Oct 21, 2024

Logarithmic Weighted Random Selector Algorithm: A Novel Approach for Biasing Selection Based on Positional Order Without Hyperparameters

23RD MEXICAN INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE
XVII WORKSHOW OF HYBRID INTELLIGENT SYSTEMS

By Iván Alejandro Ramos Herrera

WHAT?

"Logarithmic Random Weighted Selector“ (LWRS) objective is to make random selections from a list of elements, weighting the decision based on the preference order of the elements, without the need for known weight values or hyperparameters.

WHAT?

WHY?

The need for an algorithm that makes weighted selection decisions for phenomena with **unknown numerical weighting values,** but when their **order of preference is known.**

CONTEXT

Project MIA

MIA is a project for an algorithm that performs autonomous musical composition without the need for training based on musical pieces (work in progress).

In musical composition there are phenomena for which it is difficult to know a number that defines the frequency of appearance of certain elements, but for which their preference is known. For example, the selection of a *4/4-time signature* over *3/4* or *2/4*.

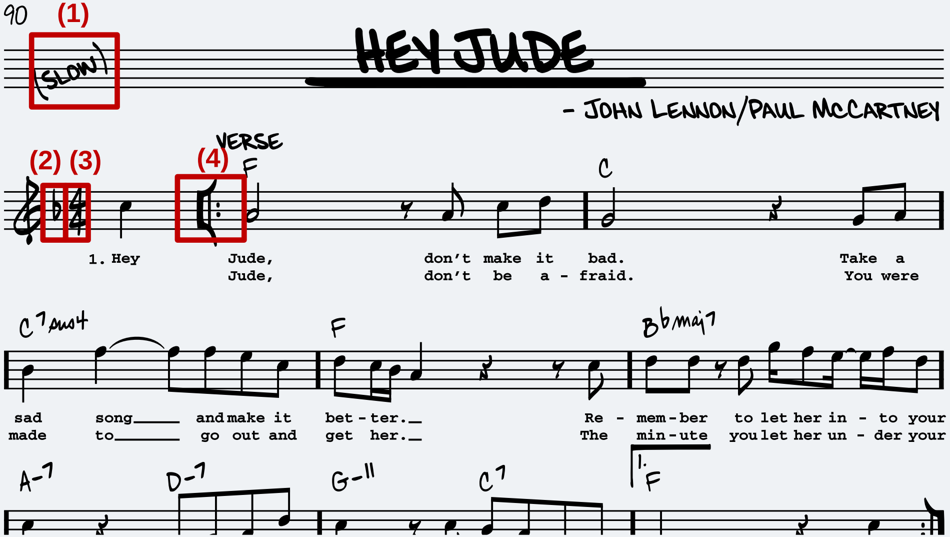


Fig 1. Music Sheet of "Hey Jude" song in which decision variables regarding the composition of a song are indicated, such as speed(1), tonality(2), time signature(3), and anacrusis(4).

Source: Musescore. (2024, 14 mayo). Hey Jude [Jazz version] - The Beatles (Real Book - Melody-Chords-Lyrics). https://musescore.com/user/39593079/scores/16030360.

Popular music exhibits a proclivity towards the *4/4-time* signature, simple major-minor chords, short chord progressions or repetitive rhythmic patterns [1].

In attempting to replicate popular music composition with generative algorithms without classic *Machine learning* techniques, it is imperative to consider these characteristics as the most common, but without losing the variety of decision.

WHAT?

"Logarithmic Random Weighted Selector" (LWRS) objective is to make random selections from a list of elements, weighting the decision based on the preference order of the elements, without the need for known weight values or hyperparameters.

METHODOLOGY*

1. **Understanding and establishment of the problem conditions.**
2. **Analysis of similar algorithms.**
3. **Discovery of weighting mechanisms.**
4. **Implementation.**
5. **Experimentation and results.**
6. **Conclusions and discussion.**

*Adaptation of the steps of *"How to solve a problem"* from the book in reference [2].

RELATED TECHNIQUES

Technique*	Randomness?	Offer bias?	Numeric weights?	Hyperparametrers?
Simple Random Sampling	✓			
Systematic Random Sampling	✓			✓
Stratified Random Sampling	✓	✓		✓
Cluster Random Sampling	✓	✓		✓
Multistage Random Sampling	✓	✓		✓
Quota Sampling		✓		✓
Fitness-proportionate Selection	✓	✓	✓	
Tournament Selection	✓	✓	✓	✓
Rank-based Selection	✓	✓		✓

*All these techniques and their characteristics are defined on references [3, 4, 5, 6].

●●●●

WHAT?

WHY?

HOW?

