A new upper bound of the Euclidean TSP constant

Julien Yu\textsuperscript{1} and John Gunnar Carlsson\textsuperscript{*1}

\textsuperscript{1}Epstein Department of Industrial and Systems Engineering, University of Southern California

Abstract

Let $X_1, X_2, \ldots, X_n$ be $n$ independent and uniformly distributed random points in a compact region $R \subset \mathbb{R}^2$ of area 1. Let $TSP(X_1, \ldots, X_n)$ denote the length of the optimal Euclidean traveling salesman tour that traverses all these points. The classical Beardwood-Halton-Hammersley theorem (1959) proved the existence of a universal constant $\beta_2$ whose best bounds are $0.625 \leq \beta_2 \leq 0.92116$. Building upon an approach proposed by Steinerberger (2015), we present a computer-aided proof that improves its upper bound to $\beta_2 < 0.90304$.

1 Introduction

For any $n$ independent and uniformly distributed random points $X_1, \ldots, X_n$ in the unit square $[0, 1]^2$, let $TSP(X_1, \ldots, X_n)$ denote the length of the shortest Euclidean traveling salesman tour through all these points. Beardwood et al. [3] proposed one of the earliest limit theorems in the realm of combinatorial optimization.

**Theorem 1.** With probability one, there exists a universal constant $\beta_2$ such that

$$\lim_{n \to \infty} \frac{TSP(X_1, \ldots, X_n)}{\sqrt{n}} = \beta_2$$

where $\beta_2$ is a constant between 0.625 and 0.92116.

The Beardwood-Halton-Hammersley (BHH) Theorem is now considered classic and was frequently cited in a variety of books on probability theory [1, 5, 8, 10, 21, 26, 27]. It holds for any compact region $R \subset \mathbb{R}^2$ of area 1, and under periodic boundary conditions [15, 18]. Similar limiting behaviors are seen in the minimum spanning tree, Steiner tree and minimum weight matching problems [19, 20]. Beardwood et al. [3] also proved the existence of an absolute constant $\beta_d$ for every integer $d \geq 2$.

**Theorem 2.** For any dimension $d$, let $\{X_i\}$ be a sequence of independent and identically distributed random variables that are sampled from an absolutely continuous probability density function $f$ in a...
compact region $R \subset \mathbb{R}^d$. Then with probability one, there exists a universal constant $\beta_d$ such that

$$
\lim_{n \to \infty} \frac{TSP(X_1, \ldots, X_n)}{n^{\frac{d-1}{d}}} = \beta_d \int_R f_c(x)^{\frac{d-1}{d}} \, dx
$$

where $\beta_d$ is a dimension-dependent constant.

The values of $\beta_d$ are presently unknown, though in prior literature there were a myriad of approaches to give numerical estimates for $\beta_2$ [7, 13, 15, 16, 17, 18, 22, 25], $\beta_3$ and $\beta_4$ [15, 18]. These approximations were mainly built upon the linear programming relaxation by Held and Karp [11, 12]. The most recent estimates by Applegate et al. [1, 2] simulated very large instances and showed that $\beta_2 \sim 0.7124$. Yet, not until 2015 did literature [9, 23] begin to focus on improving the guaranteed bounds of $\beta_2$, and the improvement has been slight.

The original bounds for $\beta_2$ in [3] were

$$
0.625 = \frac{5}{8} \leq \beta_2 \leq \beta_{UB}
$$

where

$$
\beta_{UB} = 2 \int_{0}^{\infty} \int_{0}^{\sqrt{3}} \sqrt{z_1^2 + z_2^2} e^{-\sqrt{3}z_1} \left(1 - \frac{z_2}{\sqrt{3}}\right) d\pi_2 \pi_1 \approx 0.92116027
$$

Here, Beardwood et al. [3] falsely claimed the better upper bound $\beta_2 \leq 0.92037$, which was cited in many books and papers. In 2015, Steinerberger [23] spotted the mistake and provided useful insights to improve the bounds of $\beta_2$ by positive constants. His bound improvements were shown as

$$
\frac{5}{8} + \frac{19}{10368} \leq \beta_2 \leq \beta_{UB} - \epsilon_0
$$

for some explicit

$$
\epsilon_0 > \frac{9}{16} 10^{-6}
$$

The lower bound was further improved in [9] to be $\beta_2 \geq 0.6277$. However, the explicit upper bound improvement was negligible. In this paper, we propose a framework to prove that $\beta_2 < 0.9$, which is also applicable to higher dimensional $\beta_d$.

2 Approaches for the upper bound

2.1 Prior work and preliminaries

Note that for independent and uniformly distributed $X_1, \ldots, X_n$, the following limiting property holds

$$
\frac{\mathbb{E}[TSP(X_1, \ldots, X_n)]}{\sqrt{n}} \to \beta_2
$$
Let $\mathcal{P}_n$ be a Poisson process with intensity $n$ in a unit square $[0, 1]^2$, we have

$$\frac{\mathbb{E}[TSP(\mathcal{P}_n)]}{\sqrt{n}} \to \beta_2$$

where $TSP(\mathcal{P}_n)$ denotes the length of the shortest traveling salesman tour of points sampled from the Poisson process $\mathcal{P}_n$.

We look at a particular set $S_h$ within the unit square $[0, 1]^2$, written as

$$S_h = \{(x, y) \in \mathcal{P}_n : y \leq \frac{h}{\sqrt{n}}\}$$

The set $S_h$ includes all $\mathcal{P}_n$-generated points within a horizontal strip of height $h/\sqrt{n}$ and width 1. The $x$-coordinates of points within the strip follow a Poisson process with intensity $h \sqrt{n}$ on $[0, 1]$. To find a path connecting all points in $S_h$, the easiest way is to connect adjacent points with respect to their $x$-coordinates.

![Figure 1: A strip containing some points](image)

**Definition 1.** Label $(x_1, y_1), \ldots, (x_k, y_k) \in S_h$ such that $x_1 \leq x_2 \leq \ldots \leq x_k$. For $i = 1, \ldots, k-1$, connect $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ with a line segment. Such a path connecting all points in $S_h$ is called an $x$-adjacent path. See Figure 1.

In an $x$-adjacent path, the $x$-coordinate difference between any two endpoints of a segment follows an exponential distribution with mean $1/h \sqrt{n}$. $y$-coordinates of all points follow the uniform distribution with bounds 0 and $h/\sqrt{n}$. Consider $k$ $\mathcal{P}_n$-generated points $(X_1, Y_1), \ldots, (X_k, Y_k)$ within the horizontal strip of height $h/\sqrt{n}$ and width 1, where $k \sim h \sqrt{n}$ and $X_1 \leq X_2 \leq \ldots \leq X_k$. As $n \to \infty$,

$$X_i - X_{i-1} \sim \frac{1}{h \sqrt{n}} Z_i, \ Z_i \sim \text{Exponential}(1)$$

Meanwhile, we can write $Y_i = (h/\sqrt{n}) U_i$, where $U_i \sim \text{Unif}(0, 1)$. The expected distance of two $x$-adjacent points in Figure 1 is

$$\mathbb{E} \|(X_{i+1} - X_i, Y_{i+1} - Y_i)\| = \frac{1}{h \sqrt{n}} \mathbb{E} \|(Z_{i+1}, h^2(U_{i+1} - U_i))\|$$

The expected traveling salesman tour is bounded above by $n$ times the above quantity.

$$\frac{\mathbb{E}[TSP(\mathcal{P}_n)]}{\sqrt{n}} \leq \sqrt{n} \mathbb{E} \|(X_{i+1} - X_i, Y_{i+1} - Y_i)\| = \frac{1}{h} \mathbb{E} \|(Z_{i+1}, h^2(U_{i+1} - U_i))\| \approx 0.92116$$

\**3**
when \( h = \sqrt{3} \). Note that \( h = \sqrt{3} \) by default as it leads to the smallest default upper bound.

In Section 4, analysis of alternative \( h \) will be further useful in deriving potentially better upper bounds of \( \beta_2 \). Last but not the least, endpoint connections between adjacent strips occur \( \sqrt{n}/h \) times. The cost of each connection is of order \( O(1/\sqrt{n}) \), leading to an additional aggregate cost of order \( O(1) \). It does not contribute to the constant \( \beta_2 \), as \( O(\sqrt{n}) \) is needed.

![Figure 2: A shorter zigzag than the x-adjacent path when \( k = 3 \)](image)

Now consider the question: to connect the points \((x_1, y_1), \ldots, (x_k, y_k)\), is there a shorter alternative to the x-adjacent path? This lays the foundation for our proof of a better upper bound of \( \beta_2 \).

We start by simulating local point clusters \((X_0, Y_0), \ldots, (X_k, Y_k)\), i.e., \( k + 1 \) points with adjacent x-coordinates. We can assume without loss of generality that the leftmost point’s x-coordinate is 0. When \( k + 1 \) points are simulated, the joint distribution follows

\[
(X_0, Y_0, (X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)
\]

\[
= (0, \frac{h}{\sqrt{n}} U_0), (\frac{h}{\sqrt{n}} Z_1, \frac{h}{\sqrt{n}} U_1), (\frac{h}{\sqrt{n}} (Z_1 + Z_2), \frac{h}{\sqrt{n}} U_2), \ldots, (\frac{1}{h \sqrt{n}} \sum_{j=1}^{k} Z_j, \frac{h}{\sqrt{n}} U_k)
\]

where \( Z_i \sim i.i.d \text{ Exponential}(h\sqrt{n}), \forall i = 1, 2, \ldots, k \), and \( U_i \sim i.i.d \text{ Unif}(0, 1), \forall i = 0, 1, \ldots, k \).

