The U Manifold and Ontological Differentiation

Pablo García Cuadrillero

Contents

I. Abstract 1

II. Motivation 1

III. Definitions and Properties 2

1. U Manifold 2

2. Ontological Differentiation 2
   2.1. Types . . . 3
      2.1.1. Weak Ontological Differentiation . . . 3
      2.1.2. Strong Ontological Differentiation . . . 5
      2.1.3. Great Ontological Differentiation . . . 6
      2.1.4. Eigen Ontological Differentiation . . . 7
      2.1.5. Others . . . 8

3. Properties 8
   3.1. Topology . . . 8
      3.1.1. Manifold . . . 8
      3.1.2. Non-Euclidean Duality . . . 10
   3.2. Measure . . . 10
   3.3. Connectivity . . . 11

IV. Functions and Order 14

4. Finite Functions 14
   4.1. $f(x) = x + k$ . . . 15
      4.1.1. Results for WOD . . . 16
      4.1.2. Results for SOD . . . 26

5. Recursive Functions 35
   5.1. $f(x) = \frac{1}{x}$ . . . 36
      5.1.1. Results for SOD . . . 36
      5.1.2. Results for WOD . . . 41
   5.2. $f(x) = \sin x$ . . . 45
      5.2.1. Results for WOD . . . 46
      5.2.2. Results for SOD . . . 48

5.3. $f(x) = \cos x$ . . . 51
5.3.1. Results for WOD . . . 51
5.3.2. Results for SOD . . . 55
5.4. $f(x) = x + k$ . . . 59
5.4.1. Results for Decimals . . . 60
5.4.2. Results for Integers . . . 62

6. Randomness **63**
6.1. Results for Unconstrained . . . 65
6.2. Results for Regular Uniqueness . . . 70
6.3. Results for Strict Uniqueness . . . 71
6.4. Results for Weighted . . . 72

V. Applications 73

7. Language Model **73**
7.1. Construction of $\mathbf{U}$ . . . 73
7.2. One vs All Results . . . 74
7.3. All vs All Results . . . 77

8. Physical Model **79**
8.1. QFT Manifold . . . 79
8.2. Gravitation . . . 81

VI. Online Resources 81

Part I.
Abstract

A new manifold notion (**U**) is proposed along with an operation (OD) to calculate distances between points in such manifold. In this work the different properties of the manifold and the operation are shown as well as the results of defining it alongside functions. Finally it is applied on a language model and it is discussed how it could be applied onto physical models.

Part II.
Motivation

The concept of manifold was born out of the need of describing the intuitive idea of space, that is, the distribution of different points and their relations. This idea was developed during the 19th and 20th century specially, it has since then been a central part of many disciplines and theories both in mathematics and physics. While many kinds of manifolds are possible, only a few remain well known and daily used. In this work we propose a manifold which should be a further notion from its regular version.

We will present a manifold, which we call **U**, whose elements are continuously related, every point may be defined according to other points, so every point is as relative as any other. The motivation behind the creation of such a manifold is to represent the inherent relative and related structure of ideas. It is rare, or rather impossible, to have ideas which are not defined and explained through the use of others, be it mathematical, physical or philosophical. The aim of the creation for such a manifold is precisely to give a formulation to this situation.

Since it is also important to be able to operate on manifolds, we will also present here an operation, which we call Ontological Differentiation, in order to understand the relations between the points in **U**, given their inter-defined nature.

Part III.
Definitions and Properties

1. U Manifold

Definition 1. $\mathbf{U}$ *is a set of elements* $\{P_1, ..., P_n\}$ *of size* $n \in \mathbb{N}$*, for which every element* $P_i \in \mathbf{U}$ *is itself a set of size* $m \in \mathbb{N}$ *of elements in* $\mathbf{U}$*, that is* $P_i = \{P_{i_1}, ..., P_{i_m}\}$ *where every* $P_{i_j} \in \mathbf{U}$*.*

2. Ontological Differentiation

Definition 2. *We call* R *the read function, an homomorphism such that* $\forall\, P_i \in \mathbf{U}$, $R^n(P_i) : \mathbf{U} \to \mathbf{U}$*, whose purpose is to expand the sets* n *times.*

That is, if we have as input some element $P_i \in \mathbf{U}$ whose content is given as $P_i = \{P_{i_1}, ..., P_{i_m}\}$, the read function R^1 will produce an output: $R^1(P_i) = P_{i_1}, ..., P_{i_m}$. The n parameter specifies how many times this expansion operation is to be carried out, being $n = 0$ just the reading $R^0(P_i) = P_i$. For instance, $R^2(P_i) = R^1(P_{i_1}), ..., R^1(P_{i_m})$. This can be generalized as $R^n(P_i) = R^{n-1}(P_{i_1}), ..., R^{n-1}(P_{i_m})$, and furthermore as $R^n(P_i) = R^{n-1}(R^{n-2}(...R^0(P_i)...))$.

Definition 3. *Let us have a subset* $\mathcal{A}$ *of* $\mathbf{U}$ *of size* k*. Let us call* $\mathcal{B}$ *another subset of* $\mathbf{U}$ *which contains* ω *orders of the read function operated on* $\mathcal{A}$*. That is, if we have that* $\mathcal{A} = \{P_1, ..., P_k\}$*, then we have that* $\mathcal{B} = \{R^0(P_1), ..., R^0(P_k), ..., R^\omega(P_1), ..., R^\omega(P_k)\}$*.*

In order to indicate up to what order of R^n in $\mathcal{A}$ we refer to, we write a sub-index n in $\mathcal{A}_n \subseteq \mathbf{U}$, therefore $\mathcal{A}_n \subseteq \mathcal{A}$, and of course we have that $\mathcal{A}_\omega = \mathcal{A}$.

Definition 4. *A cancellation rule* κ *determines how an element* $\alpha \in R^n(P_s) \in \mathcal{B}$ *gets cancelled.*

Definition 5. *A termination rule* τ *determines, based on the results of applying a cancellation rule* κ*, the value of* ω*.*

Definition 6. *A cancellation function equipped with a cancellation and a termination rule is expressed as* $F_{\kappa\tau}$*. Given some* $\mathcal{A} \subseteq \mathbf{U}$ *of size* k*, based on* κ *and* τ*,* $F_{\kappa\tau}$ *will act on* $\mathcal{A}$ *determining what elements get cancelled and the value of* ω*. Formally,* $F_{\kappa\tau} : \mathbf{U}^{k\omega} \to \mathbf{U}$*.*

Definition 7. *The result quantification of a cancellation function is given by* $\sum_{n=0}^{\omega} nF_{\kappa\tau}(\mathcal{A}_n)$*.*

Definition 8. *The distance between the sets in some subset* $\mathcal{A}$ *of* $\mathbf{U}$ *is given by an Ontological Differentiation (OD), which is a cancellation function* $F_{\kappa\tau}$ *with its result quantification* $\sum_{n=0}^{\omega} nF_{\kappa\tau}(\mathcal{A}_n)$*, and it is expressed as* $OD(\mathcal{A})$*.*

2.1. Types

In this subsection we will present some types of Ontological Differentiations which seem to the author to be the most intuitively straight forward and which will be used later on applied cases. In order to give examples, we will first present the Sample Set, a set example of **U** for us to illustrate the different OD's:

Example 1. *The Sample Set which we will use for some of the examples in this work:*

1: [2, 6]
2: [6, 1]
3: [4, 5]
4: [5, 3]
5: [3, 4]
6: [1, 2]
7: [6, 8]
8: [9, 10]
9: [10, 11]
10: [11, 12]
11: [12, 8]
12: [10, 9]
13: [14, 15]
14: [15, 13]
15: [13, 14]
16: [17, 18]
17: [18, 16]
18: [16, 17]
19: [18, 20]
20: [21, 22]
21: [22, 19]
22: [21, 20]
24: [22, 23]

2.1.1. Weak Ontological Differentiation

The Weak Ontological Differentiation (WOD) is the most simple and less restrictive type of the here to be presented. It is given by the following cancellation and termination rules.

Definition 9. *The WOD cancellation rule is expressed as: If $\alpha \in R^n(P_s) \wedge \beta \in R^h(P_r) \wedge \alpha = \beta$, where $n, h \in (0, \omega)$ and $r, s \in (1, k)$, then $\alpha \in F_{\kappa\tau}(\mathcal{A})$.*

Definition 10. *The WOD termination rule is expressed as: ω is reached when $\forall \alpha \in R^n(P_s)\, \exists\, \beta \in R^h(P_r)$, where $n, h \in (0, \omega)$ and $r, s \in (1, k)$, such that $\alpha = \beta$.*

This means that in WOD, elements get cancelled when they are repeated at least once in any $R^n(P_s)$, that is, any element that appears at least twice anywhere, at any order of the read function, across any set in $\mathcal{A}$, will get cancelled. The WOD terminates when there is at least one $R^n(P_s) \in \mathcal{B}$ which has all of its elements cancelled.

Let us illustrate this with an example from the Sample Set:

Example 2. $WOD(16, 20)$*:*

Level 0: *16; 20*

Level 1: *[17, 18]; [21, 22]*
16; 20
17, 18; 21, 22

Level 2: *[18, 16, 16, 17]; [22, 19, 21, 20]*
16(c); 20(c)
17(c), 18(c); 21(c), 22(c)
18(c), 16(c), 16(c), 17(c); 22(c), 19, 21(c), 20(c)

Terminating at Level 2 because one side is fully canceled.

Total distance: 18.

Let us explain the format a bit so it is easier to understand. The levels here refer to the order of the read function as one can easily identify. What we have done after every level is update every cancellation status of every level (every cancelled element is marked with "(c)"), until we have found one side of the differentiated sets (which we separate by the symbol ";") to be fully cancelled. We then count how many cancelled elements there are for every level and we multiply that number by the level number.

The purpose of this format is to offer a visual aid to the process, but we can certainly be more effective and optimized in our format. Let us call U_{nm} the set of uncancelled elements of level n and of differentiated set/side m, and R_{nm} the set of repeated (cancelled) elements of level n and of differentiated set/side m. It will stop when any U_{nm} becomes empty after updating after a new level of the read function, which is the exact equivalent of stopping when one side of any level has all of its elements fully cancelled. We place the number of times the element is repeated in its respective R_{nm} so the total distance can be easily calculated. The result shown is the total update when the calculation is terminated, not the step by step update as we had in the previous format. This optimized version is then:

Example 3. $WOD(16, 20)$*:*

Level 0:
Side 1 - U_0_1: {}, R_0_1: {'16': 1}

Side 2 - U_0_2: {}, R_0_2: {'20': 1}

Level 1:
Side 1 - U_1_1: {}, R_1_1: {'18': 1, '17': 1}
Side 2 - U_1_2: {}, R_1_2: {'22': 1, '21': 1}

Level 2:
Side 1 - U_2_1: {}, R_2_1: {'16': 2, '18': 1, '17': 1}
Side 2 - U_2_2: {'19'}, R_2_2: {'22': 1, '20': 1, '21': 1}

Termination condition met at Level 2 *because U_0_1, U_0_2, U_1_1, U_1_2, U_2_1 became empty.*

Total distance: 18

2.1.2. Strong Ontological Differentiation

The Strong Ontological Differentiation (SOD) is a stricter version of the WOD, while the termination rule is the same, the cancellation rule only applies to elements that are not found in the same differentiating set/side. So if two elements are found to be repeated across any level of the same side, they will not be cancelled unless they are also found on the other side.

Definition 11. *The SOD cancellation rule is expressed as: If* $\alpha \in R^n(P_s) \wedge \beta \in R^h(P_r) \wedge \alpha = \beta$*, where* $n, h \in (0, \omega)$ *and* $r, s \in (1, k)$ *but* $r \neq s$*, then* $\alpha \in F_{\kappa\tau}(\mathcal{A})$*.*

Definition 12. *The SOD termination rule is expressed as:* ω *is reached when* $\forall \alpha \in R^n(P_s)\, \exists\, \beta \in R^h(P_r)$*, where* $n, h \in (0, \omega)$ *and* $r, s \in (1, k)$ *but* $r \neq s$*, such that* $\alpha = \beta$*.*

Example 4. $SOD(16, 20)$*:*

Level 0: *16; 20*

Level 1: *[17, 18]; [21, 22]*
16; 20
17, 18; 21, 22

Level 2: *[18, 16, 16, 17]; [22, 19, 21, 20]*
16; 20
17, 18; 21, 22
18, 16, 16, 17; 22, 19, 21, 20

Level 3: *[16, 17, 17, 18, 17, 18, 18, 16]; [21, 20, 18, 20, 22, 19, 21, 22]*
16; 20
17, 18(c); 21, 22
18(c), 16, 16, 17; 22, 19, 21, 20
16, 17, 17, 18(c), 17, 18(c), 18(c), 16; 21, 20, 18(c), 20, 22, 19, 21, 22

Level 4: *[17, 18, 18, 16, 18, 16, 16, 17, 18, 16, 16, 17, 16, 17, 17, 18]; [22, 19, 21, 22, 16, 17, 21, 22, 21, 20, 18, 20, 22, 19, 21, 20]*
16(c); 20
17(c), 18(c); 21, 22
18(c), 16(c), 16(c), 17(c); 22, 19, 21, 20
16(c), 17(c), 17(c), 18(c), 17(c), 18(c), 18(c), 16(c); 21, 20, 18(c), 20, 22, 19, 21, 22
17(c), 18(c), 18(c), 16(c), 18(c), 16(c), 16(c), 17(c), 18(c), 16(c), 16(c), 17(c), 16(c), 17(c), 17(c), 18(c); 22, 19, 21, 22, 16(c), 17(c), 21, 22, 21, 20, 18(c), 20, 22, 19, 21, 20

Terminating at Level 4 because one side is fully canceled.

Total distance: 113.

As for the optimized format of the SOD calculation we have:

Example 5. $SOD(16, 20)$*:*

Level 0:
Side 1 - U_0_1: {}, R_0_1: {'16': 1}
Side 2 - U_0_2: {'20'}, R_0_2: {}

Level 1:
Side 1 - U_1_1: {}, R_1_1: {'18': 1, '17': 1}
Side 2 - U_1_2: {'21', '22'}, R_1_2: {}

Level 2:
Side 1 - U_2_1: {}, R_2_1: {'18': 1, '16': 2, '17': 1}
Side 2 - U_2_2: {'22', '19', '21', '20'}, R_2_2: {}

Level 3:
Side 1 - U_3_1: {}, R_3_1: {'18': 3, '16': 2, '17': 3}
Side 2 - U_3_2: {'21', '20', '22', '19'}, R_3_2: {'18': 1}

Level 4:
Side 1 - U_4_1: {}, R_4_1: {'16': 6, '17': 5, '18': 5}
Side 2 - U_4_2: {'22', '19', '21', '20'}, R_4_2: {'16': 1, '17': 1, '18': 1}

Termination condition met at Level 4 *because U_0_1, U_1_1, U_2_1, U_3_1, U_4_1 became empty.*

Total distance: 113

2.1.3. Great Ontological Differentiation

The Great Ontological Differentiation (GOD) is a more extensive approach than the past two types. The cancellation rule is the same as in WOD, but the termination

rule requires that the calculation stop when no new element appears with a new level of the read function, that is, every single future read function will produce repeated elements. This OD will be used later on to detect those those sets which can loop towards infinity in an OD (it will depend if **U** is finite or infinite size, or if it's SOD or WOD), and it will be also used to find all the related sets to a set.