1. **Logarithmic weighting**
2. **Index Linear weighting**
3. **Exponential weigthing**

Based on the literature on techniques read, for selection mechanisms that offers bias, these three mechanisms were discovered, which were implemented in random selection algorithms with weighting based on the order of the elements for the posed problem.

LOGARITHMIC MECHANISM FOUND

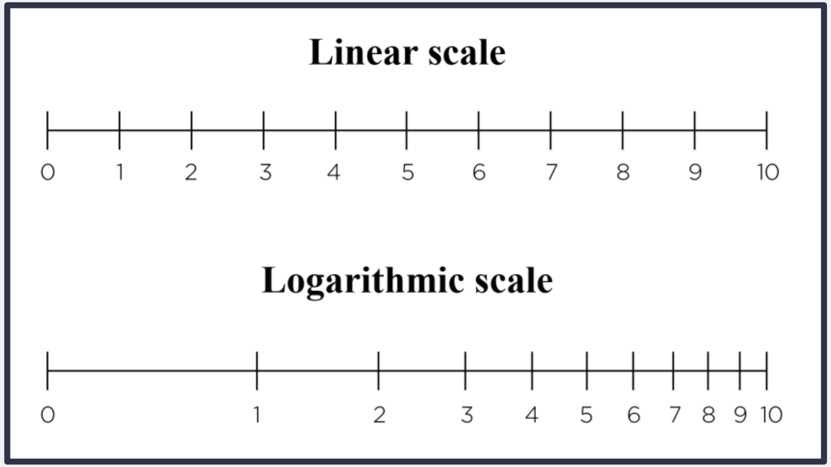


Fig 2. Comparison between Linear and Logarithmic sales.

For the logarithmic mechanism

Taking Simple Random Sampling as a reference and understanding its operation as a linear scale, it was discovered that there could be a mechanism based on a logarithmic scale.

LOGARITHMIC FORMULATION

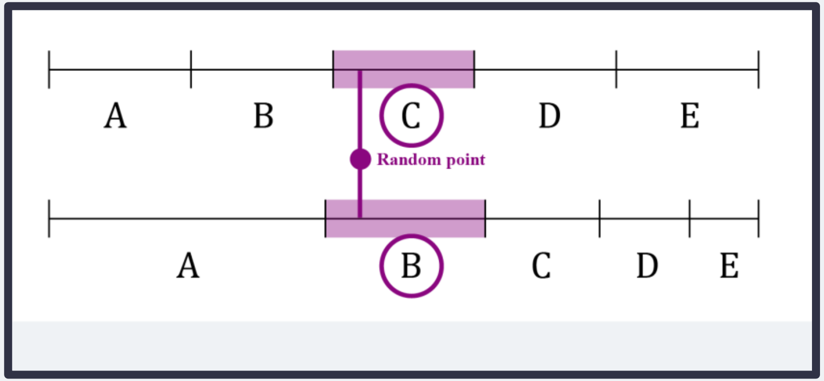


Fig 3. Comparison in selection example for five elements using a linear and logarithmic scales. The first scale (linear) works as a *Simple Random Selector*, where each element has the same space (probability to be selected). The second scale gives different spaces based on the order of the elements, based on a Logarithmic scale of five spaces.

For selection, LWRS builds a logarithmic scale adapted to the *n* elements of the input list, and the "space" assigned to each of the objects corresponds to its selection probability.

LOGARITHMIC FORMULATION

$$\mathbb{P} = \{\mathrm{p_i} : \mathrm{p_i} = (n \log_{\mathrm{n+1}}(i + 1), 0), n, i \in \mathbb{N}, 1 \leq i \leq n\}$$

The way to build a logarithmic scale adapted to the n quantity of objects to be selected is possible through simulating a $\mathbb{P}$ set of points on a Cartesian axis in which each point delimits the selection space of each of the n elements on the list. Where:

- p_i is (x, y) coordinate.
- n is the number of items.
- i is the index of point.

LOGARITHMIC FORMULATION

$$d_i = n \log_{n+1}(i + 1)$$

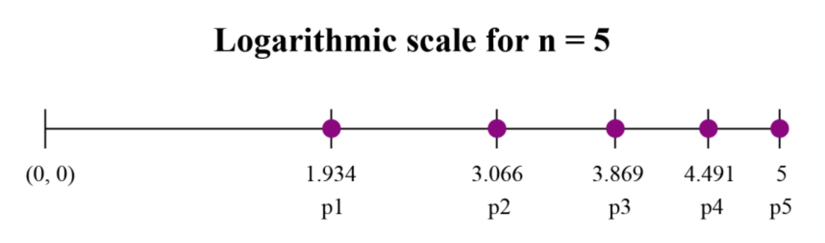
$$\boldsymbol{p_1} = (5 \log_{5+1}(1 + 1), 0) = \mathbf{(1.934, 0)}$$