For the simulation instance shown in Figure 2, there exists a shorter path from \((x_0, y_0)\) to \((x_k, y_k)\) than the x-adjacent path. We hereby call such shorter path a zigzag. When \( k = 3 \), a zigzag exists iff

\[
\left\| \frac{X_0 - X_2}{Y_0 - Y_2} \right\| + \left\| \frac{X_2 - X_1}{Y_2 - Y_1} \right\| + \left\| \frac{X_1 - X_3}{Y_1 - Y_3} \right\| \leq \left\| \frac{X_0 - X_1}{Y_0 - Y_1} \right\| + \left\| \frac{X_1 - X_2}{Y_1 - Y_2} \right\| + \left\| \frac{X_2 - X_3}{Y_2 - Y_3} \right\|
\]

We are interested in instances where the x-adjacent path is shorter than the alternative path. Recall

\[
\frac{1}{h} \mathbb{E} \left\| \frac{Z_{i+1}}{h^2(U_{i+1} - U_i)} \right\| = \beta_{UB} \sim 0.92116, \forall i = 0, 1, 2
\]

We define the quantities \( L_{0123} \) and \( L_{0213} \) that are proportional to the x-adjacent path length and the zigzag length, respectively.

\[
L_{0123} = \frac{1}{h} \left( \left\| \frac{Z_1}{h^2(U_0 - U_1)} \right\| + \left\| \frac{Z_2}{h^2(U_1 - U_2)} \right\| + \left\| \frac{Z_3}{h^2(U_2 - U_3)} \right\| \right)
\]

\[
L_{0213} = \frac{1}{h} \left( \left\| \frac{Z_1 + Z_2}{h^2(U_0 - U_2)} \right\| + \left\| \frac{Z_2}{h^2(U_2 - U_1)} \right\| + \left\| \frac{Z_2 + Z_3}{h^2(U_1 - U_3)} \right\| \right)
\]
It is evident that $\mathbb{E}(L_{0123}) = 3\beta_{UB}$. To achieve a better upper bound of $\beta_2$ (i.e., the 2-dimensional Euclidean TSP constant) than $\beta_{UB}$, we need instances where the zigzag is strictly shorter than the $x$-adjacent path. In general, our quantity of interest is the expectation of the minimum permutation.

$$\beta_2 \leq \frac{1}{kh} \mathbb{E} \min_{\sigma \in \Pi_k} \sum_{i=1}^k \left\| \frac{X_{\sigma(i)} - X_{\sigma(i-1)}}{h^2(U_{\sigma(i)} - U_{\sigma(i-1)})} \right\|$$

where $\Pi_k$ is the set of all permutations $\sigma$ of $\{0, \ldots, k\}$ such that $\sigma(0) = 0$ and $\sigma(k) = k$.

When we simulate $k+1$ points, there are $(k-1)!$ permutation options. In our analysis, we look at 5-point clusters because it strikes a balance between solution quality (i.e., upper bound improvement) and the computational cost (i.e., number of permutations). See Figure 3.

![Figure 3: Six permutation options when $k = 4$](image)

2.2 Analysis of high dimensional integral

To derive a better guaranteed upper bound of $\beta_2$, we need to convert the quantity of interest in (1) into high dimensional integral. The following theorem holds for $(X_1, \ldots, X_k)$.

**Theorem 3.** $Z_i \overset{i.i.d.}{\sim} \text{Exp}(1)$. $X_k = \sum_{i=1}^k Z_i$. Then the joint distribution of $(X_1, \ldots, X_k)$ follows

$$f(x_1, \ldots, x_k) = e^{-x_k}, 0 \leq x_1 \leq \ldots \leq x_k$$

**Proof.** We use proof by induction. The basis step is clear as $X_1 = Z_1 \sim \text{Exp}(1)$. Now assume that the statement holds for $(X_1, X_2, \ldots, X_{k-1})$, then the joint distribution on $(X_1, X_2, \ldots, X_{k-1}, X_k)$ is

$$f(x_1, \ldots, x_{k-1}, x_k) = f(x_1, \ldots, x_{k-1}) \mathbb{P}(X_k = x_k | X_{k-1} = x_{k-1})$$

Since $X_k = X_{k-1} + Z_k$, we have

$$\mathbb{P}(X_k = x_k | X_{k-1} = x_{k-1}) = \mathbb{P}(Z_k = x_k - x_{k-1}) = e^{-(x_k - x_{k-1})} \text{ for } x_k \geq x_{k-1}$$
from which the result follows. Thus, we can rewrite (1) as an integral that is not horrendously painful:

\[
\beta_2 \leq \frac{1}{kh} \int_D e^{-x_k} \min_{\sigma \in \Pi_k} \sum_{i=1}^k \left\| \left( \frac{x_{\sigma(i)} - x_{\sigma(i-1)}}{h^2(u_{\sigma(i)} - u_{\sigma(i-1)})} \right) \right\| dV \tag{2}
\]

where \( x_0 = 0 \) and

\[
D = \{ x_1, \ldots, x_k, u_0, \ldots, u_k : 0 \leq x_1 \leq \cdots \leq x_k, 0 \leq u_i \leq 1 \}
\]

We now represent the new upper bound of \( \beta_2 \) as a \( 2k + 1 \) dimensional integral. To attack this, we first take lots of samples with \( k + 1 \) points \((X_0, Y_0), \ldots, (X_k, Y_k)\), as discussed in Section 2.1. For most samples, the \( x \)-adjacent path generated from the identity permutation would still be the shortest amongst all permutation options. Yet, for a considerable percentage of samples, we find a shorter zigzag than the \( x \)-adjacent path. In Figure 4, we take 30 million samples with \( k = 4 \) and \( h = \sqrt{3} \). Out of the six permutation options, around 27% of samples have a shorter zigzag than the identity permutation. Remarkably, zigzags \( 0-1-3-2-4 \) and \( 0-2-1-3-4 \) both have around 10.5% chance of being the best permutation. Percentages would change when we pick alternative \( h \) and \( k \), but the observation that zigzags are shorter than the identity permutation in a non-negligible portion of samples is clear.

![Figure 4: Shortest zigzag length versus \( x \)-adjacent path length](image)

But how does the sampling of point clusters lead to a better guaranteed upper bound of \( \beta_2 \)? The main idea is to find lots of disjoint rectangular boxes \( B \subset D \) such that each \( B \) “prefers” a permutation \( \sigma_B \) other than the identity \( \sigma_I \). When we simulate \( k + 1 \) 2-dimensional points in a sample, we could rewrite it as a \( 2k + 1 \) dimensional hyperpoint denoted as \((x_1, x_2, \ldots, x_k, y_0, y_1, \ldots, y_k)\). We then seek lots of \( 2k + 1 \) dimensional rectangular boxes \( B \) that contain some simulated hyperpoints and satisfy

\[
B = \{ x_1, \ldots, x_k, u_0, \ldots, u_k : s_i \leq x_i \leq t_i, a_j \leq u_j \leq b_j, \forall i, j \}
\]

\[
s.t. \quad 0 \leq s_1 \leq t_1 \leq \cdots \leq s_k \leq t_k, 0 \leq a_j \leq b_j \leq 1, \forall j = 0, 1, \ldots, k
\]
For all $x$ dimensions in the hyperrectangular box $B$, the lower and upper bounds are $s_i$ and $t_i$. Lower and upper bounds for $y$ dimensions are $a_j$ and $b_j$.

When the selection of box $B$ and permutation $\sigma_B$ leads to a shorter zigzag than the $x$-adjacent path (i.e., identity permutation), the upper bound of $\beta_2$ can be written as

\[
\beta_2 \leq \frac{1}{kh} \int_{D_B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_i - x_i-1}{h^2(u_i - u_{i-1})} \right\| \, dV + \frac{1}{kh} \int_{B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_{\sigma_B(i)} - x_{\sigma_B(i-1)}}{h^2(u_{\sigma_B(i)} - u_{\sigma_B(i-1)})} \right\| \, dV
\]

\[
= \frac{1}{kh} \int_{D} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_i - x_i-1}{h^2(u_i - u_{i-1})} \right\| \, dV (= \beta_{UB} \approx 0.92116 \text{ as } h = \sqrt{3})
\]

\[
- \frac{1}{kh} \int_{B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_i - x_i-1}{h^2(u_i - u_{i-1})} \right\| \, dV + \frac{1}{kh} \int_{B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_{\sigma_B(i)} - x_{\sigma_B(i-1)}}{h^2(u_{\sigma_B(i)} - u_{\sigma_B(i-1)})} \right\| \, dV
\]

\[
= \beta_{UB} - \epsilon(\sigma_B) \leq \beta_{UB}
\]

We call $\epsilon(\sigma_B)$ the contribution of permutation $\sigma_B$ on box $B$. It is formally defined as

\[
\max \left\{ 0, \frac{1}{kh} \left( \int_{B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_i - x_i-1}{h^2(u_i - u_{i-1})} \right\| \, dV - \int_{B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_{\sigma_B(i)} - x_{\sigma_B(i-1)}}{h^2(u_{\sigma_B(i)} - u_{\sigma_B(i-1)})} \right\| \, dV \right) \right\}
\]

We consider positive contribution only because when identity permutation is the best, we choose not to take the box into consideration and the corresponding contribution would be zero.

When we find many disjoint hyper-rectangular boxes, the corresponding contributions are cumulative. So does our improvement to the guaranteed upper bound of $\beta_2$.