Definition 13. *The GOD cancellation rule is expressed as: If* $\alpha \in R^n(P_s) \wedge \beta \in R^h(P_r) \wedge \alpha = \beta$*, where* $n, h \in (0, \omega)$ *and* $r, s \in (1, k)$*, then* $\alpha \in F_{\kappa\tau}(\mathcal{A})$*.*

Definition 14. *The GOD termination rule is expressed as:* $\forall P_i \in \mathcal{A} \forall \alpha \in R^n(P_i) \exists \beta \in R^{n-1}(R^{n-2}(...R^0(P_i)...))$ *such that* $\alpha = \beta$*.*

Example 6. $GOD(16, 20)$*:*

Level 0:
Already Existing Elements - Repeated: ; Uncanceled:
New Elements in This Iteration: 16, 20
Updated Repeated Elements:
Updated Uncanceled Elements: 16, 20

Level 1:
Already Existing Elements - Repeated: ; Uncanceled: 16, 20
New Elements in This Iteration: 17, 18
Updated Repeated Elements:
Updated Uncanceled Elements: 16, 17, 20, 18

Level 2:
Already Existing Elements - Repeated: ; Uncanceled: 16, 17, 20, 18
New Elements in This Iteration:
Updated Repeated Elements: 16, 17, 18
Updated Uncanceled Elements: 20

Stopping at Level 2 because no new elements were found.

As we can see this format is different from the previous ones, as it is adapted to be more explicit and focused on the appearing of new elements, essential factor of this OD, that is why the total distance is not shown, but it certainly could be.

2.1.4. Eigen Ontological Differentiation

An Eigen Ontological Differentiation (EOD) is not defined by any cancellation or termination rule, but by the size of the $\mathcal{A}$ input, which is 1. EOD performs an OD of one set onto itself. There can be Weak, Strong or Great EOD's, we show here a $WEOD$:

Example 7. $WEOD(16)$*:*

Level 0: [16]

Level 1: [17, 18]
16
17, 18

Level 2: [18, 16, 16, 17]
16(c)
17(c), 18(c)
18(c), 16(c), 16(c), 17(c)

Stopping at Level 2 because the set is fully canceled.
Total distance: 10

An *EOD* will be useful later on to explain the concept of measure in **U**.

2.1.5. Others

As one can easily imagine, there can be as many variations of the rules as one wishes, mixing types, alternating them, perturb them, add to them and so on. The scope of this work is not to show a wide spectrum of the infinite possibilities, but to present the general concept and the most intuitive versions of it along their applications.

3. Properties

3.1. Topology

3.1.1. Manifold

In this subsection we will show that **U** is a topological manifold and can therefore be treated as such.

First, we will establish the concept of neighbourhood in topology, that is, a set of points that are within a distance k from one same point, so each point belongs to every one of its neighbourhoods. Here the analogous notion for a neighbourhood of P_i is presented:

Definition 15. *Neighbourhood of P_i:*

$$\mathrm{Nbd_k}P_i = \{P_j : OD(P_i, P_j) \leq k \in \mathbb{N}\}. \tag{1}$$

Now we will state the manifold definition and prove each of these statements for our case.

Definition 16. *A topological manifold is a second-countable, Hausdorff, locally Euclidean space. It has dimension n if it is locally Euclidean of dimension n.*

Definition 17. *A Hausdorff space is a space in which for any two distinct points, there exists neighbourhoods of each which are disjoint from each other.*

Lemma 8. **U** *is a Hausdorff space.*

Proof. We must show that for any two points P_i and P_j, there are Nbd's of them which are disjoint. This is easily proved by Nbd_0 of both P_i and P_j: $\mathrm{Nbd}_0 P^i = P_i$ and $\mathrm{Nbd}_0 P_j = P_j$, since for any P_x, if $D(P_x, P_y) = 0$, then $P_x = P_y$. Therefore, since $P_i \neq P_j$, $\mathrm{Nbd}_0 P_i \cap \mathrm{Nbd}_0 P_j = \emptyset$. □

This states that P's are separated.

Definition 18. *A second-countable space is a topological space with a countable base. T is a second-countable space if there is some countable collection $\mathcal{C} = \{C_i\}_{i=1}^{\infty}$ of open subsets of T such that any open subset of T can be written as a union of elements of some subfamily of $\mathcal{U}$.*

Lemma 9. **U** *is a second-countable space.*

Proof. We have to show that **U** is a second-countable space. Let us call $\{P_i\}_{i=1}^{\infty}$ our countable collection of open subsets of **U**. They are open because for any $P_i = \{P_{i_1}, ..., P_{i_m}\}$, we have that its elements $\{P_{i_1}, ..., P_{i_m}\}$ (of any level from the read function), its points they are all within $\mathrm{Nbd}_1 P_i$ (or for any k for $\mathrm{Nbd}_{\mathrm{k}} P_i$ if we want to take another level). Now, let us take some open subset of **U**, let us take some $\mathcal{A}$ as we have been doing, that is $\mathcal{A} = \{P_1, ..., P_k\}$ for some k. We have that $\mathcal{A}$ is a union of the elements $\{P_1, ..., P_k\}$, which all of them are elements in $\{P_i\}_{i=1}^{\infty}$. Therefore **U** must be second-countable. □

Definition 19. *A locally Euclidean space is a topological space $\mathcal{M}$ of dimension n if every point p in $\mathcal{M}$ has a neighbourhood $\mathcal{N}$ such that there is a homeomorphism ϕ from $\mathcal{N}$ onto an open subset of $\mathbb{R}^n$.*

Lemma 10. **U** *is a locally Euclidean space.*

Proof. We have to show that **U** is locally Euclidean. For that we have to show that every point P_i in **U** has a neighbourhood $\mathrm{Nbd}_{\mathrm{k}} P_i$ (for some $k \in \mathbb{N}$) which is homemorphic to an open subset of $\mathbb{R}^n$. Let us have that $\mathrm{Nbd}_{\mathrm{k}} P_i = \{P_1, ..., P_s\}$ for some $s \in \mathbb{N}$. Let us call ϕ the association of every subindex of the members in the neighbourhood with a natural number, up to s: $\{P_{\phi(1)}, ..., P_{\phi(s)}\}$. Since we have mapped with ϕ every member of the neighbourhood with a natural number and $\mathbb{N} \subset \mathbb{R}$, we have that **U** is locally Euclidean. □

Corollary 10.1. **U** *is a topological manifold.*

There is one rather remarkable aspect that can be obtained from mapping elements of **U** to $\mathbb{N}$. As we will see later on, in the section where we apply OD to a language model, a set of words or letters can be used as a **U** manifold. Let us imagine we want to show that this set up is also locally Euclidean. Let us have a function ϕ which maps every letter of the Latin alphabet to a base of ten with the exponent being the place the letter occupies in the alphabet (e.g. Owl=$P^{10^{15}10^{23}10^{12}}$, where the numbers do not represent a multiplication but just an abbreviation so $10^2 10^3 = 1001000$). Since numbers, any number, is a word made out of letters (e.g. 32,5 = thirty two comma five), we have that ϕ can map any $\alpha \in \mathbb{R}$ onto some $\beta \in \mathbb{N}$.

3.1.2. Non-Euclidean Duality

Let us take from the Sample Set the sets 16, 20 and 21. The triangle inequality states that $d(p_1, p_3) \leq d(p_1, p_2) + d(p_2, p_3)$, so we should have that $OD(16, 20) \leq OD(16, 21) + OD(21, 20)$. If we take the SOD of each and we substitute into the inequality we find that $113 \leq 42 + 3$, which clearly breaks the inequality. Therefore, $\mathbf{U}$ can be a manifold which is locally Euclidean, and at the same time it can be non-Euclidean and have their distances behave in probability-like relations.

3.2. Measure

We say a set $A \subseteq \mathbf{R}$ is Lebesgue measurable if for any other set $B \subseteq \mathbf{R}$, the following condition applies:

$$m(B) = m(B \cap A) + m(B \cap A^c). \tag{2}$$

Let us have our $\mathbf{U}$ to be the sets formed by $P_i = \{P_i + 1, P_i + 2\}$ (which we will later see that this is ordered after $f(x) = x + 1$, since $x = \{f(x), f(f(x))\}$. Let us take from those just $1 = [2, 3]$ and $2 = [3, 4]$, which we can denote as A_1 and A_2. Let us see if the Lebesgue measure condition is fulfilled for A_2:

$$\begin{aligned} m(A_2) &= m(A_2 \cap A_1) + m(A_2 \cap A_1^c) \\ &= m([3, 4] \cap [2, 3]) + m([3, 4] \cap ((-\infty, 2) \cup (-\infty, 3))) \\ &= m(3) + m([3, 4]). \end{aligned} \tag{3}$$

Up to this stage, we can say that it is Lebesgue measurable, since we will have:

$$m(A_2) = m([3, 4]) = 4 - 3 = 1 = m(3) + m((3, 4]) = 0 + 1. \tag{4}$$

However, we also know that these sets are interdefined, that is, their definition can go beyond the first level from the read function. That means, initially, that since $3 = [4, 5]$ we would have:

$$m([3, 4]) = m(3) + m([3, 4]) = m([4, 5]) + m((3, 4]) = 2. \tag{5}$$

At this second stage the Lebesgue measure is not fulfilled, therefore $\mathbf{U}$ would not be Lebesgue measurable. This second stage directly states that: $m(2) = m(2) + m(3)$. This would not necessarily have to end here, we can also further develop the expression:

$$\begin{aligned} m(2) &= m([4, 5]) + m((3, 4]) \\ &= m([5, 6] \cup [6, 7]) + m([4, 5] \cup [5, 6]) \\ &= m([5, 7]) + m([4, 6]) \\ &\text{etc.} \end{aligned} \tag{6}$$

since the measure of $m([a, b, c]) = m([a, c])$. So at this stage we would have that the measure is of sets given by the ordering $f(x) = x + 2$ instead of $f(x) = x + 1$, no

longer the same function, and as one keeps using the read function (vaguely noted here as "etc"), one would be raising $f(x) = x + k$ one by one.

So for A_n and A_{n-1} we have that:

- First stage: $m(A_n) = m(A_n \cap A_{n-1}) + m(A_n) \cap A_{n-1}^c$.
- Second stage: $m(A_n) = m(A_n) + m(A_{n+1})$.
- Third stage: Measure of sets given by increasing function of $y = x + k$.

And of course we have the zeroth stage, which is the trivial case where we have that the measure of any point $m(a) = 0$, so $m(A_n) = 0$. These so called stages here are the direct implication of the intrinsic read function.

We can also selectively alternate different stages in the same expression, arriving at situations like:

$$\begin{aligned} m([3,4]) &= m(3) + m([3,4]) \\ &= m([4,5]) + m([4,5] \cup [5,6]) \\ &= m([4,5]) + m([4,5,6]) \\ &= m([4,5]) + m([4,5] \cup 6) \\ &= m([4,5]) + m([4,5] \cup [7,8]) \end{aligned} \tag{7}$$

So if we decided to stop here, we could have the expression for the measure of any A_n and A_{n-1} as:

$$m(A_n) = m(A_{n+1}) + m(A_{n+1} \cup A_{n+4}) \tag{8}$$

All these situations here described for A_n and A_{n-1} can be generalized for A_n and A_{n-k}, therefore we have an infinite spectacle of possible options for measures of a single set in $\mathbf{U}$, this means that $\mathbf{U}$ can be chosen to be a "normal" and Lebesgue measurable if we just stay with first level given by the read function, or it can be chosen not to be and display any kind of measure.

A more solid approach to calculate the measure of a set in $\mathbf{U}$ is to calculate its EOD, since it aims more appropriately at the question behind measure, which can be put to be "Given the elements of a set, where does it finish and what size is that from its beginning?". An $SEOD$, like any SOD, can either finish or not finish (depending on its $GEOD$ and if it is a finite or infinite size $\mathbf{U}$). An $SEOD$ which does not finish has infinite size.

3.3. Connectivity

The elements in $\mathbf{U}$ are defined, or better said, they are interdefined, its interdefinition is a connection between sets. Its connectivity is therefore an interesting matter

to study.

The connectivity among sets in **U** is a consequence of the fulfillment or violation of any of two conditions:

Condition 1: Every element of a set is a set itself (contains other elements): $\forall \alpha \in P_i \, \exists \, P_j \in \mathbf{U}$ such that $\alpha = P_j$.

Condition 2: Every set is itself an element of a different set: $\forall P_i \in \mathbf{U} \, \exists \, P_j \in \mathbf{U}$ such that $P_i \in \ P_j$.

Options:

1.- Fulfillment of both conditions, which is the same as fulfilling only Condition 2.
2.- Fulfillment of Condition 1, but not of Condition 2. OD does not work for those sets which don't fulfill condition 2. We can identify them, and then stop the operation, or it will either produce an error or run forever, applying infinitely the read function on a P_i which shows nothing.
3.- Violation of both conditions, which is the same as violation of Condition 1. In this case there will be sets for which OD does not work either, it will just produce blankness. We can identify these operations beforehand by identifying said sets, or we can let them grow to infinity.

We then give the following terminology to the type of sets which violate some condition:
- Unicularity: A set which violates Condition 2, it contains elements but it is itself not contained anywhere.
- Phantom: A set which violates Condition 1, and therefore condition 2, it is contained in set or sets, but it does not contain other elements, other sets.

Now let us define some more notions to study the connectivity of **U**:

Definition 20. *The Island Finder Operator $I(P_i)$ takes an element P_1 from* **U***, it performs a Great Eigen Ontological Differentiation (GEOD) on it, and every P_i produced in that GEOD constitutes the set $GEOD(P_1)$. Then for every element $\alpha \in GEOD(P_1)$, it picks those $P_i \in R^1((GEOD(P_1))^c)$, where $(GEOD(P_1))^c$ is the complement of $GEOD(P_1)$, such that $\alpha \in P_i$. Let us have the set of those which fulfill the condition to be $\{P_2, ..., P_k\}$. Then it carries the following operation $\underset{i=2}{\overset{k}{I}}(P_i)$ iteration after iteration until the following condition is met: $\nexists \, \alpha \in GEOD(P_i)$ such that $\alpha \in R^1((GEOD(P_i))^c)$.*

Definition 21. *The set which contains all the elements from the initial $GEOD(P_1)$ until the final one $GEOD(P_f)$ when $I(P_i)$ finishes, is called an Island.*

Definition 22. *The SOD-able Condition is defined as: Let $\mathcal{A}$ be a subset of* **U** *as usual, let n be the maximum level (R^n) that $GOD(\mathcal{A})$ takes to finish, if $SOD(\mathcal{A})$ finishes within R^n, then $\mathcal{A}$ is SOD-able.*

Definition 23. *A Sub-Island is a subset of an Island whose elements fulfill the SOD-able Condition.*

The concept of an Island refers to those sets that are part of other sets, it establishes a "walkable" relation between them, that is, one can walk through some *OD* between them, they are reachable within some *OD* process. However, if one of the elements in an Island is a unicularity, the unicularity can walk to the non-unicularity elements, but not otherwise. On the opposite case, if a phantom is present in an Island, the other elements can walk to the phantom (and stop there), but not otherwise.

A Sub-Island is the set of elements in an Island which can be performed an *SOD* among them. If Islands are the most basic notion of connection between sets, Sub-Islands are so far the strictest.

This is of course for a **U** of finite size, if we were to consider a **U** of infinite size, then even if there were no unicularities or phantoms, many *OD*'s of well-behaved P_i's would never finish since its *GEOD* would never stop, for the read function is always adding up new elements after every level (as we will see in the recursive functions sub-section).