$$\boldsymbol{p_2} = (5 \log_{5+1}(2 + 1), 0) = \mathbf{(3.066, 0)}$$

$$\boldsymbol{p_3} = (5 \log_{5+1}(3 + 1), 0) = \mathbf{(3.869, 0)}$$

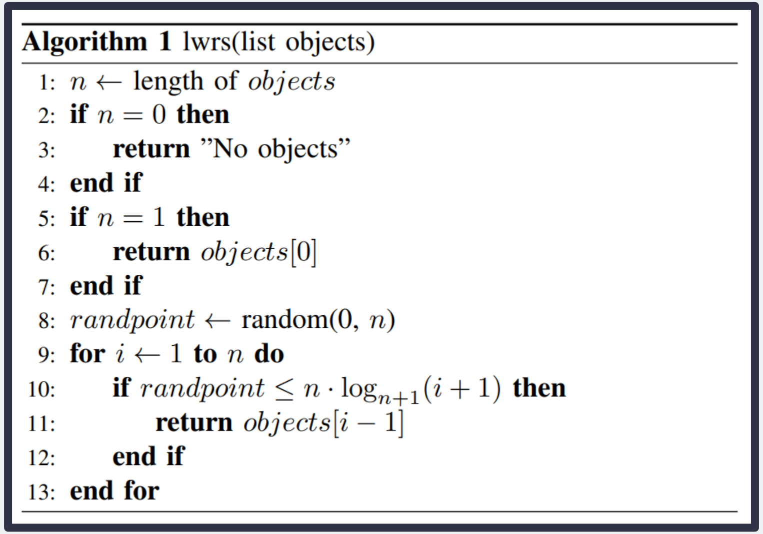
$$\boldsymbol{p_4} = (5 \log_{5+1}(4 + 1), 0) = \mathbf{(4.491, 0)}$$

$$\boldsymbol{p_5} = (5 \log_{5+1}(5 + 1), 0) = \mathbf{(5, 0)}$$



By calculating the distance of each point *p* on the scale, it is possible to know the weighting of each element assigned to the space delimited by said point.

LOGARITHMIC WEIGHTING SELECTION



Algorithm 1 lwrs(list objects)

```
$n \leftarrow$ length of $objects$
if $n = 0$ then
    return "No objects"
end if
if $n = 1$ then
    return $objects[0]$
end if
$randpoint \leftarrow$ random(0, $n$)
for $i \leftarrow 1$ to $n$ do
    if $randpoint \leq n \cdot \log_{n+1}(i + 1)$ then
        return $objects[i - 1]$
    end if
end for
```

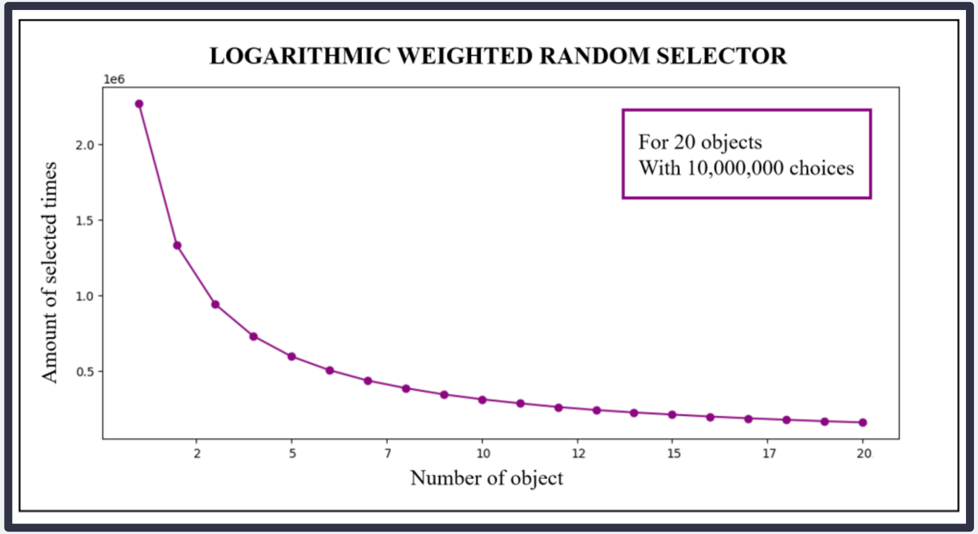
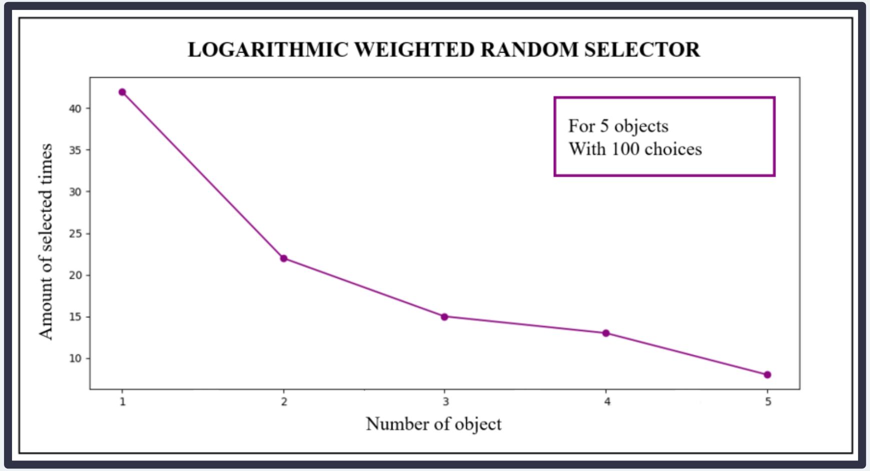