Meanwhile, note that the following integral separates:

\[
\int_{B} e^{-x_k} \sum_{i=1}^{k} \left\| \frac{x_{\sigma_B(i)} - x_{\sigma_B(i-1)}}{h^2(u_{\sigma_B(i)} - u_{\sigma_B(i-1)})} \right\| \, dV = \sum_{i=1}^{k} \int_{B} e^{-x_k} \left\| \frac{x_{\sigma_B(i)} - x_{\sigma_B(i-1)}}{h^2(u_{\sigma_B(i)} - u_{\sigma_B(i-1)})} \right\| \, dV
\]

This means that all we have to do is to evaluate expressions of the form

\[
\int_{B} e^{-x_k} \left\| \frac{x_{\sigma_B(i)} - x_{\sigma_B(i-1)}}{h^2(u_{\sigma_B(i)} - u_{\sigma_B(i-1)})} \right\| \, dV \tag{3}
\]

for $i = 1, 2, \ldots, k$. The variable $x_k$ only appears inside the very last summand where $\sigma_B(k) = k$. So for the first $k - 1$ summands, the expression (3) reduces to

\[
C_{B,v,w} \int_{s_v}^{t_v} \int_{s_w}^{t_w} \int_{a_w}^{b_w} \int_{a_v}^{b_v} \left\| \frac{x_v - x_w}{h^2(u_v - u_w)} \right\| \, du_v du_w dx_v dx_w \tag{4}
\]

where $v = \sigma_B(i), w = \sigma_B(i - 1)$, and $C_{B,v,w}$ is a quantity dependent on $B, v$ and $w$. The step-by-step calculation of $C_{B,v,w}$ will be demonstrated in Appendix 7.1. In Section 3, we will show the calculation of the above four-dimensional integral using FriCAS [24], a general purpose computer algebra system that can calculate antiderivatives for polynomials.
For the last summand in which \( i = k \), the expression (3) reduces to

\[
C_{B,v} \int_{s_k}^{t_k} \int_{s_v}^{t_v} \int_{a_k}^{b_k} \int_{a_v}^{b_v} e^{-x_k} \left\| \frac{x_v - x_k}{h^2(u_v - u_k)} \right\| du_v du_k dx_v dx_k
\]

(5)

where \( v = \sigma_B(k-1) \), and \( C_{B,v} \) is a quantity dependent on \( B \) and \( v \). The step-by-step calculation of \( C_{B,v} \) is also demonstrated in Appendix 7.1. Even though there is no closed form expression for (5), we can still handle it with FriCAS without harming the sanity of our proof.

3 Derivation of the upper bound

3.1 Calculation of antiderivative

We showed in Section 2.2 that the explicit calculation of expressions (4) and (5) would guarantee our improvement to the upper bound of \( \beta_2 \). In this section, we illustrate the calculation of antiderivatives using FriCAS [24]. If \( x_v \neq x_w \), expression (4) can be explicitly written as:

\[
C_{B,v,w} \int_{s_v}^{t_v} \int_{s_w}^{t_w} \int_{a_v}^{b_v} \int_{a_w}^{b_w} \left( \frac{x_v - x_w}{h^2(u_v - u_w)} \right) du_v du_w dx_v dx_w = C_{B,v,w} \frac{A_1 A_2 + B_1 B_2 + C + D_1 + D_2}{2880h^2}
\]

\[A_1 = 240h^4x_wu_w^4 - 960h^4x_wu_w^3 + 1440h^4x_wu_w^2 - 960h^4x_wu_w - 240h^4x_w^2 + 240h^4x_w^4 + 960h^4x_uu_u^3 - 1440h^4x_uu_u^2 + 960h^4x_uu_u - 240h^4x_u^4
\]

\[A_2 = \log\left( \sqrt{x_v^2 - 2x_vx_w + x_w^2 + h^2u_w^2} - 2h^2u_w x_w + h^2u_w - x_v - hu_w + hu_v \right)
\]

\[B_1 = 120h^4x_w^2u_w - 120h^4x_w - 480hx_wx_w^3u_w + 480hx_wx_w^3u_w + 720h^2x_w^2u_w - 720h^2x_w^2 - 480hx_wx_w^3u_w + 480hx_wx_w^3u_w + 720h^2x_w^2u_w - 480hx_wx_w^3u_w + 120h^4x_wu^3 + 120h^4x_wu^4
\]

\[B_2 = \log\left( \sqrt{x_w^2 - 2x_wx_u + x_u^2 + h^2u_u^2} - 2h^2u_u x_u + h^2u_u - x_w - h^2u_u + hu_u \right)
\]

\[C = (-48x_v^4 + 192x_vx_w^3 - 288x_v^2x_w^2 + 144h^2x_v^2u_w^2 - 288h^2x_v^2u_w - 144h^2x_v^2u_w^2 + 192x_v^4x_w - 288h^2x_v^2u_w + 576hx_vx_wu_w - 288hx_vx_wu_w - 48x_v^4 + 144h^2x_v^2u_w - 288h^2x_v^2u_w + 144h^2x_v^2u_w - 288h^4u_w^2 + 192h^4u_w^4 - 288h^4u_w^2 + 192h^4u_w^4 - 48h^4u_w^4)\sqrt{x_v^2 - 2x_vx_w + x_w^2 + h^2u_w^2} - 2h^2u_w x_w + h^2u_w - x_v - h^2u_w + hu_w
\]

\[D_1 = (-240h^4x_wx_w^4 + 960h^4x_wx_w^3 - 1440h^4x_wx_w^2u_w^2 + 960h^4x_wx_w^3u_w - 240h^4x_wx_w^4 + 240h^4x_wx_w^4 - 960h^4x_wx_w^3u_w + 1440h^4x_wx_w^2u_w^2 - 960h^4x_wx_w^3u_w + 240h^4x_wx_w^4)\log\left( |x_v - x_w| \right)
\]

\[D_2 = (-120h^4x_wx_w^4 + 480hx_wx_w^3u_w - 720hx_wx_w^3 - 240h^4x_wx_w^4 - 120h^4x_wx_w^4)\log(2) + 3x_w^5 - 30hx_w^4u_w + 30hx_w^4u_w - 30hx_w^4u_w + 280hx_w^3u_w - 280hx_w^3u_w - 40hx_w^3u_w - 40hx_w^3u_w - 420hx_w^3u_w + 420hx_w^3u_w + 280hx_w^3u_w - 280hx_w^3u_w + 120hx_w^3u_w
\]

\[h = \sqrt{3} \text{ by default but it can be any positive constant. In the limiting case where } x_w - x_v \to 0, \ A_2 \text{ and } B_2 \text{ are undefined. SageMath [6] is used to handle limiting cases, and the algebraic expressions in three possible limiting cases are explicitly shown in Appendix 7.2. Also, see the function } F_4 \_ \text{dir} \]
in `helper.py` for more details.

Though more complicated, expression (5) can also be written as:

$$C_{B,v} \int_{s_k}^{t_k} \int_{s_v}^{t_v} \int_{a_k}^{b_k} \int_{a_v}^{b_v} e^{-x_k} \left( \frac{x_v - x_k}{h^2(u_v - u_k)} \right) du_v du_k dx_v dx_k = C_{B,v} \int_{s_k}^{t_k} e^{-x_k} D(x_k) dx_k \tag{6}$$

for which a closed form expression does not exist. We can still use FriCAS to integrate out $x_v, u_v, u_k$ and solve $D(x_k)$ explicitly. All that is left is a 1-dimensional integral. If $x_v \neq x_k$ and $u_v \neq u_k$,

$$D(x_k) = \int_{s_v}^{t_v} \int_{a_k}^{b_k} \int_{a_v}^{b_v} \left( \frac{x_v - x_k}{h^2(u_v - u_k)} \right) du_v du_k dx_v = \frac{A_1 A_2 + B_1 B_2 + C + D}{2880 h^2}$$

$$A_1 = 48h x_k^3 u_v - 48h x_k^3 u_v - 48h x_k^3 u_v + 144h x_k^2 u_v + 144h x_k^2 u_v - 144h x_k^2 u_v - 48h x_k^3 u_k + 48h x_k^3 u_v + 12h^4 u_v - 48h^4 u_v u_k^3 + 72h^4 u_k^3 u_v - 48h^4 u_v u_k + 12h^4 u_v$$

$$A_2 = \log(\sqrt{x_k^2 - 2x_v x_k + x_v^2 + h^2 u_v^2 - 2h^2 u_v u_k + h^2 u_k^2 + hu_k - hu_v})$$

$$B_1 = -24h^4 u_v^4 + 96h^4 u_v u_k^3 - 144h^4 u_v u_k^2 + 96h^4 u_k^3 u_v - 24h^4 u_v$$

$$B_2 = \log(\sqrt{x_k^2 - 2x_v x_k + x_v^2 + h^2 u_v^2 - 2h^2 u_v u_k + h^2 u_k^2 - x_k + x_v + u_k - u_v})$$

$$C = (-24h^4 x_k^3 + 72h^2 x_k^2 - 72h^2 x_k x_v + 36h^2 x_v^2 + 72h^2 x_k x_v + 36h^2 x_v u_v + 24h^2 x_v^2 + 72h^2 x_k u_v - 36h^2 x_v u_v^2) \sqrt{x_k^2 - 2x_v x_k + x_v^2 + h^2 u_v^2 - 2h^2 u_v u_k + h^2 u_k^2}$$

$$D = (12h^4 u_k^4 + 96h^4 u_v u_k^3 + 144h^4 u_v u_k^2 - 96h^4 u_v^3 u_k + 144h^4 u_v^2 u_k^2 - 96h^4 u_k^3 u_v + 24h^4 u_k^4) \log(\sqrt{u_v - u_k}) + (48h^4 u_v^3 u_k - 72h^4 u_k^2 u_v + 48h^4 u_v^3 u_k) \log(2) + 36h^2 u_v u_k x_k - 8h^3 u_k^3 x_v + 8h^3 u_k^3 x_v - 3h^4 u_k^4 + 28h^4 u_k^3 u_v^2 + 42h^4 u_v^2 u_k^2 + 28h^4 u_v^3 u_k$$

In the limiting case where $x_k - x_v \to 0$ or $u_k - u_v \to 0$, $A_2, B_2$ and $D$ can be undefined. Limiting cases can be handled with SageMath [6], and their algebraic expressions are explicitly shown in Appendix 7.2. See the function `F_4_x4out_dir` in `helper.py` for more details. At this stage, the only unresolved component is the 1-dimensional integral in Equation (6).