Example 11. *Let us take the Sample Set to investigate its connectivity.*
Islands found:

- ***Island 1**: {1, 2, 6, 7, 8, 9, 10, 11, 12}*
- ***Island 2**: {3, 4, 5}*
- ***Island 3**: {13, 14, 15}*
- ***Island 4**: {16, 17, 18, 19, 20, 21, 22, 23, 24}*

Sub-Islands in Island 1:

- ***Sub-Island 1**: {7, 8, 9, 10, 11, 12}*
  - *Unicularities in this sub-island: {7}*
- ***Sub-Island 2**: {1, 2, 6, 7}*
  - *Unicularities in this sub-island: {7}*

Sub-Islands in Island 2:

- ***Sub-Island 1**: {3, 4, 5}*

Sub-Islands in Island 3:

- ***Sub-Island 1**: {13, 14, 15}*

Sub-Islands in Island 4:

- ***Sub-Island 1**: {16, 17, 18, 19, 20, 21, 22, 24}*
  - *Unicularities in this sub-island: {24}*
- ***Sub-Island 2**: {23, 24}*
  - *Phantoms in this sub-island: {23}*
  - *Unicularities in this sub-island: {24}*

Part IV. Functions and Order

In this part we will deal with how the order or structure of the sets/elements in **U** can be arranged and what role some regular functions play in it, as well as randomness. We will then see how OD can operate on functions, since a **U** arranged according to some function $f(x)$ is just a set of its values. Let us first define what order is:

Definition 24. *For some $x_{0_0} \in \mathbf{U}$, for some function f, the order of its elements $x_{0_0} = [x_{0_1}, ..., x_{0_k}]$ for some $k \in \mathbb{N}$ is given by a structure S: $\{S(f(x_{0_i}))\}_{i=1}^{k}$.*

Here we have used x instead of P in the notation just so it is visually more familiar to what we are used to when describing functions.

Example 12. *Let us have the function $f = x + 1$ and the structure $S(f(x_{0_{n-1}})) = x_{0_n}$, then $x_{0_0} = [x_{0_1}, ..., x_{0_k}]$ will be given by $x_0 = [f(x_{0_0}), .., f(x_{0_{k-1}})]$. So if we take $x_{0_0} = 1$ then we have $1 = [2, 3, 4, ...]$ up to size k.*

4. Finite Functions

In this sub-section we will deal with a finite size **U** and functions which will consider this finiteness into its calculation. The fact that our set **U** is finite imposes a non-recursive nature into the operations, making it impossible to operate in $\mathbb{R}$ (which we will do when we get to the recursive functions sub-section).

For a same function $f(x) = x + k$, we will select just the one following structure S:

$$x_{0_n} = \begin{cases} x_{0_0} + 1 & \text{if } n = 1 \\ f(x_{0_{n-1}}) & \text{if } n \neq 1. \end{cases} \tag{9}$$

Let us see it in the following example:

Example 13. *For a* **U** *of size 10, with elements of size 3, for a function* $f(x) = x+2$*, and the above described structure* S*, we would have the following sets:*

Set 0: [1, 3, 5]
Set 1: [2, 4, 6]
Set 2: [3, 5, 7]
Set 3: [4, 6, 8]
Set 4: [5, 7, 9]
Set 5: [6, 8, 0]
Set 6: [7, 9, 1]
Set 7: [8, 0, 2]
Set 8: [9, 1, 3]
Set 9: [0, 2, 4]

As one can easily see, given the finiteness of **U**, when the function result would surpass the maximum set number, it outputs what the result would be taking the sets as a continuation (a circle if you will). So we have that Set $8 : [9, 1, 3]$ where 9 is just $9 + 1$, 1 is just $9 + 2$ (the continuation goes as $9, 0, 1$) and of course 3 is just $1 + 2$.

4.1. $f(x) = x + k$

We will investigate the OD's applied to a finite **U**, ordered for the given structure S, and the function $f(x) = x + k$, having as variable parameters: k in the function, the size of **U** and the size of its elements.

In the following results, the tag of "one vs all" stands for the first set differentiated vs the rest, i.e. $OD(1, x)\forall x \in \mathbf{U}$. The tag "all vs all" stands for the differentiation of each set against the rest. The different values of k will be the range k-range= $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ and for the number of elements num elements= $[2, 3, 4, 5, 10]$. As one will see, we have used a fixed size of num sets (50), if we were to explore for that variable in this results sections the results would be too lengthy, the reader is nonetheless invited to explore it herself, running and plotting the code provided in the repository.

4.1.1. Results for WOD

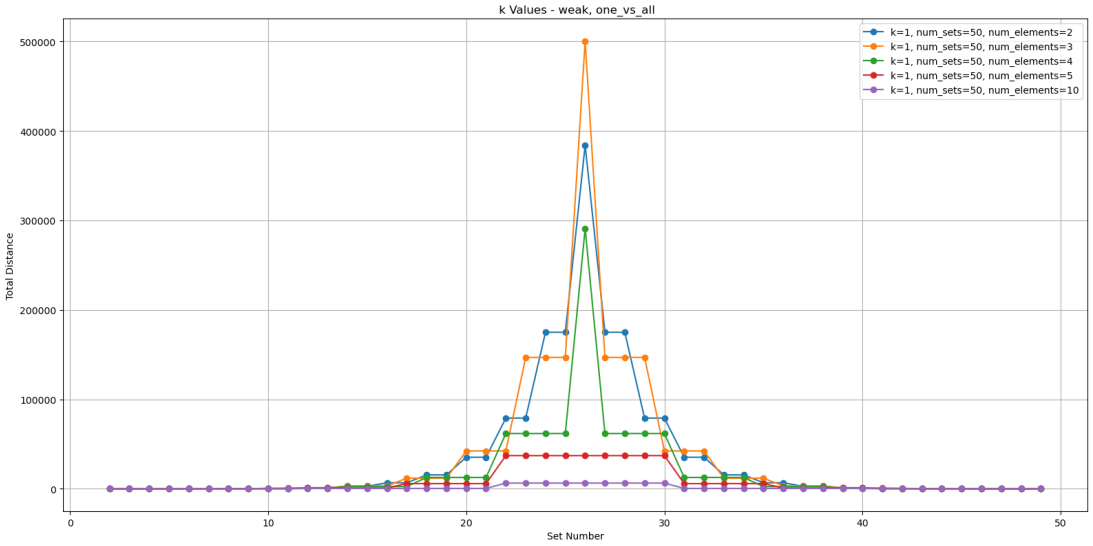

Figure 1: WOD for $k = 1$ and all num elements.

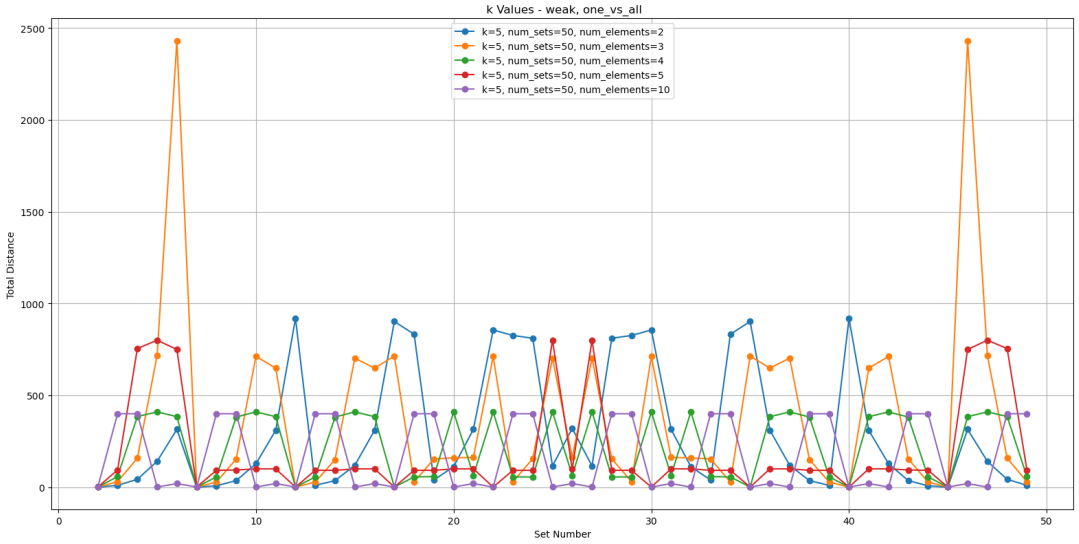

Figure 2: WOD for $k = 5$ and all num elements.

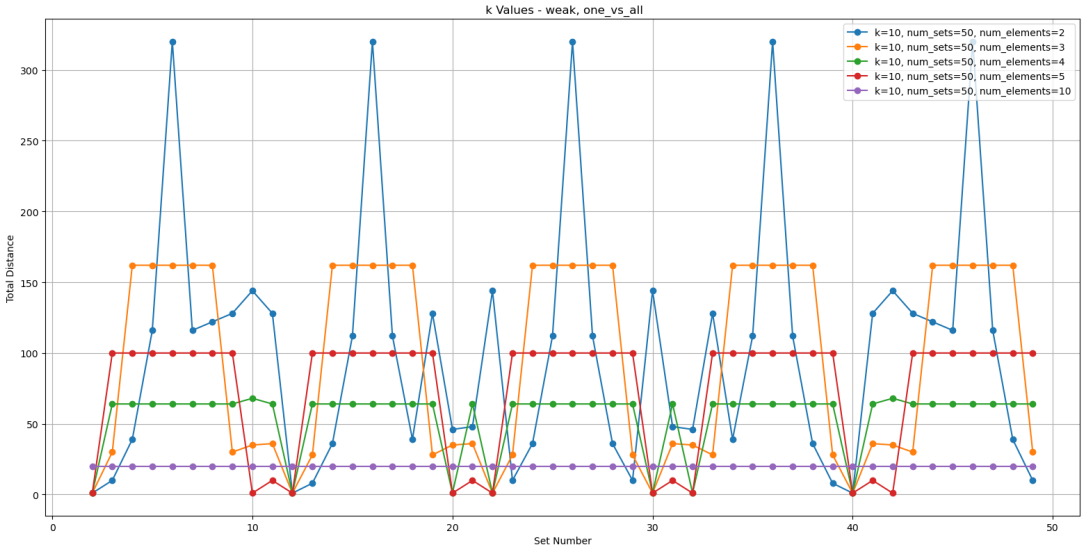

Figure 3: WOD for $k = 10$ and all num elements.

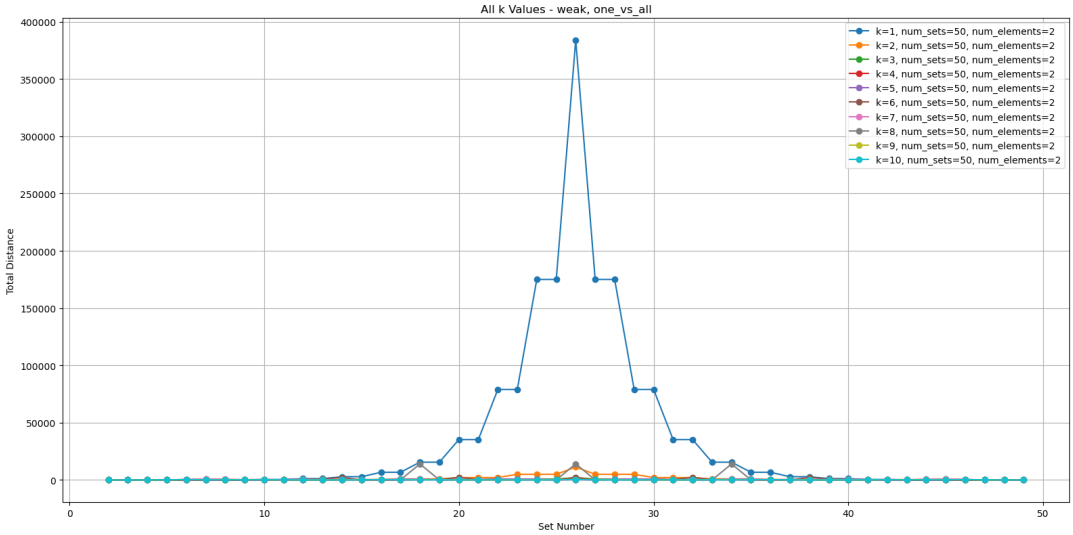

Figure 4: WOD for all k values for num elements = 2.

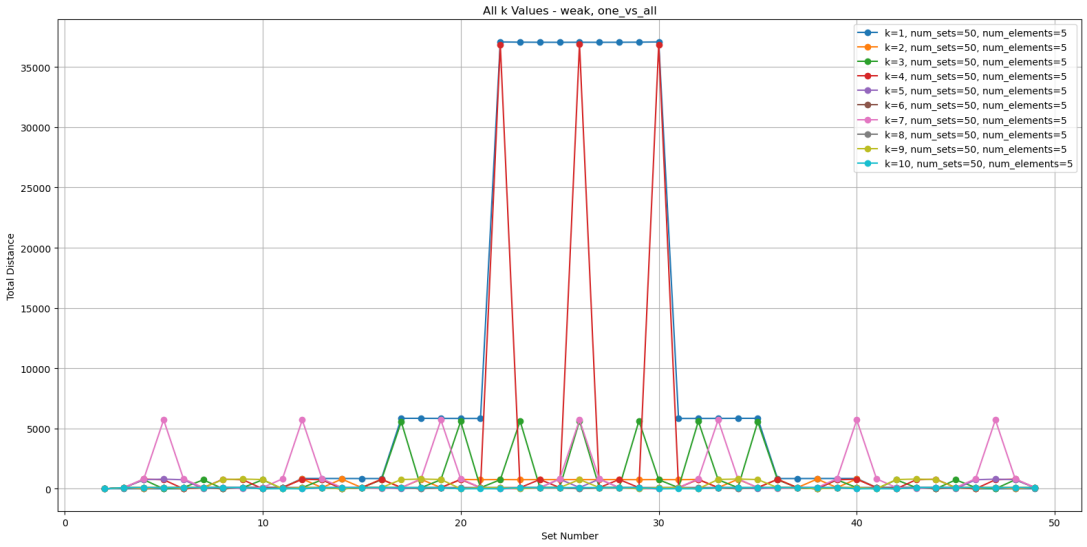

Figure 5: WOD for all k values for num elements = 5.

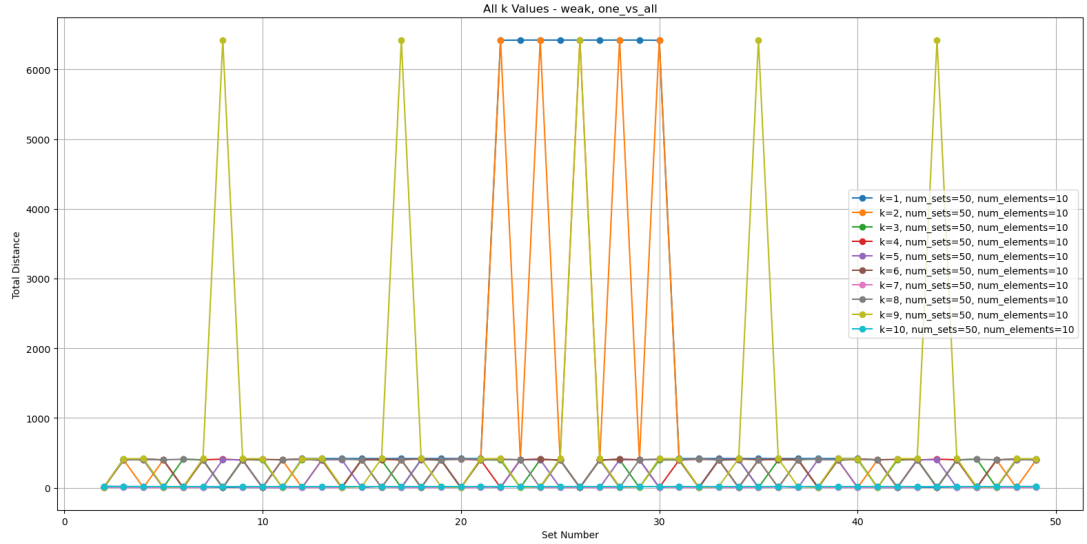

Figure 6: WOD for all k values and num elements=10.