Fig. 4. Pseudocode of LWRS algorithm, with time complexities as:

- Worst case: O(n).
- Average case: Θ(log(n)).
- Best case: Ω(1).

Fig. 5. Experiment in which 10,000,000 repetitions of the LWRS algorithm were made, selecting from a list of 20 elements, counting and graphing how many times each object was selected to check the bias towards each one.

LOGARITHMIC WEIGHTING SELECTION



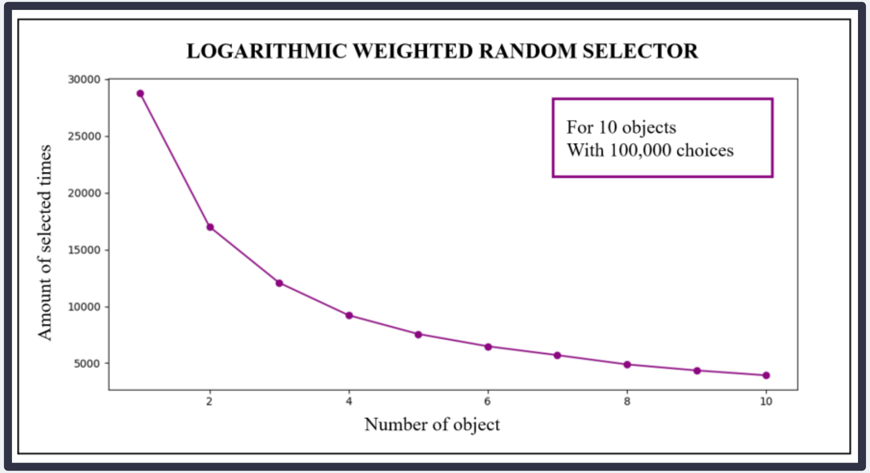
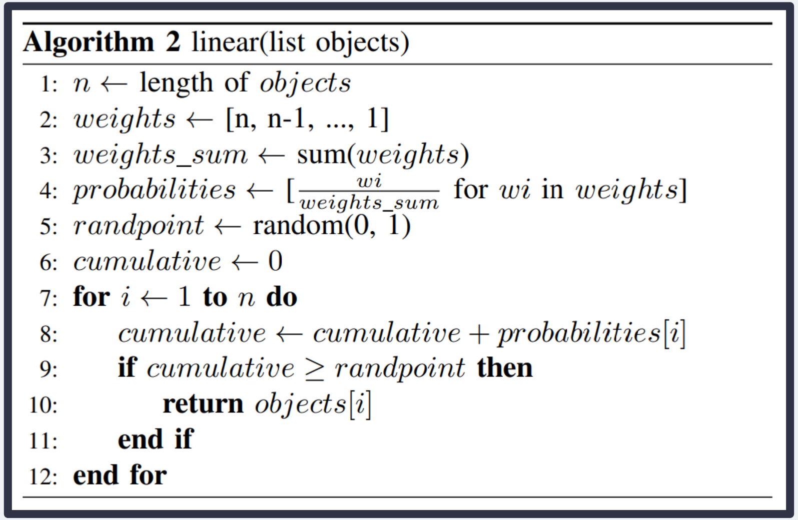


Fig. 6. Experiment using 5 objects and 100 repetitions of LWRS selecting.

Fig. 7. Experiment using 10 objects and 100,000 repetitions of LWRS selecting.