### 3.2 Numerical analysis

In this section, we calculate the lower and upper bounds of the integral

$$\int_{s_k}^{t_k} e^{-x_k} D(x_k) dx_k$$

**Lemma 1.** $D(x_k)$ is convex and monotonically increasing.

**Proof.** Recall that

$$D(x_k) = \int_{s_v}^{t_v} \int_{a_k}^{b_k} \int_{a_v}^{b_v} \left( \frac{x_v - x_k}{h^2(u_v - u_k)} \right) du_v du_k dx_v, \text{ for } v = 1, 2, ..., k - 1$$
As \( x_k \geq x_v, D(x_k) \) is monotonically increasing in \( x_k \). Also, the Euclidean norm \( N(x_k; x_v, u_v, u_k) = \| x_v - x_k, h^2(u_v - u_k) \| \) is convex in \( x_k \) for every \( x_v, u_v, u_k \). Convexity is preserved under nonnegative scaling and addition, and this property extends to integrals [4].

More precisely, as \( N(x_k; x_v, u_v, u_k) \) is convex in \( x_k \) for arbitrary \( x_v, u_v, u_k \) in the domain \( A = \{(x_v, u_v, u_k) : s_v \leq x_v \leq t_v, a_v \leq u_v \leq b_v, a_k \leq u_k \leq b_k \} \),

\[
D(x_k) = \int_A N(x_k; x_v, u_v, u_k) \, du_v \, du_k \, dx_v
\]

is convex in \( x_k \). This completes our proof of the convexity of \( D(x_k) \).

\( e^{-x_k} \) is convex and monotonically decreasing. To find accurate bounds for expression (6), it is natural to approximate the exponential term \( e^{-x_k} \) with an \( n \)th degree Taylor polynomial \( P_n(x_k) \)

\[
e^{-x_k} \approx P_n(x_k) = 1 - x_k + \frac{x_k^2}{2} - \frac{x_k^3}{6} + \ldots + (-1)^n \frac{x_k^n}{n!}
\]

whose Lagrange error bound \( R_n(x_k) \) is

\[
R_n(x_k) = \frac{x_k^{n+1}}{(n + 1)!}
\]

This method is theoretically sound because FriCAS could solve antiderivatives with polynomial integrand. However, the process is computationally heavy on FriCAS. When \( k = 4, n = 22 \) terms are needed to achieve \( \mathbb{E}[R_n(X_k)] < 10^{-8} \). Hence, we partition the \( x_k \) axis and use tangent or secant lines to derive the lower or upper bound.

First partition the domain of \( x_k \) (i.e., \([s_k, t_k]\)) evenly into \( n \) pieces such that \( s_k = p_0 \leq p_1 \leq p_2 \leq \ldots \leq p_{n-1} \leq p_n = t_k \) and \( p_i = s_k + \frac{i(t_k - s_k)}{n} \). In our computational experiment, we let \( n = 1000 \).
For arbitrary interval \([p_i, p_{i+1}]\), calculate \(D(p_i), D(p_{i+1})\). Given the convexity of \(D(x_k)\) on \([s_k, t_k]\), the secant line that connects \((p_i, D(p_i))\) to \((p_{i+1}, D(p_{i+1}))\) can be denoted as \(a_i x_k + b_i\) and represents an overestimate of the function \(D(x_k)\) on \([p_i, p_{i+1}]\). The secant line that connects \((p_i, e^{-p_i})\) to \((p_{i+1}, e^{-p_{i+1}})\) is an overestimate of \(e^{-x_k}\) on \([p_i, p_{i+1}]\) and can be denoted as \(c_i x_k + d_i\). A quadratic upper bound \((a_i x_k + b_i)(c_i x_k + d_i)\) is then derived for \(e^{-x_k} D(x_k)\) for arbitrary \(x_k\). See Figure 5. We exaggerate interval lengths to make the upper bound visually discernible.

Similarly, a quadratic lower bound of \(e^{-x_k} D(x_k)\) for arbitrary \(x_k\) can be derived using tangent lines. Given the convexity of \(D(x_k)\) and \(e^{-x_k}\) on \([s_k, t_k]\), tangent lines that represent underestimates of both functions must exist. For arbitrary \([p_i, p_{i+1}]\), we use \(D'\left(\frac{p_i + p_{i+1}}{2}\right)\) and \(D(\frac{p_i + p_{i+1}}{2})\) to calculate the slope and intercept of the linear lower bound of \(D(x_k)\). See Appendix 7.2 for the expression \(D'(x_k)\). We essentially use trapezoidal and midpoint rule to construct upper and lower bounds, respectively.

It is worth emphasizing that we always use lower bound estimation for the \(x\)-adjacent path length and upper bound estimation for any potential zigzag length. The upper bound improvement of \(\beta_2\) is guaranteed when the contribution \(\epsilon(\sigma_B)\) is still positive. Also, when we want to prove a rigorous upper bound of \(\beta_2\), the floating point issue must be noticed. We use the interval arithmetic functionality in \textit{mpmath} [14], a Python library for arbitrary-precision floating-point arithmetic. The cumulative upper bound improvement of \(\beta_2\) can be represented as the lower bound of the interval arithmetic.

### 4 Computation of improvement

In this section, we calculate the improved upper bound of \(\beta_2\). Recall that the original upper bound by Beardwood et al. [3] was \(\beta_{UB} \approx 0.92116027\). The improved upper bound by Steinerberger [23] was \(\beta_2 \leq \beta_{UB} - \epsilon_0 \approx 0.9211597\). In Section 4.1, we discuss how different choices of \(h\) and \(k\) would affect the theoretical best upper bound using our approach. In Section 4.2, we introduce a mechanism to select \(2k + 1\) dimensional hyperrectangular boxes \(B\), and then present a bound calculation technique based on decision tree classification. In Section 4.3, we showcase our new guaranteed upper bound.

#### 4.1 Selection of parameters

In Section 2.2, we highlight that in order to improve the upper bound, we want to find lots of disjoint hyperrectangles \(B\), cost effective zigzag permutations \(\sigma_B\), and positive contributions \(\epsilon(\sigma_B)\). Recall that for \(h = \sqrt{3}\) and arbitrary \(k\)

\[
\frac{1}{kh} \int_{D} e^{-x_k} \sum_{i=1}^{k} \left\| \left( \frac{x_i - x_{i-1}}{h^2(u_i - u_{i-1})} \right) \right\| dV = \beta_{UB}(h) \approx 0.92116
\]

Now assume we select an alternative \(h\), expecting that it would lead to a better result for \(\beta_{UB}^{new}\). We simulate 30 million samples for \(k = 4\) and \(5\), taking \(h^2 = 2, 3, \ldots, 8\) and using Equation (1) as a quantity of interest. We introduce the quantity \(\beta_{UB}^{sim}\) that represents the theoretical improvement limit of our approach. It is estimated by averaging the shortest path length over all the samples.

Table 1 illustrates the computational result on \(\beta_{UB}^{sim}\). Each \((h, k)\) entry represents the 95% confidence interval of its theoretical best upper bound, obtained using bootstrapping.
Table 1: Selecting $h$ with Monte Carlo simulation

<table>
<thead>
<tr>
<th>$h$</th>
<th>$\beta_{UB}^{sim}$ as $k = 4$</th>
<th>$\beta_{UB}^{sim}$ as $k = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.414</td>
<td>$0.934055 \pm 5.3 \times 10^{-5}$</td>
<td>$0.931268 \pm 3.0 \times 10^{-5}$</td>
</tr>
<tr>
<td>1.732</td>
<td>$0.890195 \pm 4.1 \times 10^{-5}$</td>
<td>$0.884025 \pm 3.7 \times 10^{-5}$</td>
</tr>
<tr>
<td>2.000</td>
<td>$0.884960 \pm 3.2 \times 10^{-5}$</td>
<td>$0.874237 \pm 2.5 \times 10^{-5}$</td>
</tr>
<tr>
<td>2.236</td>
<td>$0.894581 \pm 3.0 \times 10^{-5}$</td>
<td>$0.878494 \pm 3.3 \times 10^{-5}$</td>
</tr>
<tr>
<td>2.449</td>
<td>$0.910974 \pm 2.8 \times 10^{-5}$</td>
<td>$0.888951 \pm 2.9 \times 10^{-5}$</td>
</tr>
<tr>
<td>2.646</td>
<td>$0.930837 \pm 3.4 \times 10^{-5}$</td>
<td>$0.902545 \pm 2.6 \times 10^{-5}$</td>
</tr>
<tr>
<td>2.828</td>
<td>$0.952621 \pm 2.7 \times 10^{-5}$</td>
<td>$0.917852 \pm 4.4 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

In our computational experiments in Section 4.2, we still mainly look at $(k, h) = (4, \sqrt{3})$ whose $\beta_{UB}^{sim}$ is roughly 0.8902 (this number is also stated in [23]). Indeed, selection of alternative $(h, k)$ combinations could lead to better results if we had unconstrained computation power and an immaculate mechanism to find hyperrectangles $B$ with positive contributions $\epsilon(\sigma_B)$. As shown in Table 1, $h = 2$ results in better $\beta_{UB}^{sim}$ than the default choice $h = \sqrt{3}$, for both $k = 4$ and 5.