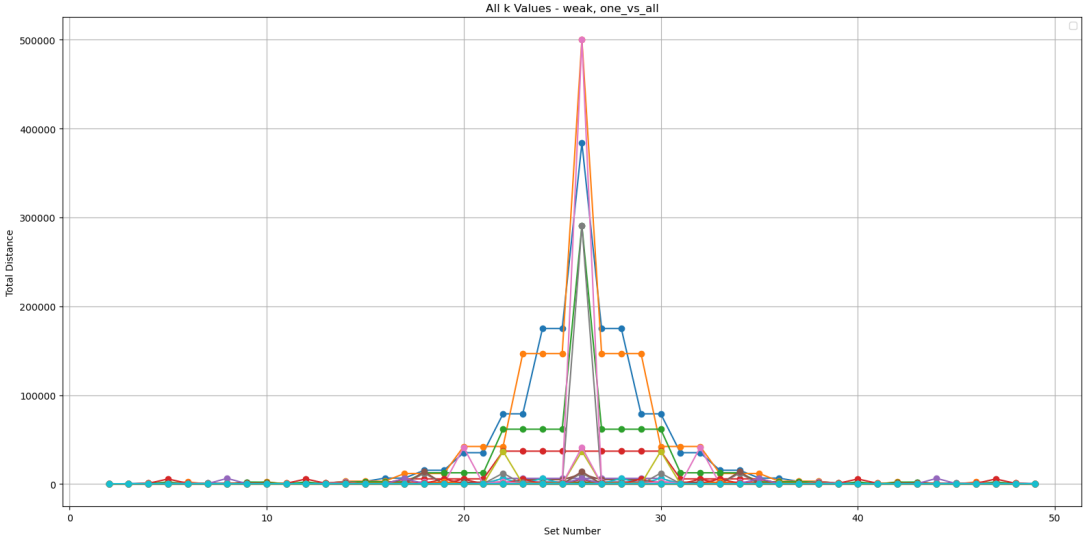

Figure 7: WOD for all k values and for all num elements with num sets=50.

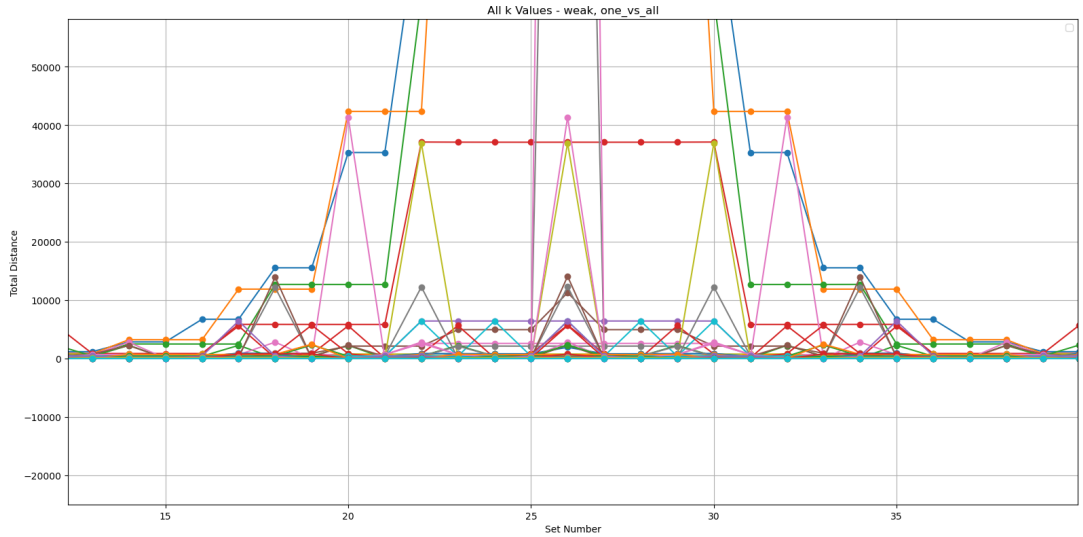

Figure 8: WOD for all k values and for all num elements with num sets=50, zoomed in.

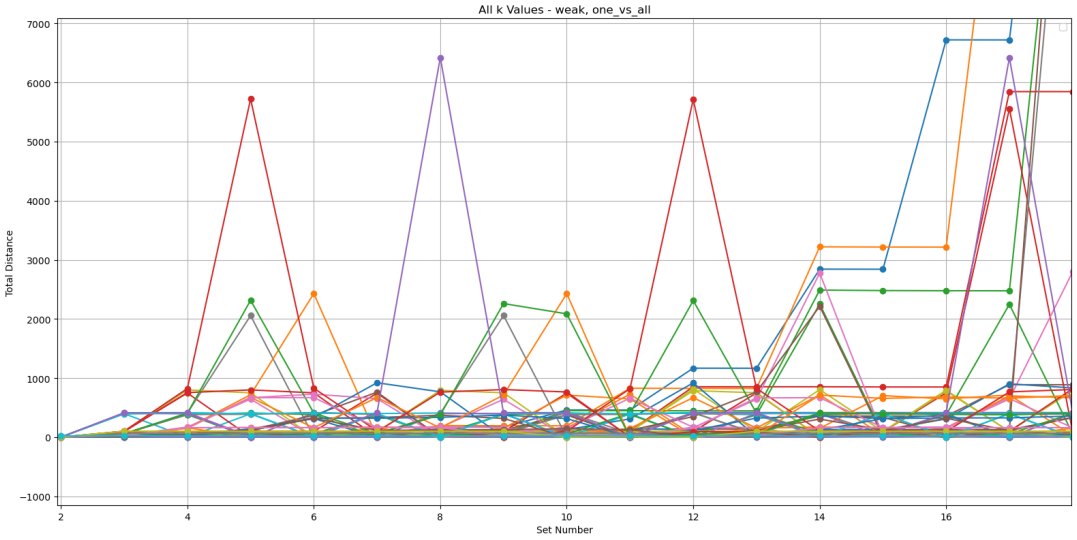

Figure 9: WOD for all k values and for all num elements with num sets=50, zoomed in.

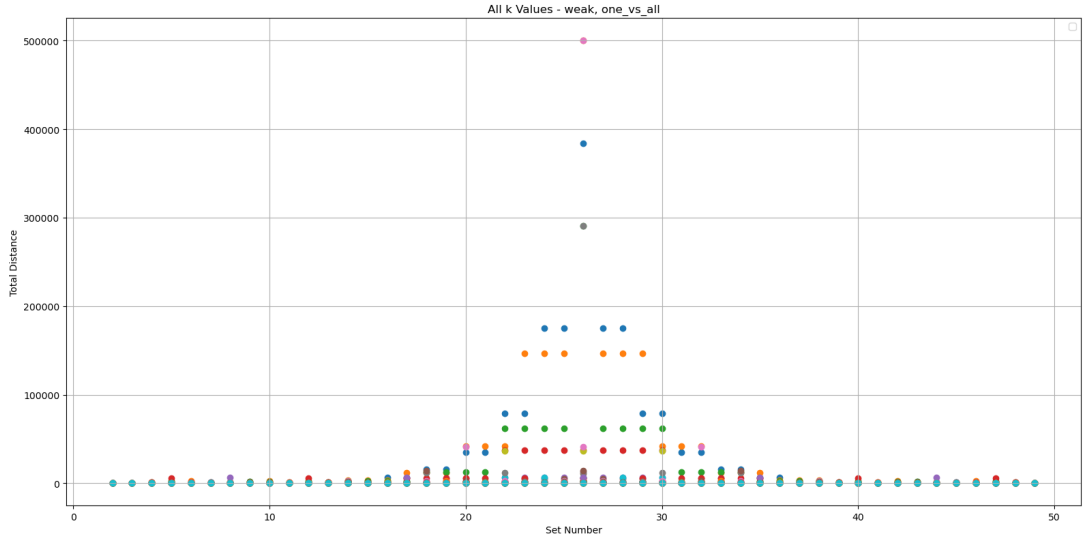

Figure 10: WOD for all k values and for all num elements with num sets=50, scattered.

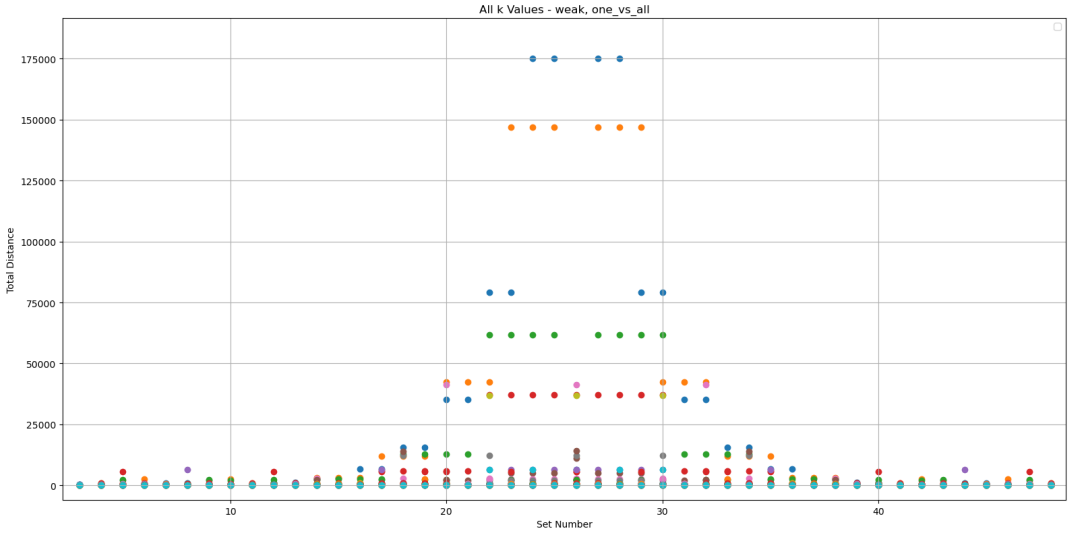

Figure 11: WOD for all k values and for all num elements with num sets=50, zoomed in, scattered.

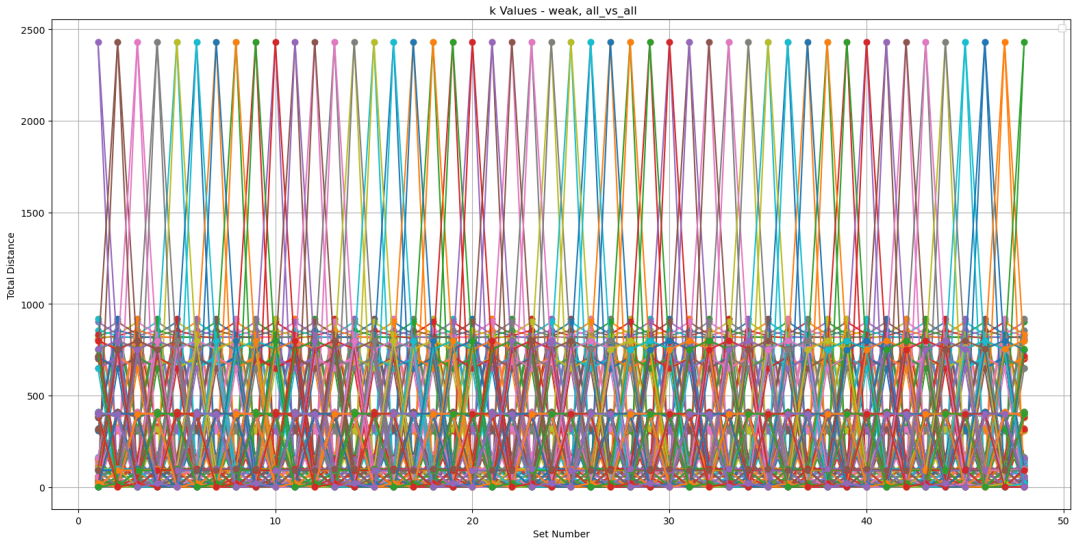

Figure 12: WOD of all vs all for num sets=50 for $k = 5$ and all num elements.

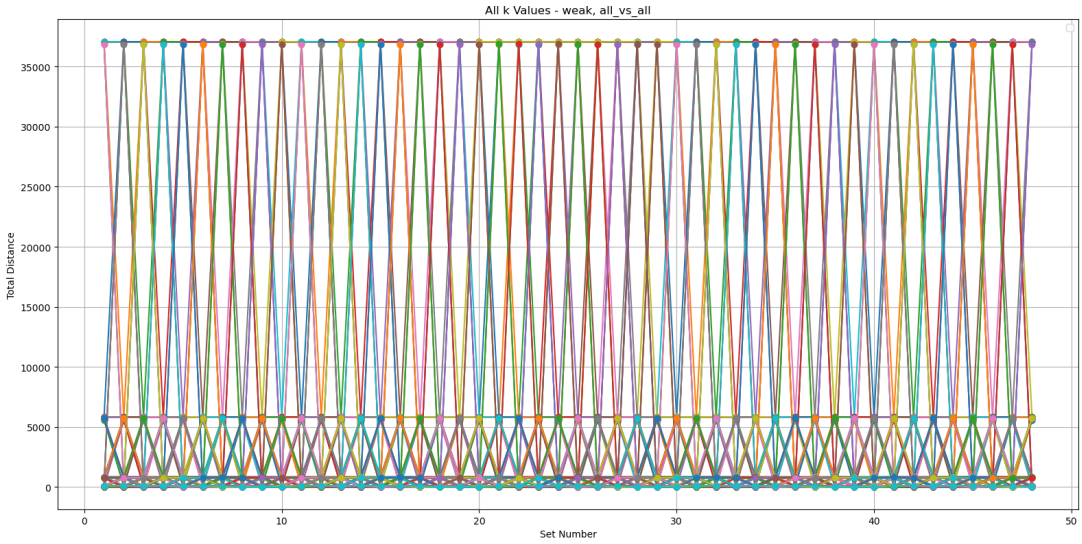

Figure 13: WOD of all vs all for num sets=50 for all k values for num elements = 5.

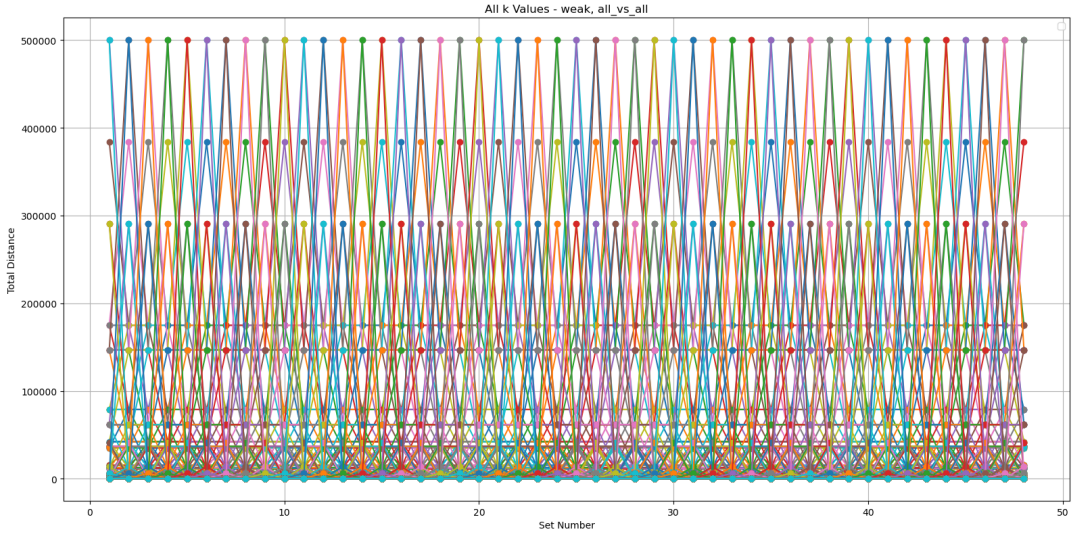

Figure 14: WOD of all vs all for all k values and for all num elements with num sets=50.