LINEAR WEIGHTING SELECTION



```
Algorithm 2 linear(list objects)
$n \leftarrow$ length of $objects$
$weights \leftarrow$ [n, n-1, ..., 1]
$weights\_sum \leftarrow$ sum($weights$)
$probabilities \leftarrow [\frac{wi}{weights\_sum}$ for $wi$ in $weights]$
$randpoint \leftarrow$ random(0, 1)
$cumulative \leftarrow 0$
for $i \leftarrow 1$ to $n$ do
    $cumulative \leftarrow cumulative + probabilities[i]$
    if $cumulative \geq randpoint$ then
        return $objects[i]$
    end if
end for
```

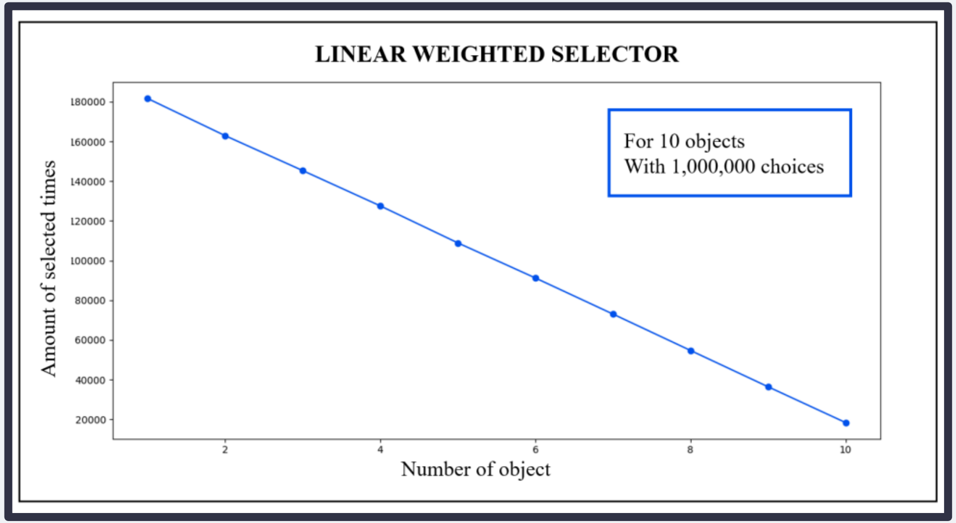
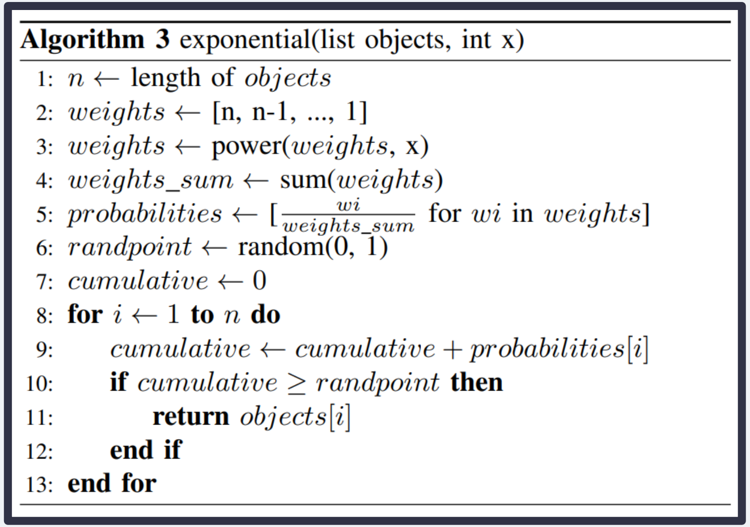


Fig. 8. Pseudocode of Algorithm 2, where the importance weight of each element is given as the result of the reverse index of the element in the list divided by the sum of indexes of the list.

Fig. 9. Experiment in which 1,000,000 repetitions of the Linear algorithm were made, selecting from a list of 10 elements, counting and graphing how many times each object was selected to check the bias towards each one.

EXPONENTIAL WEIGHTING SELECTION



```
Algorithm 3 exponential(list objects, int x)
$n \leftarrow$ length of $objects$
$weights \leftarrow$ [n, n-1, ..., 1]
$weights \leftarrow$ power($weights$, x)
$weights\_sum \leftarrow$ sum($weights$)
$probabilities \leftarrow [\frac{wi}{weights\_sum}$ for $wi$ in $weights]$
$randpoint \leftarrow$ random(0, 1)
$cumulative \leftarrow 0$
for $i \leftarrow 1$ to $n$ do
    $cumulative \leftarrow cumulative + probabilities[i]$
    if $cumulative \geq randpoint$ then
        return $objects[i]$
    end if
end for
```