When we look at larger $k$, $\beta_{UB}^{sim}$ reduces even more. We derive $\beta_{UB}^{sim}$ of as low as 0.852 when selecting $(k, h) = (9, \sqrt{5})$. As a trade-off, the required computation power skyrockets due to the exponential increase in permutations. Figure 6 illustrates how different choices of $h$, when combined with larger $k$, could result in much better $\beta_{UB}^{sim}$. However, actual improvements on $\beta_{UB}^{new}$ in this thought direction would require great progress in computation power.

4.2 Binary space partitioning

We now discuss our methodology of selecting hyperrectangular boxes $B$ in the $2k + 1$ dimensional hyperspace $D$, as defined in Section 2.2. We set $(k, h) = (4, \sqrt{3})$ and essentially use binary space
partitioning, which is generalizable to alternative \((k, h)\). We use the phrase “aggregate contribution” to denote the sum of all positive contributions \(\epsilon(\sigma_B)\), where the chosen hyperrectangles are all disjoint.

Figure 7 shows an instance of a 9-dimensional hyperrectangular box \(B \subset \mathcal{D}\), which is portrayed on the coordinate plane as one line segment together with four rectangular regions. For the illustrated hyperpoint \((x_1, x_2, x_3, x_4, u_0, u_1, u_2, u_3, u_4) \in B\), the zigzag \(\{0, 1, 3, 2, 4\}\) is shorter than the \(x\)-adjacent path \(\{0, 1, 2, 3, 4\}\). Yet, for some other hyperpoints in \(B\), some other permutation might work the best. The interim goal here is to find the ideal box \(B\), best permutation \(\sigma_B\), and contribution \(\epsilon(\sigma_B)\).

![Figure 7: Example of hyperrectangle \(B \subset \mathcal{D} \subset \mathbb{R}^9 (k = 4)\)](image)

For the box \(B\) illustrated in Figure 7, permutations \(\{0, 1, 3, 2, 4\}\), \(\{0, 2, 1, 3, 4\}\), \(\{0, 2, 3, 1, 4\}\) and \(\{0, 3, 1, 2, 4\}\) all lead to positive contribution, and \(\{0, 1, 3, 2, 4\}\) provides the highest contribution. That is, when we randomly choose a 9-dimensional hyperpoint in the Figure 7 hyperrectangle, the zigzag \(\{0, 2, 1, 3, 4\}\) is shorter in expectation than any other permutation including the identity.

The next step is to find lots of disjoint hyperrectangles such that their aggregate contribution is as large as possible. As it is unwise to select them manually, we need to partition \(\mathcal{D}\) into much smaller subsets and then assign the most ideal permutation to each of them.

We first take \(N\) samples of 9-dimensional hyperpoints and construct a data frame whose columns are \((x_1, x_2, x_3, x_4, u_0, u_1, u_2, u_3, u_4)\). The label column represents the best permutation index \((0 – 5,\text{ in lexicographical order})\) for each hyperpoint. We define the following functions in main.py:

- **calculate_gini**: calculate the Gini impurity of any nonempty dataset
- **calculate_information_gain**: calculate the information gain resulting from a binary split
- **calculate_threshold**: calculate the best feature to split and the corresponding threshold

We then construct the root node of a decision tree, starting with the \(N\)-row data set. At each internal node, the data set is split into two by calling the function **calculate_threshold**. A node is considered a leaf if and only if its corresponding data set is pure.
In Figure 8, we display a sample decision tree where \( N = 1000 \). Each node represents a subset of the 1000 sampled 9-dimensional hyperpoints. We define a class Tree that serves to construct the decision tree using recursion, and a class Node with the following attributes:

- **left**: left child of the current node, initially None
- **right**: right child of the current node, initially None
- **label**: permutation index (0 – 5), available at leaf nodes only
- **feature**: best feature \((x_i \text{ or } u_i)\) to be split at the current node
- **threshold**: best splitting threshold at the current node
- **ancestor**: deque featuring all ancestors of the current node

Further details are demonstrated in Appendix 7.3. Note that the **ancestor** attribute represents each node’s genealogy, which must be memorised in order to extract hyperrectangles at leaf nodes. For example, when we look at the second leaf node in layer 5 of Figure 8, the **ancestor** attribute constrains that the resulting hyperrectangle must satisfy: \( x_4 \leq 3.69486 \) (left), \( x_3 \geq 1.74906 \) (right), \( x_1 \geq 1.01349 \) (right), \( x_2 \geq 2.67963 \) (right), \( u_1 \leq 0.38197 \) (left). Given the limited depth of the tree, no explicit constraint is specified for \( u_0, u_2, u_3, u_4 \), and only one bound is specified for the rest of the features. Indeed, each leaf node represents a 9-dimensional subset of \( D \) but not a hyperrectangle \( B \). We need to apply some *a priori* rules to truncate the subsets into 9-dimensional hyperrectangles.

For instance, the subset at leaf node **LRRRL** can be written as

\[
\{(x_1, x_2, x_3, x_4, u_0, u_1, u_2, u_3, u_4) : x_4 \leq 3.69486, \\
1.74907 \leq x_3, 1.01350 \leq x_1, 2.67964 \leq x_2, u_1 \leq 0.38197\}
\]
We first impose the necessary condition $0 \leq s_1 \leq t_1 \leq s_2 \leq t_2 \leq s_3 \leq t_3 \leq s_4 \leq t_4$ such that

$$B \subset \{(x_1, x_2, x_3, x_4, u_0, u_1, u_2, u_3, u_4) : 1.01350 \leq x_1 \leq 3.69486, 2.67964 \leq x_2 \leq 3.69486,$$

$$2.67964 \leq x_3 \leq 3.69486, 2.67964 \leq x_4 \leq 3.69486, 0 \leq u_1 \leq 0.38197, 0 \leq u_i \leq 1 \text{ for } i = 0, 2, 3, 4\}$$

and then use some predefined rules to truncate the subset into a valid hyperrectangle. In helper.py, the function `special_box` handles lots of special circumstances while truncating, and the function `leaf_box` serves to create the valid hyperrectangle. See Appendix 7.3 for more details.

Building the decision tree takes $O(N \log N)$ time for fixed $k$. For larger $N$, the contribution at each leaf node tends to be smaller because $D$ is more finely partitioned, and the aggregate contribution tends to be larger. See Section 4.3.

### 4.3 Computational result

Setting $(k, h) = (4, \sqrt{3})$, we run 20 instances each for $N = 1000, 10000, 100000, 1000000, 10000000$, and a single instance for $N = 30000000, 50000000, 75000000$. Experiments for $N = 1000000, 10000000$ are conducted on an Intel Xeon E5-2640 v3, 2.60 GHz CPUs with 59 GB memory per node, supported by USC Center for Advanced Research Computing (CARC). The rest of our experiments are done on a personal computer with Apple M1 chip, 8 cores and 8GB memory.

![Figure 9: New upper bound as $N$ increases](image)

Figure 9 shows how we achieve better upper bounds of $\beta_2$ (i.e., $\beta_{UB}^{new}$) as the sample size $N$ increases. The best, average, worst $\beta_{UB}^{new}$, as well as computation time, are all illustrated.

- $N = 10^3$, best aggregate contribution is $1.28 \times 10^{-4}$, average scaled time is 42 seconds.
- $N = 10^4$, best aggregate contribution is $1.22 \times 10^{-3}$, average scaled time is 358 seconds.
- $N = 10^5$, best aggregate contribution is $4.27 \times 10^{-3}$, average scaled time is $3.06 \times 10^3$ seconds.
- $N = 10^6$, best aggregate contribution is $9.45 \times 10^{-3}$, average scaled time is $2.42 \times 10^4$ seconds.
- $N = 10^7$, best aggregate contribution is $1.54 \times 10^{-2}$, average scaled time is $1.75 \times 10^5$ seconds.
• $N = 3 \times 10^7$, aggregate contribution is $1.73 \times 10^{-2}$, new upper bound is at most 0.90385.
• $N = 5 \times 10^7$, aggregate contribution is $1.80 \times 10^{-2}$, new upper bound is at most 0.90314.

With $N = 7.5 \times 10^7$, our new upper bound of $\beta_2$, i.e. $\beta_{UB}^{new}$, is at most

$$\beta_{UB}^{new} = 0.92116027 - 0.018129175 = 0.903031095 < 0.90304$$

Figure 10: New upper bound as $h$ changes

Figure 10 illustrates how the selection of different $h$ would affect $\beta_{UB}^{new}$. Fixing $k = 4$ and $N = 10000000$, we run 10 instances each for $h^2 = 2, 2.25, 2.5, 2.75, 3, 3.25, 3.5, 3.75, 4, 4.25, 4.5$. Computational result shows that the default selection $h = \sqrt{3}$ still gives us the best $\beta_{UB}^{new}$ under our current scheme of selecting hyperrectangles. Yet, it does not eliminate the potential of choosing alternative $h$.

5 Conclusion and future work

In this paper, we present a rigorous computer-aided proof stating that

**Theorem 4.** $\beta_2 \leq \beta_{UB}^{new} < 0.90304$

In our current experiments on the selection of hyperrectangles, we set $(k, h) = (4, \sqrt{3})$. We would investigate the performance of $\beta_{UB}^{new}$ when larger $k$ is chosen, and its asymptotic behavior as $k \to \infty$.

We are interested in the Beardwood-Halton-Hammersley constant for higher dimension $d$, given Theorem 2. There is a distinct constant $\beta_d$ for every dimension $d \geq 2$. Using our methodology, we may improve the upper bound of $\beta_d$ in general. We also want to analyze the upper bounds of other relevant mathematical constants with unknown exact values.