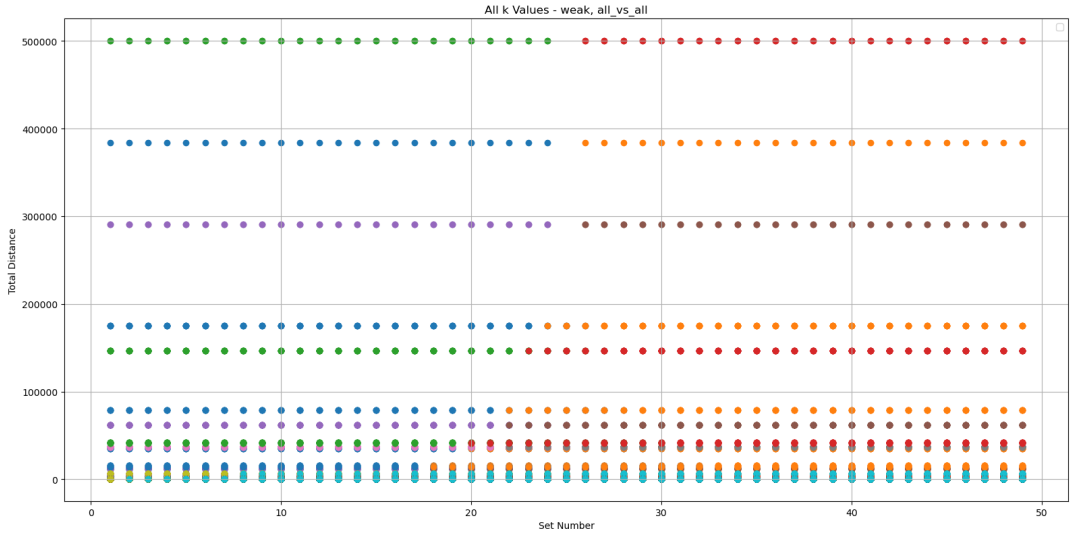

Figure 15: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, no zoom.

Now, if we take just a column of data (in this case for set 24) under 100.000, we will be able to see a further and deeper gapping patterns as we keep zooming.

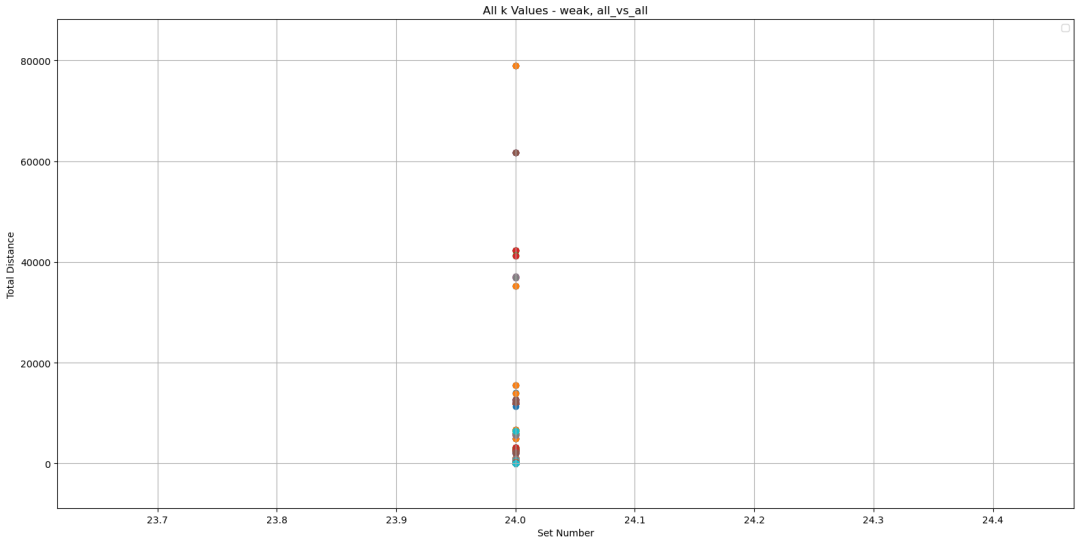

Figure 16: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, zoom 1.

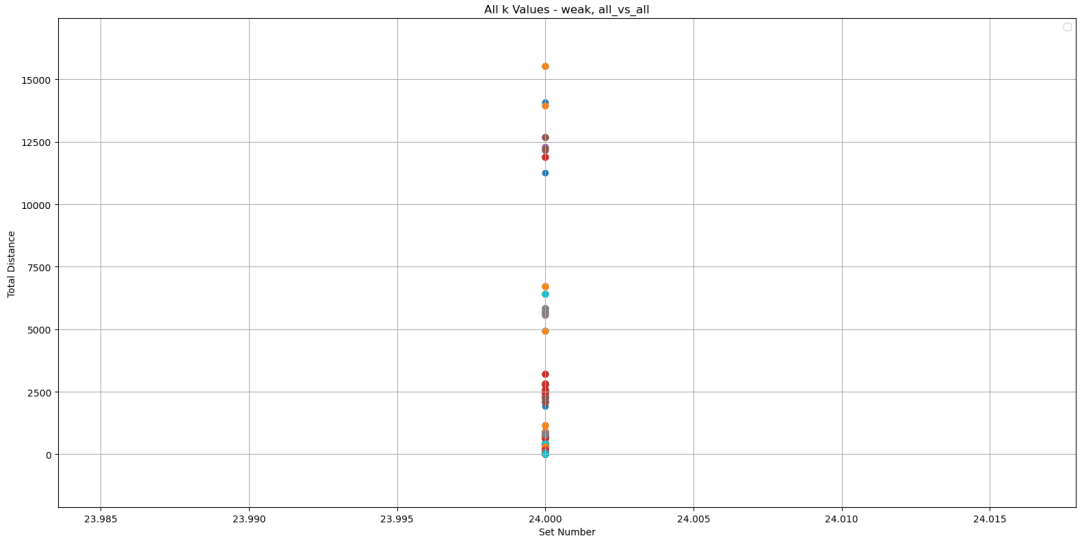

Figure 17: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, zoom 2.

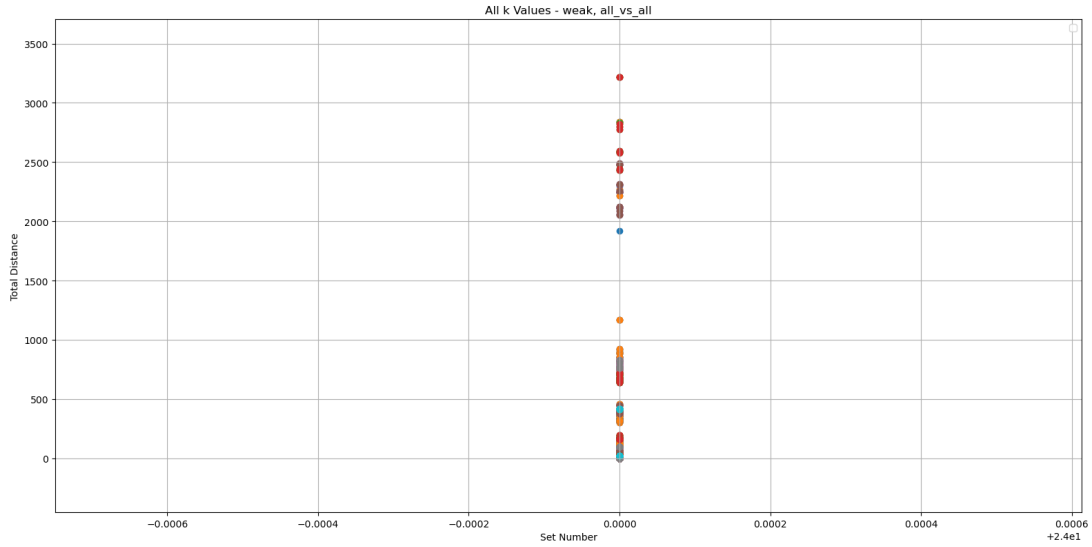

Figure 18: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, zoom 3.

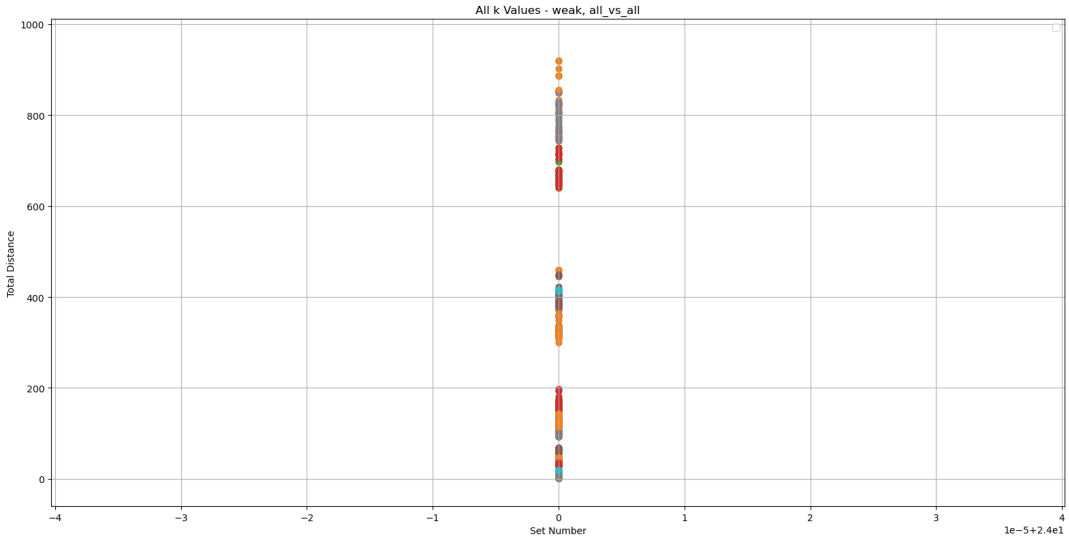

Figure 19: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, zoom 4.

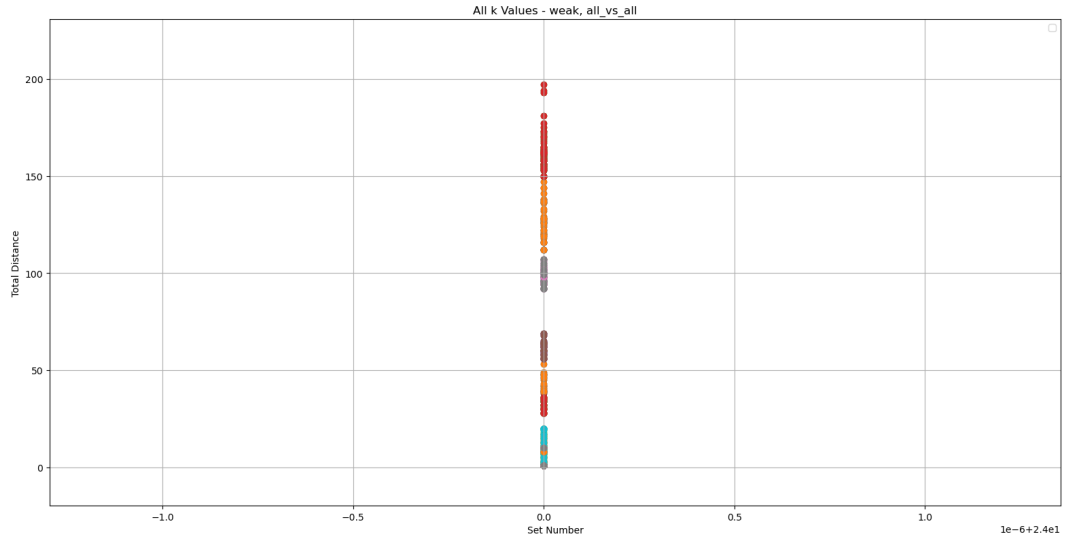

Figure 20: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, zoom 5.

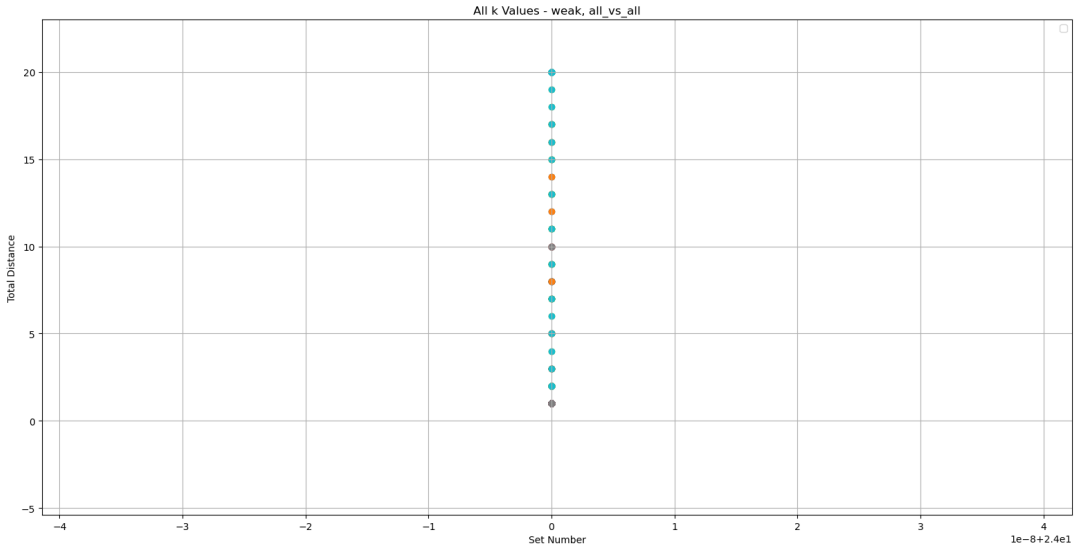

Figure 21: WOD of all vs all for all k values and for all num elements with num sets=50, scattered, zoom 6.