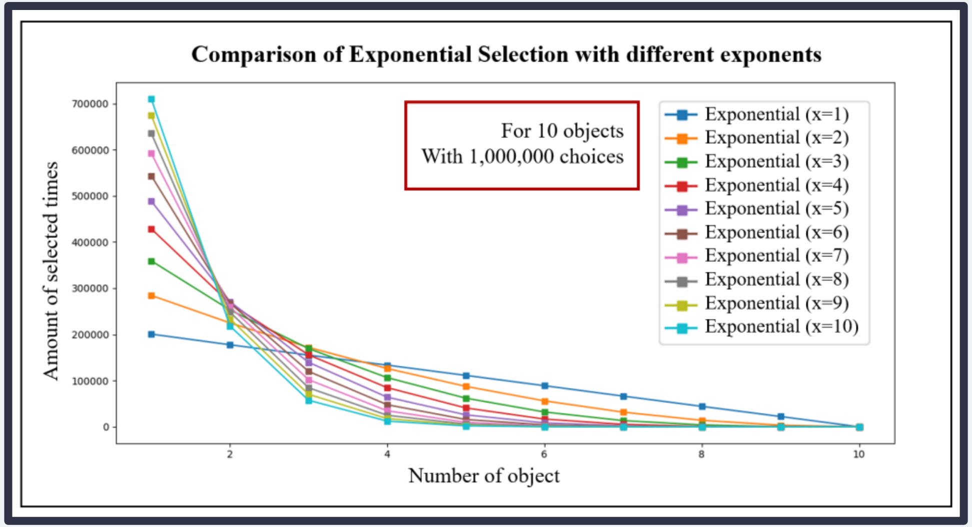


Fig. 10. Pseudocode of Algorithm 3, where the importance weight of each element is given as the result of the inverse index (powered to *x* value) of the element in the list divided by the sum of indexes of the list.

Fig. 11. Experiment in which 1,000,000 repetitions of the Exponential algorithm were made, selecting from a list of 10 elements, counting and graphing how many times each object was selected to check the bias towards each one, for *x* values from 1 to 10.

