfrak{C}_3$	0.90	1.04	1.41	2.32		$\mathfrak{C}_3$	1.64	3.07	3.51	11.33
4	$\mathfrak{C}_0$	0.02	5.98			8	$\mathfrak{C}_0$	0.43	1731.43		
	$\mathfrak{C}_1$	0.95	51.32				$\mathfrak{C}_1$	1.83	272.38		
	$\mathfrak{C}_2$	1.00	6.32	38.62	116.10		$\mathfrak{C}_2$	1.21	1037.64		
	$\mathfrak{C}_3$	0.56	3.63	20.32	37.02		$\mathfrak{C}_3$	2.27	8.17	13.03	38.73
5	$\mathfrak{C}_0$	2.75				10	$\mathfrak{C}_0$	0.23			
	$\mathfrak{C}_1$	14.47					$\mathfrak{C}_1$	1.96			
	$\mathfrak{C}_2$	2.11	91.50				$\mathfrak{C}_2$	0.31			
	$\mathfrak{C}_3$	1.46	11.17	35.12	1367.97		$\mathfrak{C}_3$	2.41	9.57	57.92	186.44

Table 2 reports the computational times of all the cases considered. The blank cells mean that the corresponding problem is not solvable due to its large problem size. For comparison, we also implement the linear programming approach of [10] (although not reported). However, both variables and constraints increase exponentially in size, making it hardly solvable except for small-size problems. Recall that $\mathfrak{C}_1$ and $\mathfrak{C}_2$ are subsets of $\mathfrak{C}_0$ but their sizes are still exponentially large. With the new set of constraints $\mathfrak{C}_3$, we see that the region for (n, m) that is solvable is much larger. Moreover, the computations are done in a reasonable amount of time. Also, the number of iterations is reasonably small so that `Lower_Bound` does not raise any practical concern at least in our extensive numerical tests. To be specific, the algorithm runs 6, 10, 8, 10 iterations for each $m \in \{2, 6, 10, 14\}$ when started with $\mathfrak{C}_3$ and $n = 6$. The numbers increase to 10, 28, 52, 70 when $n = 10$; however, the increase is proportional to the size of n and m.

4 Conclusion

Static-arbitrage bounds on basket option prices have been discussed by many academics and practitioners. While the upper bound is well understood theoretically and computationally, the lower bound is quite challenging from a computational point of view because of the intrinsic characteristics of the payoff function that makes the lower bound case fundamentally different from the upper bound case. We focused on the linear programming method proposed by several authors in recent years. In particular, we proposed to use a new solution approach in order to enlarge the efficiently solvable problem instances.

In the proposed method, we considered a discretized version of the dual semi-infinite linear program. Then, for its associated separation problem, devised was an algorithm whose required computational load is polynomial in the basket size and the number of calls per each asset. With the algorithm as a subroutine, a practical yet effective solution procedure was suggested. We noted from the numerical tests that the proposed

approach was able to solve large size problems in a computationally competitive manner. However, since the performance depends on the choice of initial set of constraints, one remaining question to be answered is to find a good set of initial constraints so as to enhance the efficiency and effectiveness of the algorithm.

Acknowledgments

The authors thank Professors Peña and Zuluaga for sending their code and paper. Kim's work was supported by the National Research Foundation of Korea, funded by the Ministry of Education (NRF-2014R1A1A2054868).

References

[1] Carmona, R., Durrleman, V.: Pricing and hedging spread options. SIAM Review. 45, 627–685 (2003)

[2] d'Aspremont, A., El Ghaoui, L.: Static arbitrage bounds on basket option prices. Mathematical Programming, Series A. 106, 467–489 (2006)

[3] Daum, S., Werner, R.: A novel feasible discretization method for linear semi-infinite programming applied to basket option pricing. Optimization. 60, 1379–1398 (2011)

[4] Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Berlin, Springer-Verlag (1988)

[5] Hobson, D., Laurence, P., Wang, T.-H.: Static-arbitrage optimal subreplicating strategies for basket options. Insurance: Mathematics and Economics. 37, 553–572 (2005)

[6] Hobson, D., Laurence, P., Wang, T.-H.: Static-arbitrage upper bounds for the prices of basket options. Quantitative Finance. 5, 329–342 (2005)

[7] Laurence, P., Wang, T.-H.: Sharp upper and lower bounds for basket options. Applied Mathematical Finance. 12, 253–282 (2005)

[8] Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. New York, John Wiley & Sons (1999)

[9] Peña, J., Saynac, X., Vera, J.C., Zuluaga, L.F.: Computing general static-arbitrage bounds for European basket options via Dantzig-Wolfe decomposition. Algorithmic Operations Research. 5, 65–74 (2010)

[10] Peña, J., Vera, J.C., Zuluaga, L.F.: Static-arbitrage lower bounds on the prices of basket options via linear programming. Quantitative Finance. 10, 819–827 (2010)

[11] Peña, J., Vera, J.C., Zuluaga, L.F.: Computing arbitrage upper bounds on basket options in the presence of bid-ask spreads. European Journal of Oprational Research. 222, 369–376 (2012)

[12] Schrijver, A.: Theory of Linear and Integer Programming. New York, Wiley (1986)

[13] Tankov, P.: Improved Frechet bounds and model-free pricing of multi-asset options. Journal of Applied Probability. 48, 389–403 (2011)