4.1.2. Results for SOD

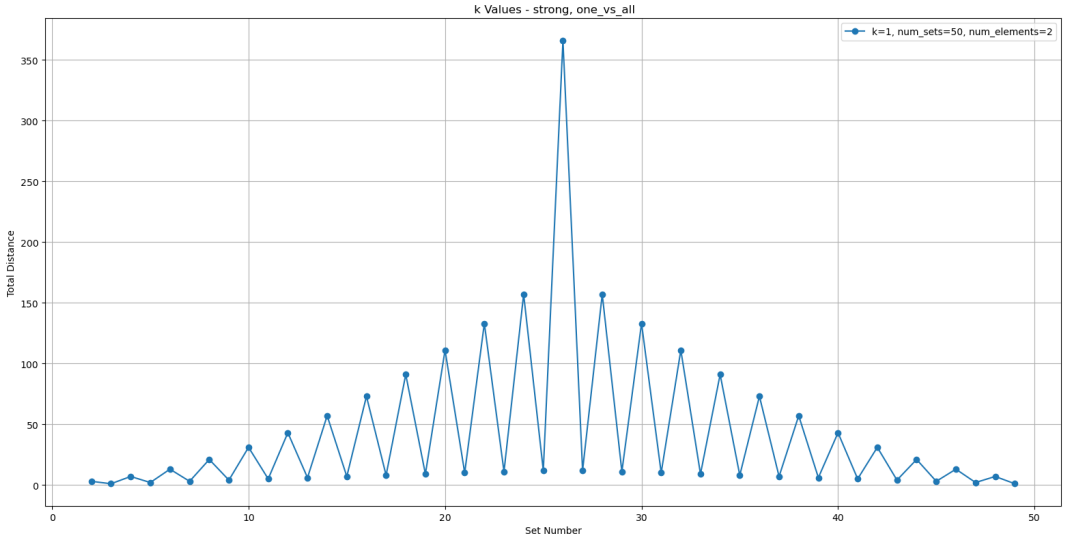

Figure 22: SOD for $k = 1$ and num elements= 2.

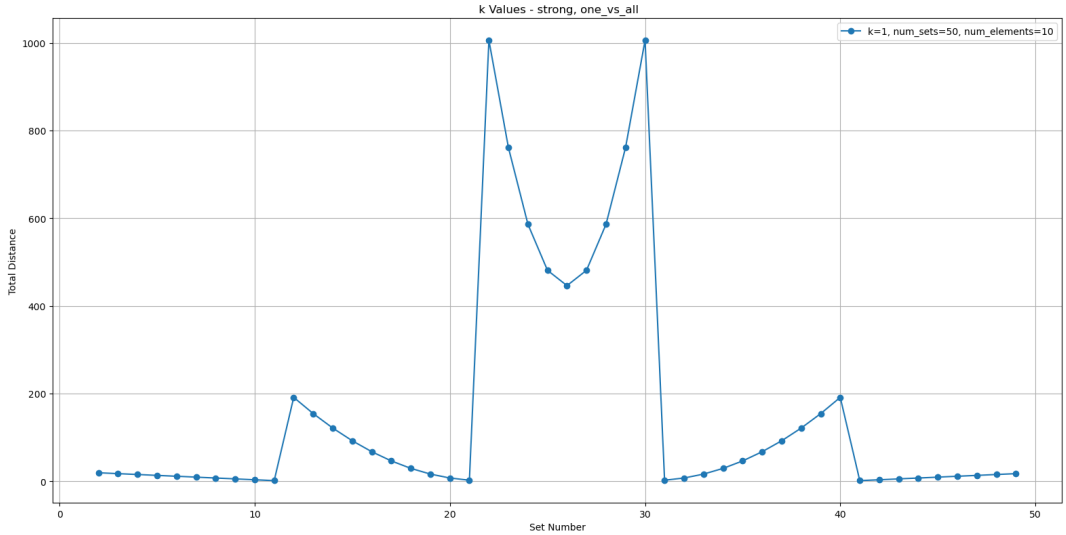

Figure 23: SOD for $k = 1$ and num elements$=$ 10.

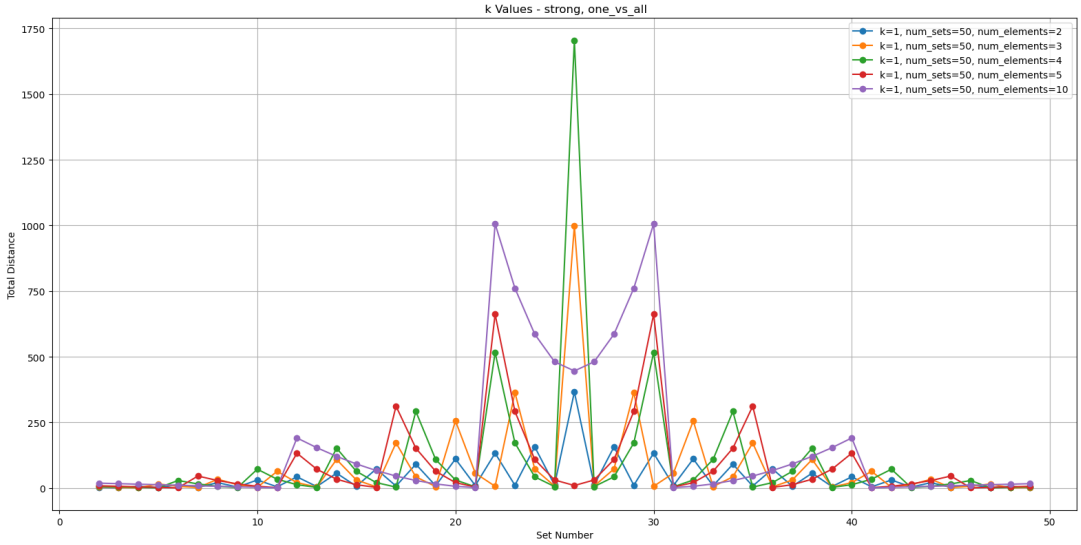

Figure 24: SOD for $k = 1$ and all num elements.

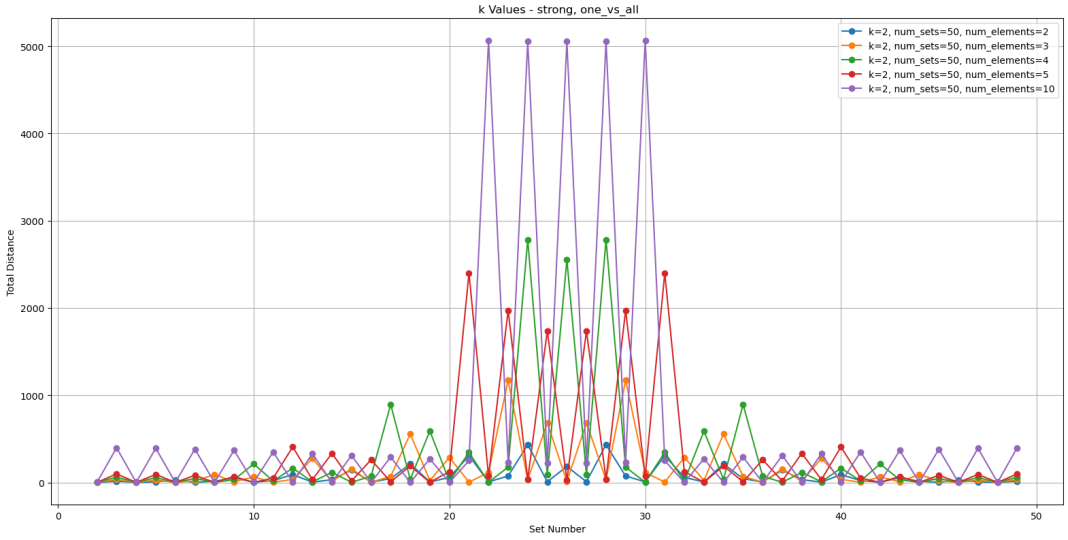

Figure 25: SOD for $k = 2$ and all num elements.

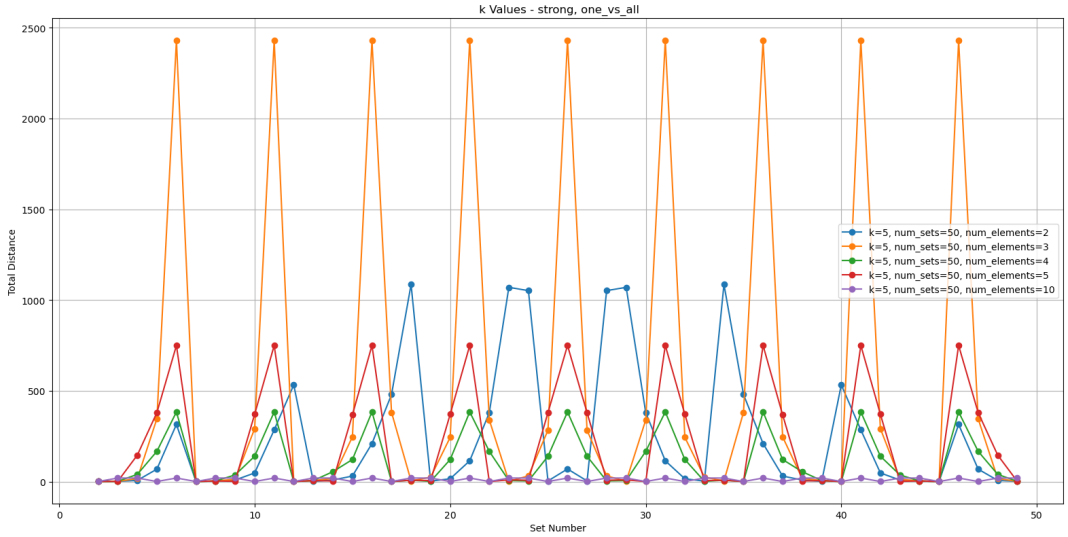

Figure 26: SOD for $k = 5$ and all num elements.

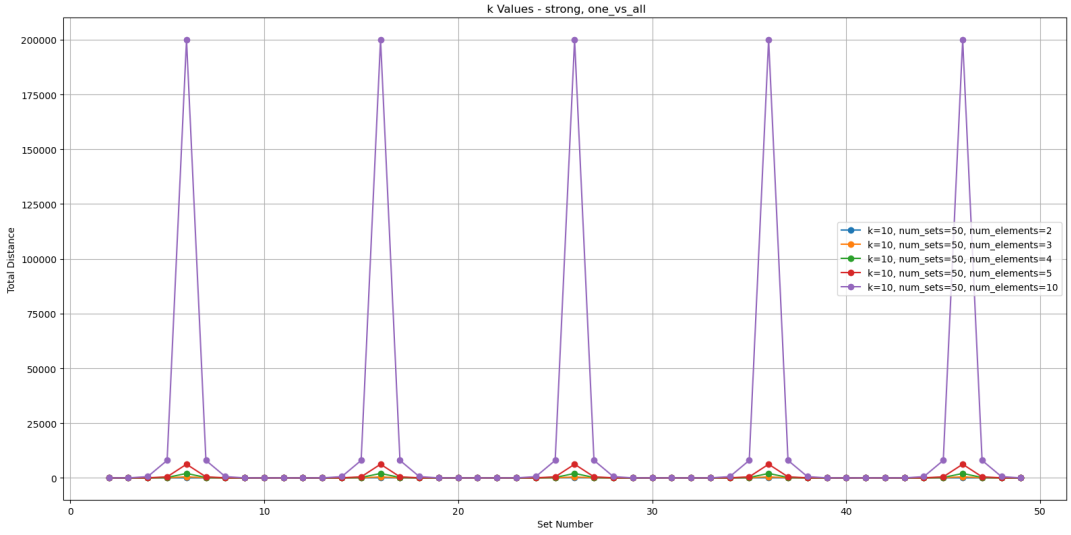

Figure 27: SOD for $k = 10$ and all num elements.

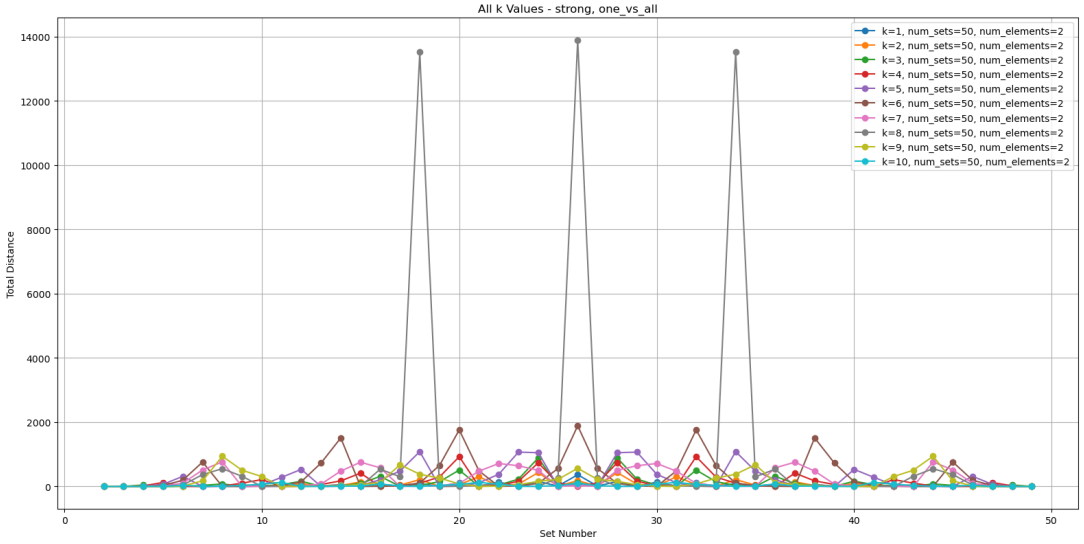

Figure 28: SOD for all k values for num elements $= 2$.

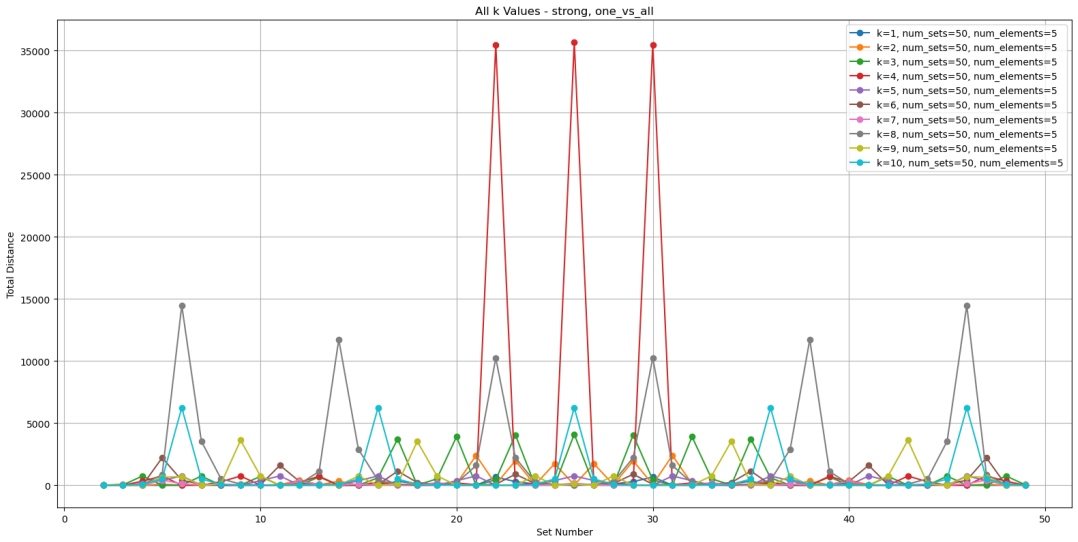

Figure 29: SOD for all k values for num elements = 5.