In 2020, Gaudio and Jaillet [9] improved the lower bound of $\beta_2$ based on the work of Steinerberger [23]. We hereby present an alternative way of thought in improving the lower bound.

Assume that $n$ points $X_1, \ldots, X_n$ are independently and uniformly sampled in the unit square $[0, 1]^2$. Divide the unit square into $\frac{n}{c}$ square cells such that each cell has side length $\sqrt{\frac{c}{n}}$. Number of
points in each cell follows the distribution \( \text{Poisson}(c) \). A lower bound of the TSP is thus

\[
TSP(X_1, X_2, ..., X_n) \geq TSP_0(X_1, X_2, ..., X_n) \geq \sum_{\text{all } k} TSP_0(C_k)
\]

where \( TSP_0 \) denotes the length of the shortest Euclidean traveling salesman tour in which traveling on the grid edges \( B \) is free, and \( C_k \) denotes the set of all points in cell \( k \). See Figure 11. The distance metric \( d_0 \) between points \( i, j \) in the same cell can be written as

\[
d_0(x_i, x_j) = \min\{\|x_i - x_j\|, d(x_i, B) + d(x_j, B)\}
\]

where \( d(x, B) \) is the shortest Euclidean distance from point \( x \) to grid edges \( B \).

We can sample the points in hexagonal grids and apply the similar relaxation idea, as shown in Figure 11. A new lower bound of \( \beta_2 \) can be calculated, based on the idea of Theorem 1.

6 Acknowledgement

The authors acknowledge the Center for Advanced Research Computing (CARC) at the University of Southern California for providing computing resources that have contributed to the research results reported within this publication. URL: https://carc.usc.edu.
References


7 Appendix

7.1 Integration (step-by-step)

This section shows how Equation (3) can be reduced to Equation (4) and (5) step-by-step. Recall that the $2k + 1$ dimensional space $\mathcal{D}$ is

$$\mathcal{D} = \{x_1, \ldots, x_k, u_0, \ldots, u_k : 0 \leq x_1 \leq \cdots \leq x_k, 0 \leq u_i \leq 1, \forall i\}$$

and any selection of hyperrectangular box $B \subset \mathcal{D}$ can be written as

$$B = \{x_1, \ldots, x_k, u_0, \ldots, u_k : s_i \leq x_i \leq t_i, a_j \leq u_j \leq b_j, \forall i, j\}$$

s.t. $0 \leq s_1 \leq t_1 \leq \ldots \leq s_k \leq t_k, 0 \leq a_j \leq b_j \leq 1, \forall j = 0, 1, \ldots, k$

For the first $k - 1$ summands, $v = \sigma_B(i), w = \sigma_B(i - 1)$ for any $i = 1, 2, \ldots, k - 1$. We have

$$(3) = \int_B e^{-x_k} \left\| \frac{x_v - x_w}{h^2(u_v - u_w)} \right\| dV$$

$$= \int_{s_1}^{t_1} \ldots \int_{s_k}^{t_k} \int_{a_0}^{b_0} \ldots \int_{a_k}^{b_k} e^{-x_k} \left\| \frac{x_v - x_w}{h^2(u_v - u_w)} \right\| du_k \ldots du_0 dx_k \ldots dx_1$$

$$= C_u \int_{s_1}^{t_1} \ldots \int_{s_k}^{t_k} \int_{a_w}^{b_w} \int_{a_v}^{b_v} e^{-x_k} \left\| \frac{x_v - x_w}{h^2(u_v - u_w)} \right\| du_v du_w dx_k \ldots dx_1$$

$$= C_B,v,w \int_{s_w}^{t_w} \int_{s_v}^{t_v} \int_{a_w}^{b_w} \int_{a_v}^{b_v} \left\| \frac{x_v - x_w}{h^2(u_v - u_w)} \right\| du_v du_w dx_v dx_w$$

where $C_B,v,w = (e^{-s_k} - e^{-t_k})\prod_{i=1, i \neq v,w}^{k-1} (t_i - s_i)\prod_{i=0, i \neq v,w}^{k} (b_i - a_i)$

For the last summand, $v = \sigma_B(k - 1)$. We have

$$(3) = \int_B e^{-x_k} \left\| \frac{x_v - x_k}{h^2(u_v - u_k)} \right\| dV$$

$$= \int_{s_1}^{t_1} \ldots \int_{s_k}^{t_k} \int_{a_0}^{b_0} \ldots \int_{a_k}^{b_k} e^{-x_k} \left\| \frac{x_v - x_k}{h^2(u_v - u_k)} \right\| du_k \ldots du_0 dx_k \ldots dx_1$$

$$= C_u \int_{s_1}^{t_1} \ldots \int_{s_k}^{t_k} \int_{a_k}^{b_k} \int_{a_v}^{b_v} e^{-x_k} \left\| \frac{x_v - x_k}{h^2(u_v - u_k)} \right\| du_v du_k dx_k \ldots dx_1$$

$$= C_B,v \int_{s_k}^{t_k} \int_{s_v}^{t_v} \int_{a_k}^{b_k} \int_{a_v}^{b_v} e^{-x_k} \left\| \frac{x_v - x_k}{h^2(u_v - u_k)} \right\| du_v du_k dx_v dx_k$$

where $C_B = \prod_{i=1, i \neq v}^{k-1} (t_i - s_i)\prod_{i=0, i \neq v}^{k-1} (b_i - a_i)$
7.2 FriCAS

This Appendix is an extension of Section 3.1. For the first \( k - 1 \) summands, the algebraic expression in limiting case 1 \( (x_v = x_w \text{ and } u_v < u_w) \) is

\[
\int_{s_w}^{t_w} \int_{s_v}^{t_v} \int_{a_w}^{b_w} \int_{a_v}^{b_v} \left( \frac{x_v - x_w}{h^2(u_w - u_v)} \right) du_v du_w dx_v dx_w = \frac{A_1 A_2 + B + C}{2880h^2}
\]

\[
A_1 = 120h^4x_w u_w - 120h^4x_w u_w - 480h^4x_w u_w^2 + 480h^4x_w u_w^3 + 720h^2x_w u_w - 720h^2x_w u_w^2 - 480h^3x_w u_w + 480h^2x_w u_w + 120h^4x_w u_w - 120h^4x_w u_w - 480h^4x_w u_w^3 + 720h^2x_w u_w^2 - 480h^4x_w u_w^3 + 120h^4x_w u_w^4 + 120h^4x_w u_w - 120h^4x_w u_w - 120h^4x_w u_w^3 - 720h^2x_w u_w^2 + 480h^4x_w u_w^3 - 120h^4x_w u_w^4
\]

\[
A_2 = \log(\left|\frac{\sqrt{x_v^2 - 2x_v x_w + x_w^2 + h^2 u_w^2} - 2h^2 u_w + h^2 u_w^2 + h u_w - h u_v}{\sqrt{x_v^2 - 2x_v x_w + x_w^2 + h^2 u_w^2} - 2h^2 u_w + h^2 u_w^2 + h u_w - h u_v}\right|)
\]

\[
B = \left(\frac{48h^4x_w u_w}{9} - 192h^4x_w u_w - 288h^2x_w u_w^2 + 480h^4x_w u_w^3 u_w + 144h^2x_w u_w^2 - 288h^2x_w u_w^3 u_w + 144h^2x_w u_w^2 + 48h^4x_w u_w^3 u_w + 120h^4x_w u_w^4 + 720h^2x_w u_w^2 - 48h^4x_w u_w^3 u_w + 120h^4x_w u_w^4 + 576h^2x_w u_w^2 - 288h^2x_w u_w^3 u_w - 48h^4x_w u_w^3 u_w + 120h^4x_w u_w^4 - 120h^4x_w u_w - 480h^4x_w u_w^3 u_w - 720h^2x_w u_w^2 + 480h^4x_w u_w^3 u_w - 120h^4x_w u_w^4 + 280h^2x_w u_w^2 - 40h^2x_w u_w^3 u_w - 40h^2x_w u_w^3 u_w - 420h^2x_w u_w^2 + 280h^2x_w u_w^2 - 280h^2x_w u_w^2 - 120h^3x_w u_w x_w + 120h^3x_w u_w x_w
\]

The algebraic expression in limiting case 2 \( (x_v = x_w \text{ and } u_v > u_w) \) is

\[
\int_{s_w}^{t_w} \int_{s_v}^{t_v} \int_{a_w}^{b_w} \int_{a_v}^{b_v} \left( \frac{x_v - x_w}{h^2(u_w - u_v)} \right) du_v du_w dx_v dx_w = \frac{A_1 A_2 + B + C}{2880h^2}
\]

\[
A_1 = 240h^4x_w u_w^4 - 960h^4x_w u_w^3 u_w + 1440h^4x_w u_w^2 u_w^2 - 960h^4x_w u_w^3 u_w + 240h^4x_w u_w^4 - 240h^4x_w u_w^4 + 960h^4x_w u_w^3 u_w - 1440h^4x_w u_w^2 u_w^2 + 960h^4x_w u_w^3 u_w - 240h^4x_w u_w^4
\]

\[
A_2 = \log(\left|\frac{\sqrt{x_v^2 - 2x_v x_w + x_w^2 + h^2 u_w^2} - 2h^2 u_w + h^2 u_w^2 + h u_w - h u_v}{\sqrt{x_v^2 - 2x_v x_w + x_w^2 + h^2 u_w^2} - 2h^2 u_w + h^2 u_w^2 + h u_w - h u_v}\right|)
\]

\[
B = \left(\frac{48h^4x_w u_w}{9} - 192h^4x_w u_w - 288h^2x_w u_w^2 + 480h^4x_w u_w^3 u_w + 144h^2x_w u_w^2 - 288h^2x_w u_w^3 u_w + 144h^2x_w u_w^2 + 48h^4x_w u_w^3 u_w + 120h^4x_w u_w^4 + 720h^2x_w u_w^2 - 48h^4x_w u_w^3 u_w + 120h^4x_w u_w^4 + 576h^2x_w u_w^2 - 288h^2x_w u_w^3 u_w - 48h^4x_w u_w^3 u_w + 120h^4x_w u_w^4 - 120h^4x_w u_w - 480h^4x_w u_w^3 u_w - 720h^2x_w u_w^2 + 480h^4x_w u_w^3 u_w - 120h^4x_w u_w^4 + 280h^2x_w u_w^2 - 40h^2x_w u_w^3 u_w - 40h^2x_w u_w^3 u_w - 420h^2x_w u_w^2 + 280h^2x_w u_w^2 - 280h^2x_w u_w^2 - 120h^3x_w u_w x_w + 120h^3x_w u_w x_w
\]