COMPARISON

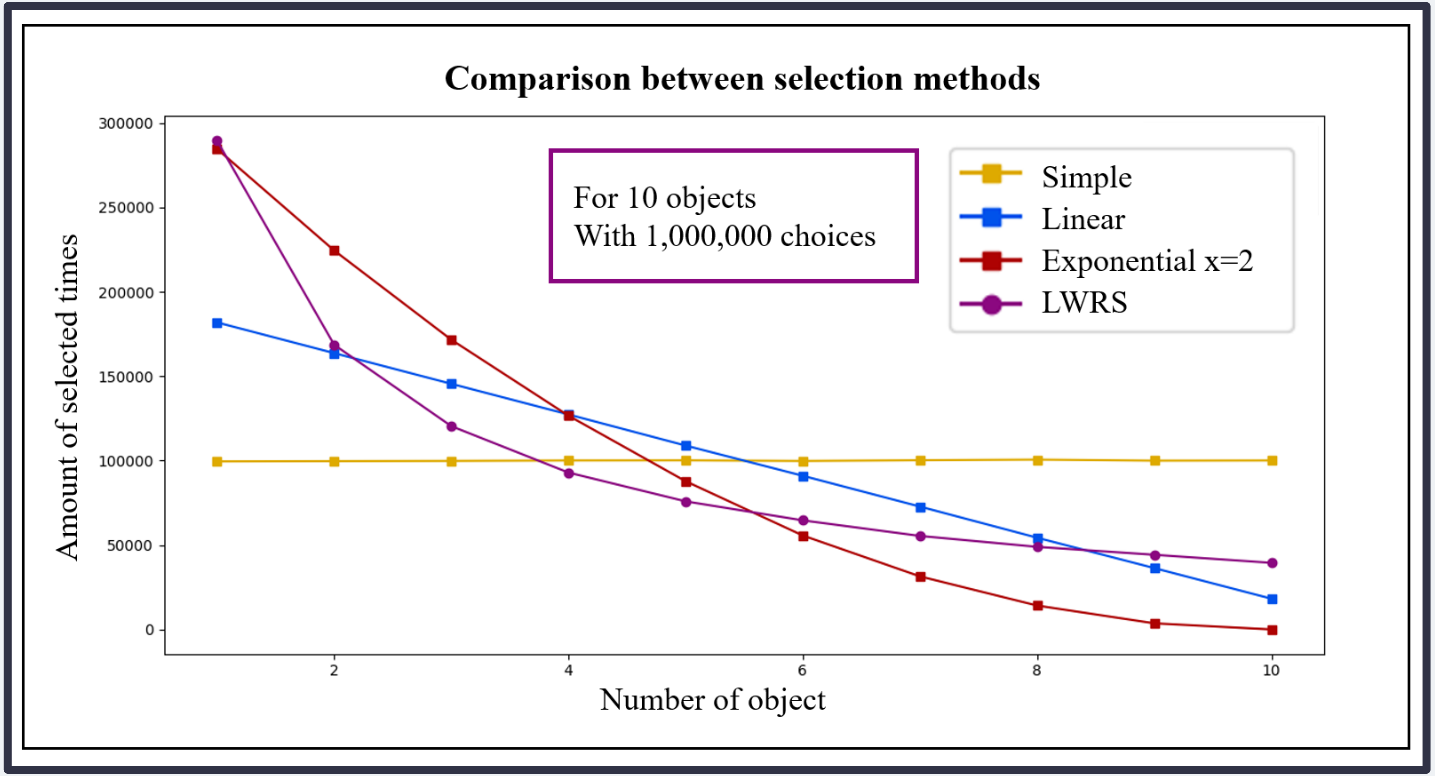


Fig. 12. Comparison between LWRS, Linear Weighted Selection, Exponential Selection with x=2 and Simple Random Selection in experiment with 10 objects and 1,000,00 choices.

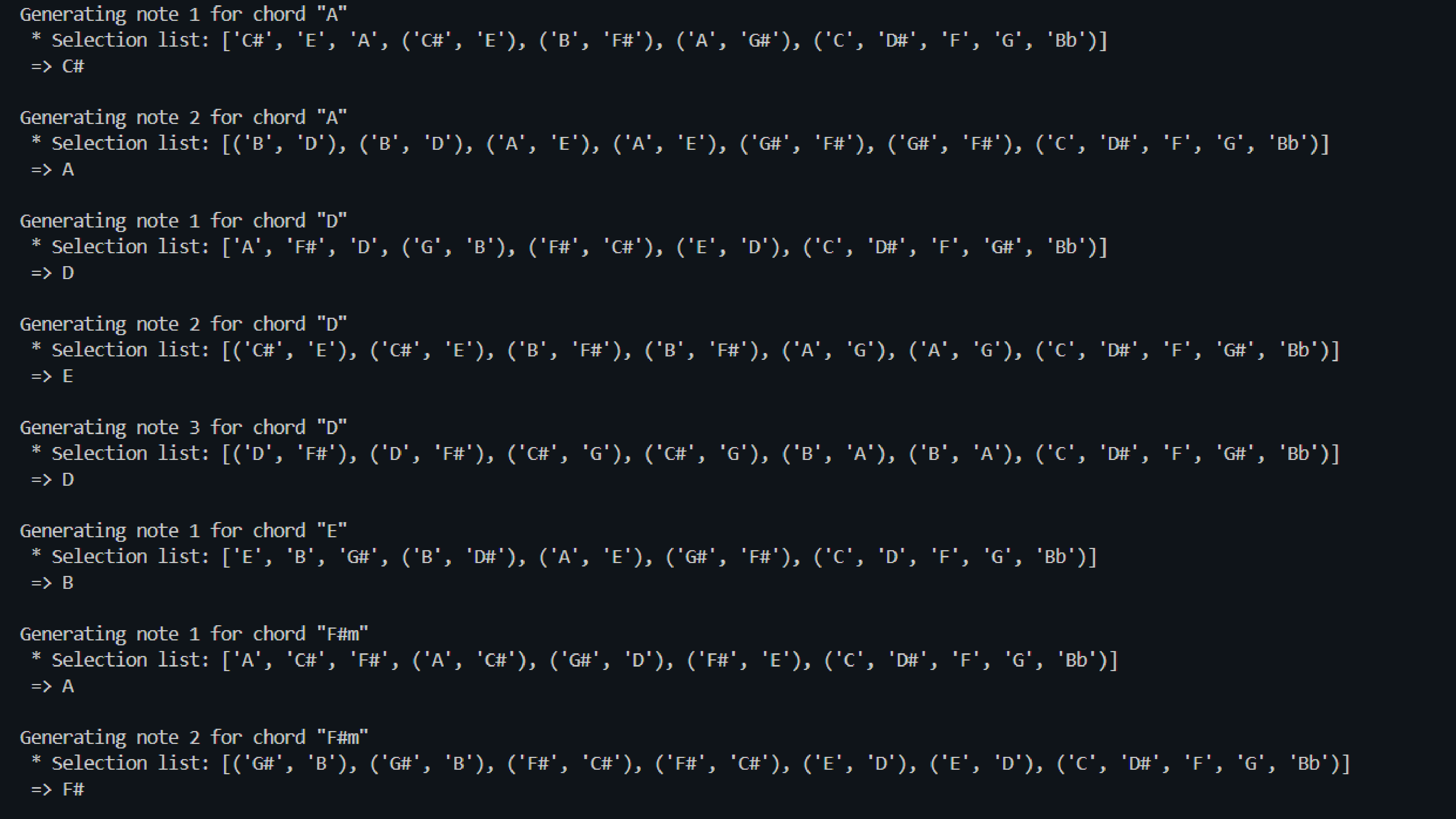
LIMITATIONS

1. **Limited applicability.**
2. **No quantitative comparison.**
3. **Difficult control in critical contexts.**
4. **Risk of unintended outcomes.**