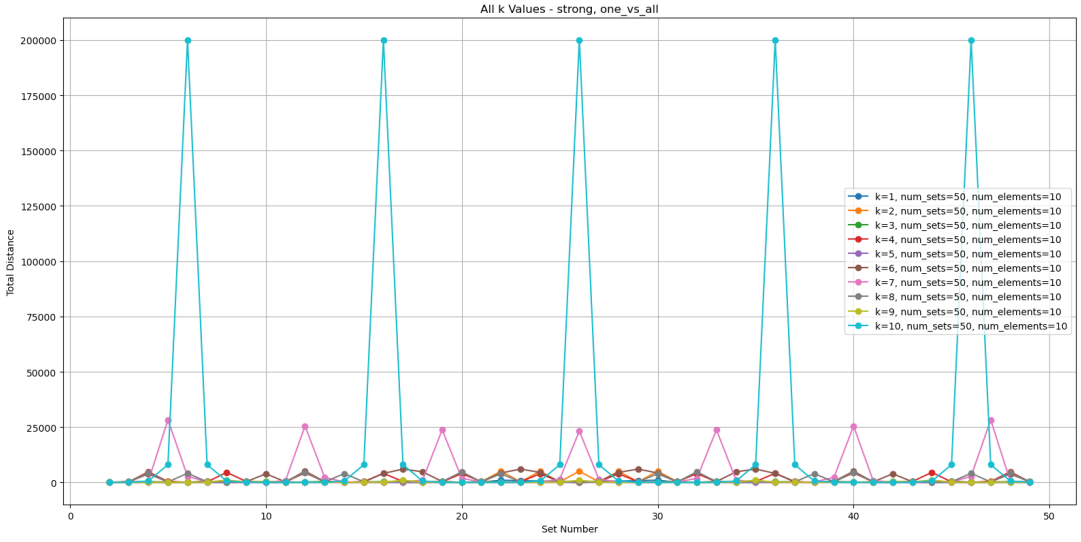

Figure 30: SOD for all k values for num elements = 10.

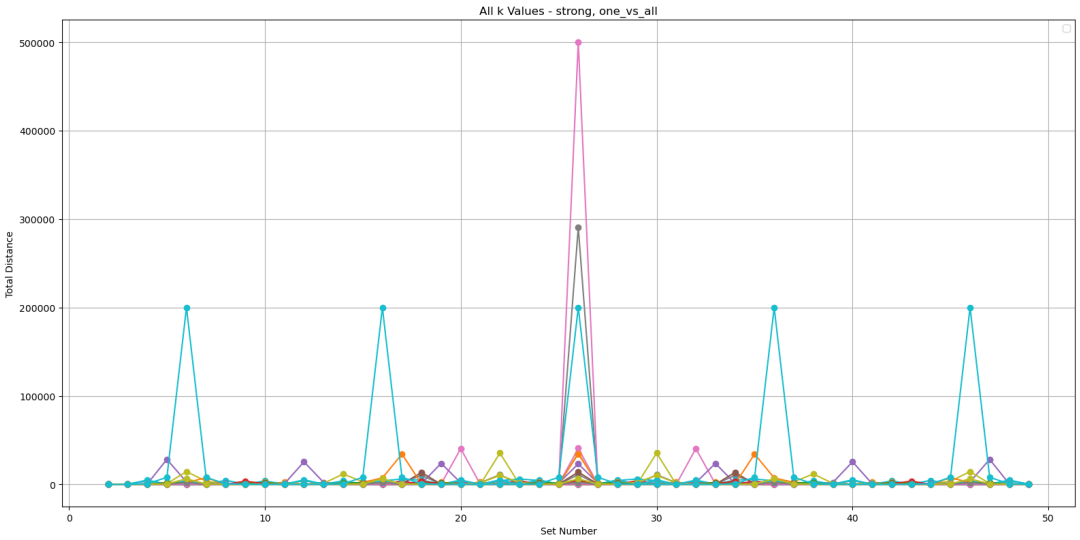

Figure 31: *SOD* for all k values and for all num elements with num sets=50.

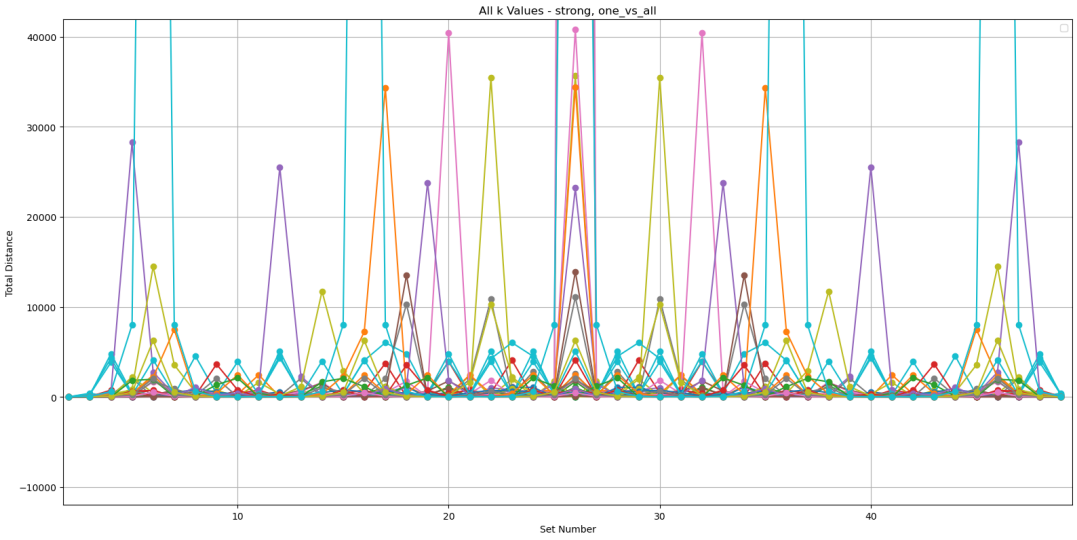

Figure 32: *SOD* for all k values and for all num elements with num sets=50, zoomed in.

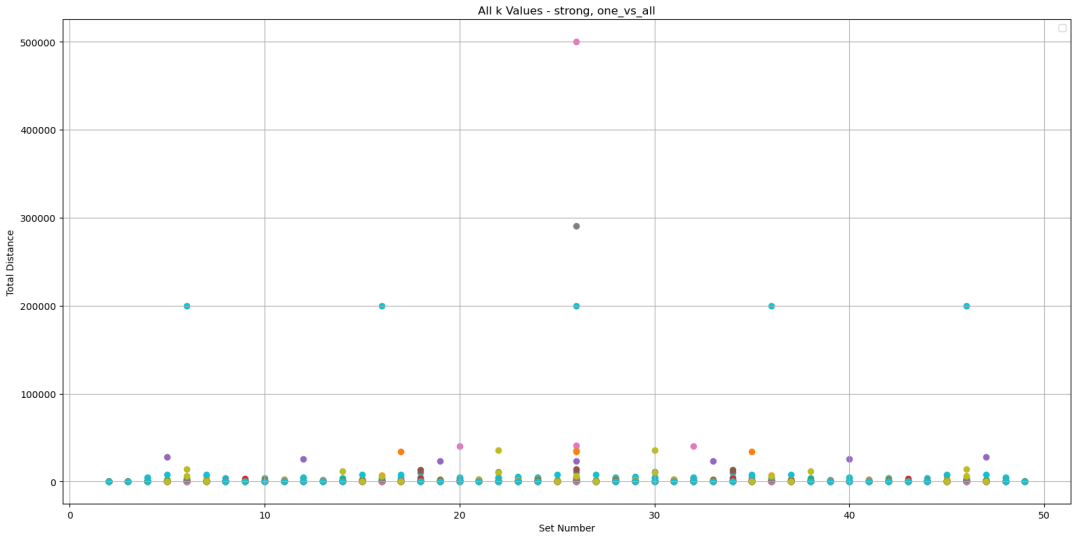

Figure 33: *SOD* for all k values and for all num elements with num sets=50, scattered.

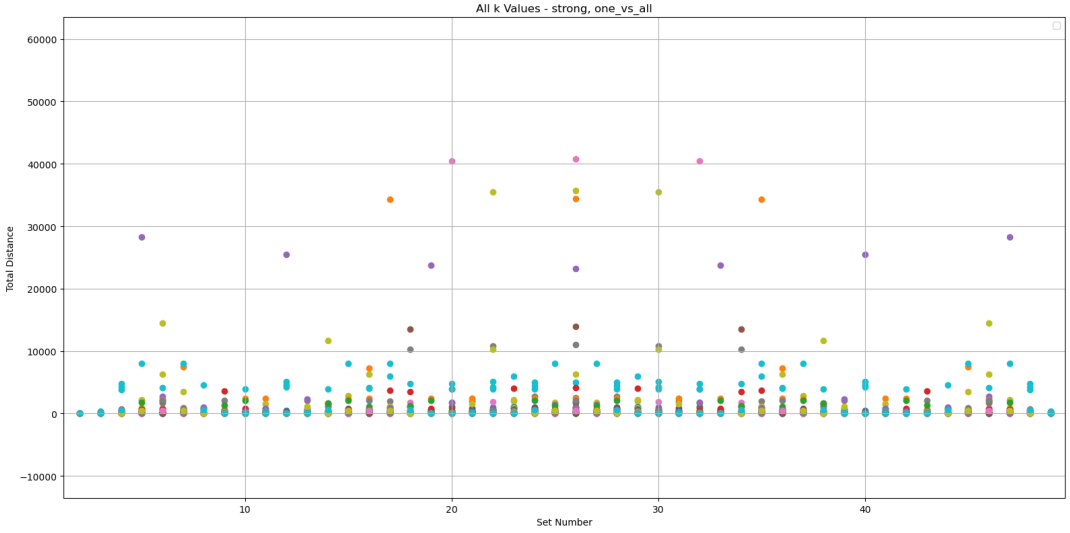

Figure 34: *SOD* for all k values and for all num elements with num sets=50, zoomed in, scattered.

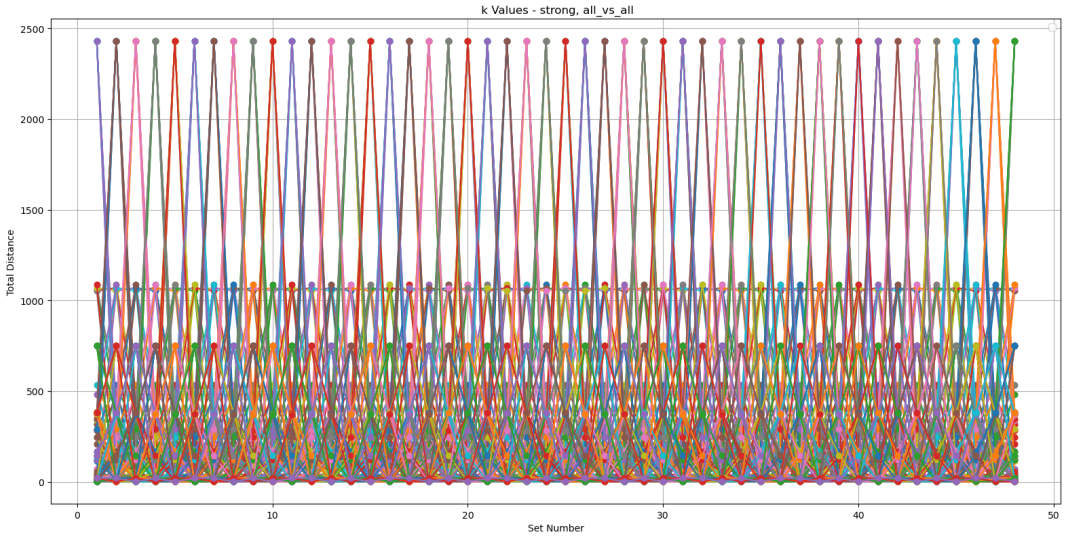

Figure 35: *SOD* of all vs all for num sets=50 for $k = 5$ and all num elements.

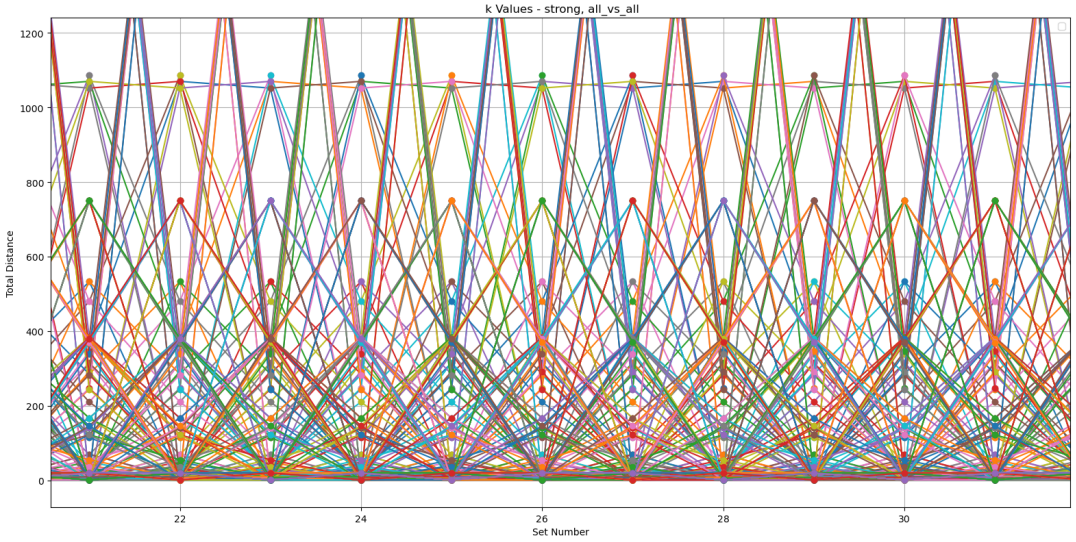

Figure 36: *SOD* of all vs all for num sets=50 for $k = 5$ and all num elements, zoomed in.

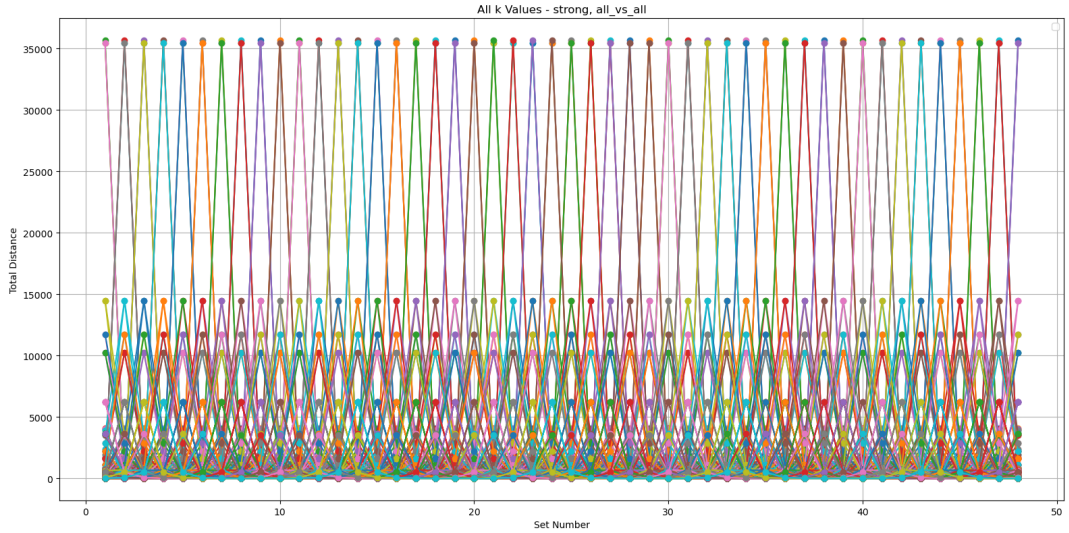

Figure 37: *SOD* of all vs all for num sets=50 for all k values and num elements=5.