The algebraic expression in limiting case 3 \( (x_v = x_w \text{ and } u_v = u_w) \) is

\[
\int_{s_w}^{t_w} \int_{s_v}^{t_v} \int_{a_w}^{b_w} \int_{a_v}^{b_v} \left( \frac{x_v - x_w}{h^2(u_w - u_v)} \right) du_v du_w dx_v dx_w = \frac{A + B}{2880h^2}
\]
\[ A = (-48x_w^4 + 192x_w x_w^3 - 288x_w^2 x_w^2 + 144h^2 x_w u_w - 288h^2 x_w^2 u_w + 144h^2 x_w^2 u_v + 192x_v^3 x_w - 288h^2 x_v x_w u_w^2 + 576h^2 x_v x_w u_v u_w - 288h^2 x_v x_w u_v^2 - 48x_v^4 + 144h^2 x_v^2 u_w^2 - 288h^2 x_v^2 u_v u_w + 144h^2 x_v^2 u_v^2 - 48h^4 u_w + 192h^4 u_v^3 u_w - 288h^4 u_v^2 u_w^2 + 192h^4 u_v^3 u_w - 48h^4 u_v^4 \sqrt{x_w^2 - 2x_v x_w + x_v^2 + h^2 u_w^2 - 2h^2 u_v u_w + h^2 u_v^2} + 576h^2 x_v x_w u_v u_w - 288h^2 x_v x_w u_v^2 - 48x_v^4 + 144h^2 x_v^2 u_w^2 - 288h^2 x_v^2 u_v u_w + 144h^2 x_v^2 u_v^2 - 48h^4 u_w + 192h^4 u_v^3 u_w - 288h^4 u_v^2 u_w^2 + 192h^4 u_v^3 u_w - 48h^4 u_v^4 \sqrt{x_w^2 - 2x_v x_w + x_v^2 + h^2 u_w^2 - 2h^2 u_v u_w + h^2 u_v^2} \]

\[ B = (-120h^4 x_w u_w^4 + 480h^4 x_v x_w u_v^3 x_w - 720h^4 x_v x_w^6 u_v^2 + 480h^4 x_v x_w^3 u_v^3 u_w - 120h^4 x_w u_v^4 \log(2) + 3x_w^5 - 30h x_v^4 u_w + 30h x_v^4 u_v u_w + 280h x_v^3 x_w u_v u_w - 280h x_v^3 x_w u_v^2 x_w - 40h^2 x_w^3 u_v u_w + 80h^2 x_w^3 u_v^2 - 420h x_v^2 x_w^2 u_v u_w + 280h x_v^2 x_w^2 u_v^2 - 280h x_v^2 x_w x_v x_w + 120h^3 u_v^3 u_w x_v x_w \]

For the last summand, the algebraic expression of \( D(x_k) \) in limiting case 1 \((x_v = x_k \text{ and } u_v < u_k)\) is

\[ \int_{t_v}^{b_k} \int_{a_k}^{b_v} \left\| \frac{x_v - x_k}{h^2 (u_v - u_k)} \right\| du_v du_k dx_v = \frac{A_1A_2 + B}{288h^2} \]

\( A_1 = -12h^4 (u_k - u_v)^4 \log(h | u_k - u_v| + h u_v - h u_k) + 24h^4 (u_k - u_v)^4 \log(|u_k - u_v|) + (-24x_v^3 + 72x_v x_k^2 - 72x_v x_k^2 x_k + 36h^2 x_v u_k^2 - 72h^2 x_k u_v u_k + 36h^2 x_k u_k^2 + 24x_v^4 - 36h^2 x_v u_k^2 + 72h^2 x_v u_v u_k - 36h^2 x_v u_k^2) \]

\( A_2 = \sqrt{x_k^2 - 2x_v x_k + x_v^2 + h^2 u_k^2 - 2h^2 u_v u_k + h^2 u_v^2} \]

\( B = (12h^4 u_k^4 - 96h^4 u_v u_k^3 + 144h^4 u_v^2 u_k^2 - 96h^4 u_v^3 u_k) \log(-2h) + (48h^4 u_v u_k^3 - 72h^4 u_v^2 u_k^2 + 48h^4 u_v^3 u_k) \log(2) + 36h^2 u_v u_k x_k^2 - 8h^3 u_k^3 x_k + 8h^3 u_k^3 x_k + 3h^4 u_k^3 + 28h^4 u_v u_k^3 - 42h^4 u_v^2 u_k^2 + 28h^4 u_v^3 u_k \]

The algebraic expression of \( D(x_k) \) in limiting case 2 \((x_v = x_k \text{ and } u_v > u_k)\) is

\[ \int_{t_v}^{b_k} \int_{a_k}^{b_v} \left\| \frac{x_v - x_k}{h^2 (u_v - u_k)} \right\| du_v du_k dx_v = \frac{A_1A_2 + B}{288h^2} \]

\( A_1 = -12h^4 (u_k - u_v)^4 \log(h | u_k - u_v| + h u_v - h u_k) + 24h^4 (u_k - u_v)^4 \log(|u_k - u_v|) + (-24x_v^3 + 72x_v x_k^2 - 72x_v x_k^2 x_k + 36h^2 x_v u_k^2 - 72h^2 x_k u_v u_k + 36h^2 x_k u_k^2 + 24x_v^4 - 36h^2 x_v u_k^2 + 72h^2 x_v u_v u_k - 36h^2 x_v u_k^2) \]

\( A_2 = \sqrt{x_k^2 - 2x_v x_k + x_v^2 + h^2 u_k^2 - 2h^2 u_v u_k + h^2 u_v^2} \]

\( B = (12h^4 u_k^4 - 96h^4 u_v u_k^3 + 144h^4 u_v^2 u_k^2 - 96h^4 u_v^3 u_k) \log(-2h) + (48h^4 u_v u_k^3 - 72h^4 u_v^2 u_k^2 + 48h^4 u_v^3 u_k) \log(2) + 36h^2 u_v u_k x_k^2 - 8h^3 u_k^3 x_k + 8h^3 u_k^3 x_k + 3h^4 u_k^3 + 28h^4 u_v u_k^3 - 42h^4 u_v^2 u_k^2 + 28h^4 u_v^3 u_k \]

The algebraic expression of \( D(x_k) \) in limiting case 3 \((u_v = u_k)\) is

\[ \int_{t_v}^{b_k} \int_{a_k}^{b_v} \left\| \frac{x_v - x_k}{h^2 (u_v - u_k)} \right\| du_v du_k dx_v = (24h^4 u_k^4 \log(2) - 36h^4 u_k^4 \log(-2h) + 11h^4 u_k^4 + 8h^3 x_v u_k^3 - 8h^3 x_v u_k^3 + 36h^2 x_k u_k^2 + (24x_v^3 - 72x_v x_k + 72x_v x_k^2 - 24x_k^2) \sqrt{x_v^2 - 2x_v x_k + x_k^2})/(288h^2) \]
Last but not the least, we need to calculate the explicit form of \( D'(x_k) \) in Section 3.2. We have

\[
D'(x_k) = (A'_1A_2 + \frac{A_1A'_{2,num}}{A'_{2,den}} + \frac{B_1B'_{2,num}}{B'_{2,den}} + \frac{C'_{num}}{C'_{den}} + D')/(2880h^2)
\]

\[
A'_1 = 144hx_k^2u_k - 144hx_k^2u_v - 288hx_vx_ku_k + 288hx_vx_ku_v + 144hx_v^2u_k - 144hx_v^2u_v
\]

\[
A'_{2,num} = x_v - x_k
\]

\[
A'_{2,den} = (hu_v - hu_k)\sqrt{x_k^2 - 2x_vx_k + h^2u_v^2 - 2h^2u_ku_v + x_v^2 + h^2u_k^2} - x_k^2 + 2x_vx_k - h^2u_v^2 + 2h^2u_ku_v - x_v^2 - h^2u_k^2
\]

\[
B'_{2,num} = x_v - x_k - \sqrt{(x_k - x_v)^2 + h^2 \cdot (u_v - u_k)^2}
\]

\[
B'_{2,den} = (x_k - x_v)^2 + h^2 \cdot (u_v - u_k)^2 + (-x_k - h \cdot (u_v - u_k) + x_v)\sqrt{(x_k - x_v)^2 + h^2 \cdot (u_v - u_k)^2}
\]

\[
C'_{num} = -96x_k^4 + 384x_vx_k^3 - 576x_v^2x_k^2 + 384x_v^3x_k + 36h^4u_v^4 - 144h^4u_ku_v^3 + 216h^4u_k^2u_v^2 - 144h^4u_k^3u_v - 96x_v^4 + 36h^4u_k^2
\]

\[
C'_{den} = \sqrt{x_k^2 - 2x_vx_k + h^2u_v^2 - 2h^2u_ku_v + x_v^2 + h^2u_k^2}
\]

\[
D' = 72h^2u_vu_kx_k - 8h^3u_k^3
\]

Expressions of \( A_1, A_2, B_1 \) are the same as those defined in Section 3.1. \( A'_{2,num}, A'_{2,den} \) are the numerator and denominator of \( A'_2 \). \( B'_{2,num}, B'_{2,den} \) are the numerator and denominator of \( B'_2 \).