CONCLUSIONS

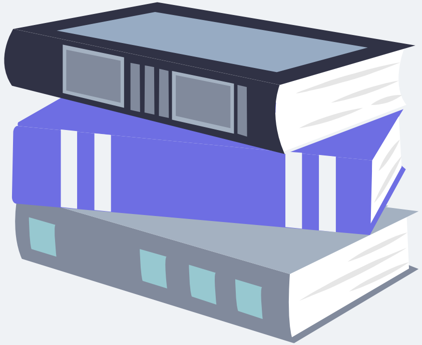
1. **Order-based selection problem adressed.**
2. **No need for hyperparameters.**
3. **Experiments validate bias towards top elements.**
4. **Applicability in diverse contexts like generative models and decisión-making models.**

EXAMPLE OF APPLICATION OF LWRS



```
Generating note 1 for chord "A"
 * Selection list: ['C#', 'E', 'A', ('C#', 'E'), ('B', 'F#'), ('A', 'G#'), ('C', 'D#', 'F', 'G', 'Bb')]
 => C#

Generating note 2 for chord "A"
 * Selection list: [('B', 'D'), ('B', 'D'), ('A', 'E'), ('A', 'E'), ('G#', 'F#'), ('G#', 'F#'), ('C', 'D#', 'F', 'G', 'Bb')]
 => A

Generating note 1 for chord "D"
 * Selection list: ['A', 'F#', 'D', ('G', 'B'), ('F#', 'C#'), ('E', 'D'), ('C', 'D#', 'F', 'G#', 'Bb')]
 => D

Generating note 2 for chord "D"
 * Selection list: [('C#', 'E'), ('C#', 'E'), ('B', 'F#'), ('B', 'F#'), ('A', 'G'), ('A', 'G'), ('C', 'D#', 'F', 'G#', 'Bb')]
 => E

Generating note 3 for chord "D"
 * Selection list: [('D', 'F#'), ('D', 'F#'), ('C#', 'G'), ('C#', 'G'), ('B', 'A'), ('B', 'A'), ('C', 'D#', 'F', 'G#', 'Bb')]
 => D

Generating note 1 for chord "E"
 * Selection list: ['E', 'B', 'G#', ('B', 'D#'), ('A', 'E'), ('G#', 'F#'), ('C', 'D', 'F', 'G', 'Bb')]
 => B

Generating note 1 for chord "F#m"
 * Selection list: ['A', 'C#', 'F#', ('A', 'C#'), ('G#', 'D'), ('F#', 'E'), ('C', 'D#', 'F', 'G', 'Bb')]
 => A

Generating note 2 for chord "F#m"
 * Selection list: [('G#', 'B'), ('G#', 'B'), ('F#', 'C#'), ('F#', 'C#'), ('E', 'D'), ('E', 'D'), ('C', 'D#', 'F', 'G', 'Bb')]
 => F#
```

REFERENCES



1. Burgoyne, J.A., Wild, J., Fujinaga, I. (2013). Compositional Data Analysis of Harmonic Structures in Popular Music. In: Yust, J., Wild, J., Burgoyne, J.A. (eds) Mathematics and Computation in Music. MCM 2013. Lecture Notes in Computer Science (), vol 7937. Springer, Berlin, Heidelberg.
2. Polya, George, and George Pólya. How to solve it: A new aspect of mathematical method. Vol. 34. Princeton university press, 2014.
3. NOVOSEL, Lorraine M. Understanding the Evidence: Sampling and Probability (Random) Sampling Designs. Urologic Nursing, 2023, vol. 43, no 4.
4. Singh, Ajay & Masuku, Micah. (2014). Sampling Techniques and Determination of Sample Size in Applied Statistics Research: An Overview. International Journal of Commerce and Management. 2. 1-22.
5. Cochran, WG (1963): Sampling Techniques, 2nd Ed., New York: John Wiley and Sons, Inc.
6. GREFENSTETTE, John. Rank-based selection. Evolutionary computation, 2000, vol. 1, p. 187-194.

@ARHCODER