### 7.3 List of functions

This section describes our python files helper.py and main.py which serve to calculate \( \beta_{UB}^{new} \), using the methodology proposed in Section 3 and 4. The goal, input and output of each function are specified.

Always let \( k = 4 \). Permutations \( \{0, 1, 2, 3, 4\}, \{0, 1, 3, 2, 4\}, \{0, 2, 1, 3, 4\}, \{0, 2, 3, 1, 4\}, \{0, 3, 1, 2, 4\}, \{0, 3, 2, 1, 4\} \) are projected into labels \( 0, 1, 2, 3, 4, 5 \) accordingly. Functions in main.py are as follow:

- **generate_data**. Generate a data frame following the distributions described in Section 2
  
  Input: \( seed \); input into \texttt{np.random.seed} 
  
  Functionality: update the global variable \texttt{data} 

- **calculate_gini**. Calculate Gini impurity of a nonempty dataset
  
  Input: \( data; \) pandas data frame with 9 feature columns and 1 rightmost label column
  
  Output: nonnegative Gini impurity which depends only on the label column 

- **calculate_information_gain**. Calculate information gain resulting from a split
  
  Input: \( threshold \); a manually chosen number that serves as the threshold of the split
  
  Input: \( feature \); an element from the set \( \{ x_1, x_2, x_3, x_4, u_0, u_1, u_2, u_3, u_4 \} \)
  
  Output: same as in **calculate_gini**
• **calculate_threshold.** Decide the best splitting feature and threshold for an impure dataset

  **Input:** data; same as in *calculate_gini*

  **Output:** tuple featuring the best splitting feature and threshold

• **Node.** Class representing a node in the decision tree

  **Attribute:** left, right, label, feature, threshold, ancestor; well explained in Section 4.2

  **Function:** insert; link the Node object to its left and right children

  **Input:** leftchild, rightchild

• **Tree.** Class representing the decision tree structure

  **Attribute:** ancestor; same as in the Node class

  **Function:** create_decision_tree; build tree and update global variable leaf_nodes

  **Input:** data; leaf_nodes; global variable initialized as an empty list

• **worker.** Perform multiprocessing with the *leaf_saving* function from helper.py

  **Input:** leaf; an object created from the Node class

  **Output:** contribution of a legitimate hyperrectangle at the Node object leaf

• **main.** Calculate aggregate contribution and thereby \( \beta_{UB}^{new} \)

  **Step 1:** generate data with *generate_data*

  **Step 2:** update leaf_nodes and save the tree structure with *create_decision_tree*

  **Step 3:** multiprocess the calculation of (aggregate) contribution with worker

Meanwhile, the calculation of antiderivatives on FriCAS and the selection of hyperrectangles are written in a separate file named helper.py. Functions in helper.py are described as follow:

• **Fu_4, Fuv_4, Fxuv_4, Fxyuv_4.** Calculate antiderivatives for the first \( k - 1 \) summands

• **Fu_4out, Fuv_4out, Fxuv_4out.** Calculate the antiderivative for the last summand

• **F_4_dir.** Calculate the following antiderivative as \( v, w \neq 4 \)

  \[
  \int \int \int \left( \frac{x_v - x_w}{h^2(u_v - u_w)} \right) du_v du_w dx_v dx_w
  \]

  **Input:** \( x_v, x_w, u_v, u_w; \) all numeric

  **Output:** FriCAS antiderivative for the first \( k - 1 \) summands

• **F_4_x4out_dir.** Calculate the following antiderivative as \( v \neq 4 \)

  \[
  \int \int \left( \frac{x_v - x_k}{h^2(u_v - u_k)} \right) du_v du_k dx_v
  \]
Input: $x_v, x_k, u_v, u_k$; all numeric

Output: FriCAS antiderivative for the last summand, leaving $dx_k$ out

- $x4\_dir$. Calculate the following definite integral as $v, w \neq 4$

$$
\int_{s_w}^{t_w} \int_{s_v}^{t_v} \int_{a_w}^{b_w} \int_{a_v}^{b_v} \left( \frac{x_v - x_w}{h^2(u_v - u_w)} \right) du_v du_w dx_v dx_w
$$

Input: $x, y, u, v$; represent $[s_v, t_v], [s_w, t_w], [a_v, b_v], [a_w, b_w]$

Output: definite integral for the first $k - 1$ summands

- $x4\_x4out\_dir$. Calculate the following definite integral as $v \neq 4$

$$
D(x_k) = \int_{s_v}^{t_v} \int_{a_v}^{b_v} \int_{a_k}^{b_k} \left( \frac{x_v - x_k}{h^2(u_v - u_k)} \right) du_v du_k dx_v
$$

Input: $x, y, u, v$; represent $[s_v, t_v], [s_k, t_k], [a_v, b_v], [a_k, b_k]$

Output: definite integral for the last $k - 1$ summands

- tangent, $x4\_tangent\_exp$. Calculate the underestimation of the last summand

Input: $x, y, u, v$; represent $[s_v, t_v], [s_k, t_k], [a_v, b_v], [a_k, b_k]$

Output: quadratic lower bound of the last summand, calculated by multiplying tangent lines

- secant, $x4\_secant\_exp$. Calculate the overestimation of the last summand

Input: $x, y, u, v$; represent $[s_v, t_v], [s_k, t_k], [a_v, b_v], [a_k, b_k]$

Output: quadratic upper bound of the last summand, calculated by multiplying secant lines

- $x4\_dir\_0101, x4\_dir\_0202, \ldots, x4\_dir\_2323$. Calculate length of segments (not the last)

Input: $box$; hyperrectangle $[s_1, t_1], [s_2, t_2], [s_3, t_3], [s_4, t_4], [a_0, b_0], [a_1, b_1], [a_2, b_2], [a_3, b_3], [a_4, b_4]$

Output: exact length of a segment other than the last in a permutation

- $x4\_LB\_3434$. Calculate the underestimation of the last segment $3 \rightarrow 4$

Input: $box$; hyperrectangle $[s_1, t_1], [s_2, t_2], [s_3, t_3], [s_4, t_4], [a_0, b_0], [a_1, b_1], [a_2, b_2], [a_3, b_3], [a_4, b_4]$

Output: underestimation of segment length $3 \rightarrow 4$

- $x4\_UB\_1414, x4\_UB\_2424, x4\_UB\_3434$. Calculate the overestimation of the last segment

Input: $box$; hyperrectangle $[s_1, t_1], [s_2, t_2], [s_3, t_3], [s_4, t_4], [a_0, b_0], [a_1, b_1], [a_2, b_2], [a_3, b_3], [a_4, b_4]$

Output: overestimation of segment length $1 \rightarrow 4, 2 \rightarrow 4$ or $3 \rightarrow 4$

- dir\_01234\_LB. Calculate the underestimation of the $x$-adjacent path length

Input: $box$; hyperrectangle $[s_1, t_1], [s_2, t_2], [s_3, t_3], [s_4, t_4], [a_0, b_0], [a_1, b_1], [a_2, b_2], [a_3, b_3], [a_4, b_4]$

Output: underestimation of the $x$-adjacent path length
• *dir_01324_UB, ..., dir_03214_UB*. Calculate the overestimation of a non $x$-adjacent path

Input: *box*; hyperrectangle $[s_1, t_1], [s_2, t_2], [s_3, t_3], [s_4, t_4], [a_0, b_0], [a_1, b_1], [a_2, b_2], [a_3, b_3], [a_4, b_4]$

Output: overestimation of a non $x$-adjacent path length

• *calculate_saving*. Calculate $\epsilon(\sigma_B)$, contribution of hyperrectangle $B$ with permutation $\sigma_B$

Input: *box*; hyperrectangle $[s_1, t_1], [s_2, t_2], [s_3, t_3], [s_4, t_4], [a_0, b_0], [a_1, b_1], [a_2, b_2], [a_3, b_3], [a_4, b_4]$

Input: *label*; permutation index that can be 0, 1, 2, 3, 4, 5

Output: nonnegative contribution $\epsilon(\sigma_B)$

• *check_monotone*. Check if hyperrectangle $B$ is a legitimate input into *calculate_saving*

Input: *box_array*; candidate hyperrectangular box with possible violations such as $a_i > b_i$

Output: boolean that returns True if and only if the hyperrectangle is legitimate

• *special_box*. Truncate candidate box array if certain conditions are satisfied

Input: *box_array*; candidate hyperrectangular box with possible violations such as $a_i > b_i$

Output: a truncated box array

• *leaf_box*. Generate candidate box array from a pure leaf *Node* object

Input: *leaf*; a pure leaf *Node* object with known genealogy

Output: candidate hyperrectangular box with possible violations such as $a_i > b_i$

• *leaf_saving*. Calculate contribution $\epsilon(\sigma_B)$ from a leaf *Node* object; imported into main.py

Input: *leaf*; a pure leaf *Node* object with known genealogy

Output: nonnegative contribution $\epsilon(\sigma_B)$

This concludes our Appendix. Please see main.py and helper.py for more details.