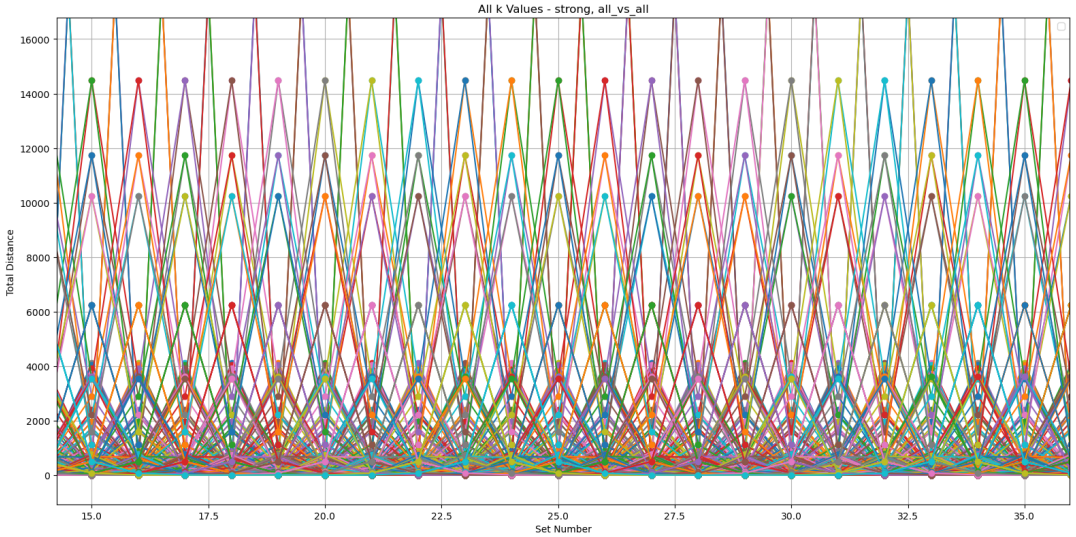

Figure 38: *SOD* of all vs all for num sets=50 for all k values and num elements=5, zoomed in.

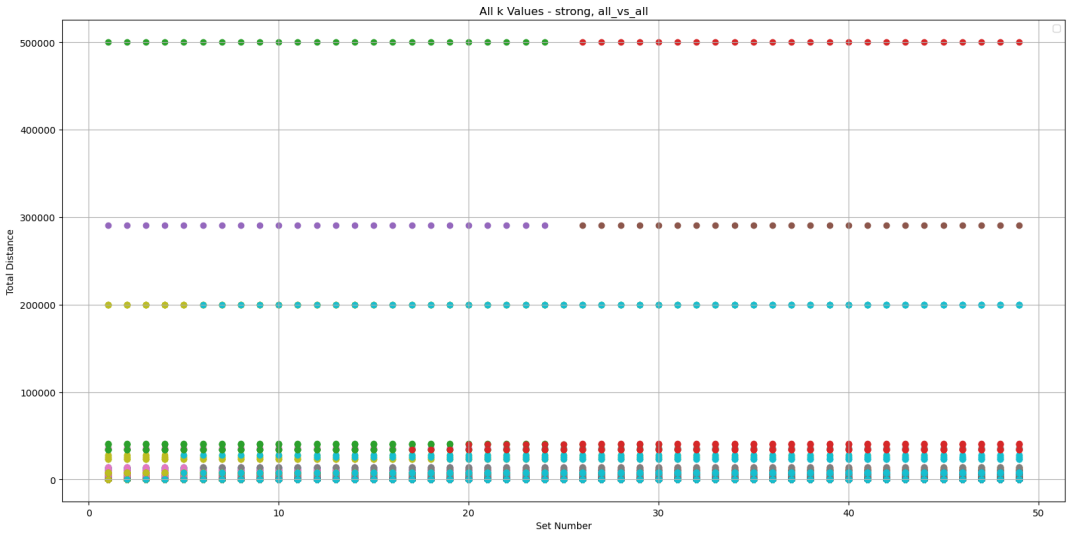

Figure 39: *SOD* of all vs all for all k values and for all num elements with num sets=50, scattered.

Just like it did for WOD, the further zooming in of lower values will also show a deeper gapping pattern, which for length reasons we will not be showing here.

5. Recursive Functions

In this sub-section we will deal with functions in the $\mathbb{R}$ realm. Our structure S will be the same as in (9), with a recursive adaptation in order for the sets to be rightfully calculated. This recursive adaptation consists of the following: If we have x_n and $x_{n_0} = [x_{n_1}, ..., x_{n_k}]$, the following points x_{n+m} will be taken from $\forall m \in x_{n_m}$.

Example 14. *For a* **U** *of size 10, with elements of size 2, for a function* $f(x) = \frac{1}{x}$*, and the already given structure* S*, we would have the following sets:*

Set 1: *[2, 0.5]*
Set 2: *[3, $0.\overline{3}$]*
Set 0.5: *[$0.\overline{6}$, 1.5]*
Set 3: *[4, 0.25]*
Set $0.\overline{3}$: *[0.75, $1.\overline{33}$]*
Set $0.\overline{6}$: *[$1.\overline{6}$, 0.6]*
Set 1.5: *[0.4, 2.5]*
Set 4: *[5, 0.2]*
Set 0.25: *[0.8, 1.25]*
Set 0.75: *[1.75, 0.57]*

We have obviously rounded up some of these numbers in this example in order to give it a shorter look.

As we can see, there are some phantoms for a **U** of this size, what we will do when one is found is expand them until the OD in question is finished. The OD's will only be performed between $Set\,1$ and the rest.

5.1. $f(x) = \frac{1}{x}$

We will investigate the OD's applied to a recursive/infinite **U**, ordered for the given structure S, and the function $f(x) = \frac{1}{x}$, where our variable parameters will only be the size of **U**, the size of the elements will be restricted to 2.

In this subsection we will start by showing the SOD case first since the WOD case will be an interesting alternation.

5.1.1. Results for SOD

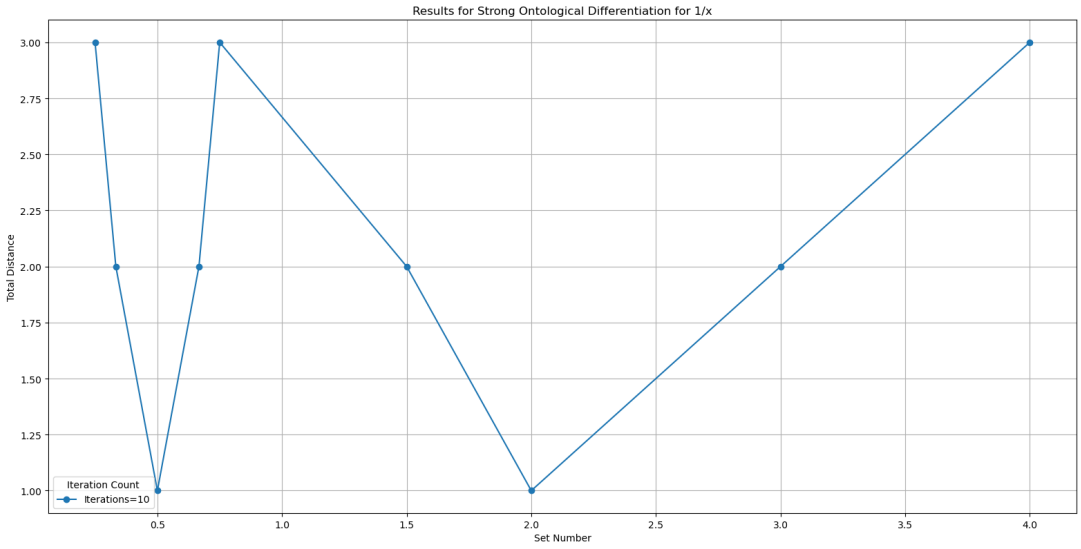

Figure 40: SOD for $1/x$ for iteration 10.

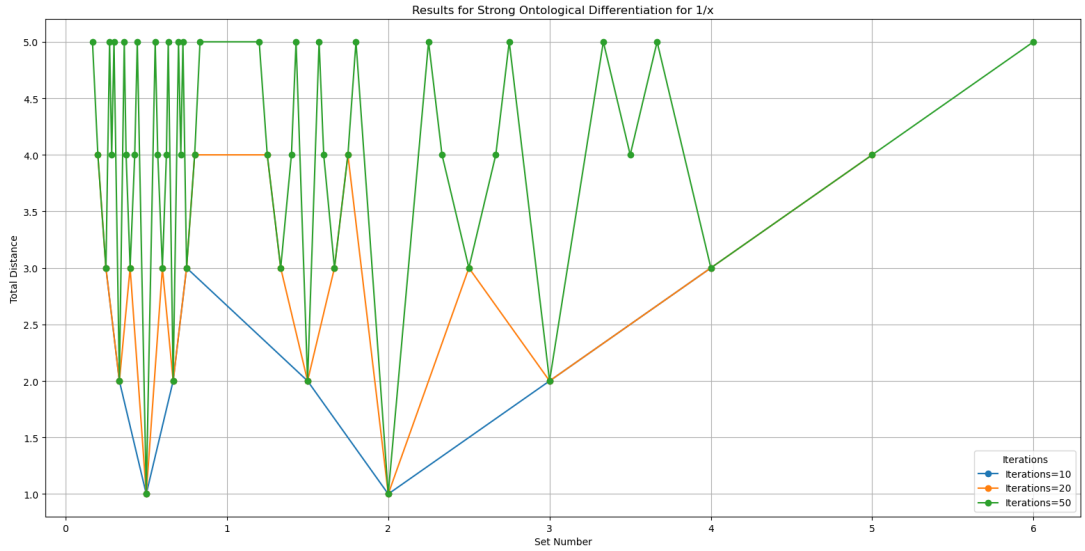

Figure 41: SOD for $1/x$ for iterations 10, 20, 50.

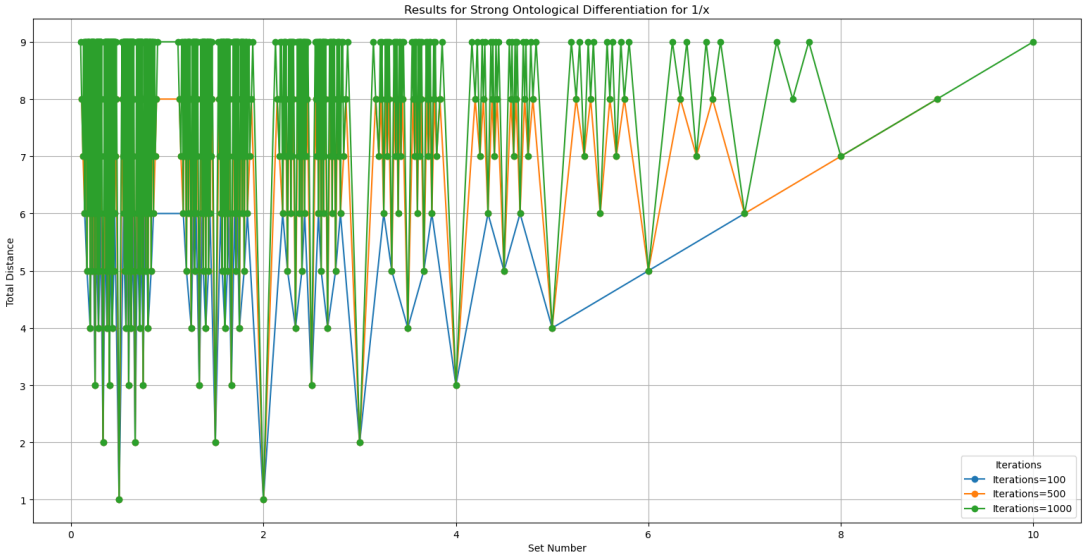

Figure 42: SOD for $1/x$ for iterations 100, 500, 1000.

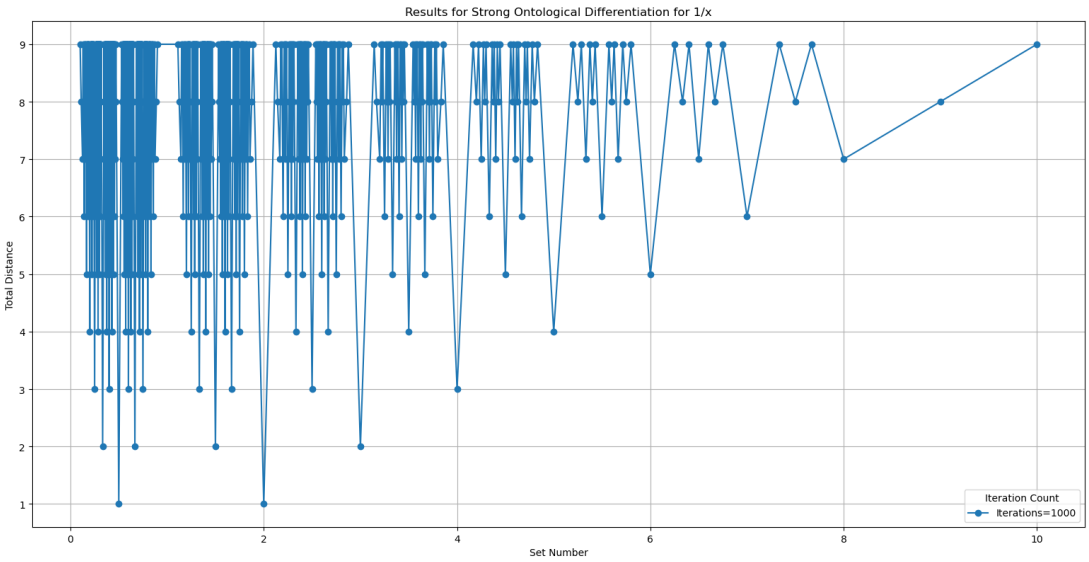

Figure 43: SOD for $1/x$ for iteration 1000.

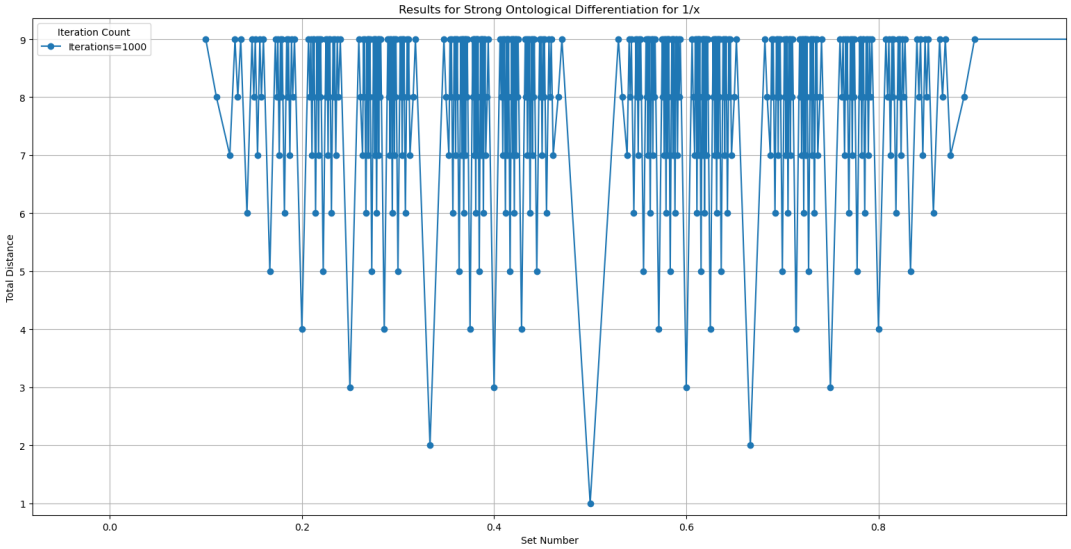

Figure 44: SOD for $1/x$ for iteration 1000, zoomed in.

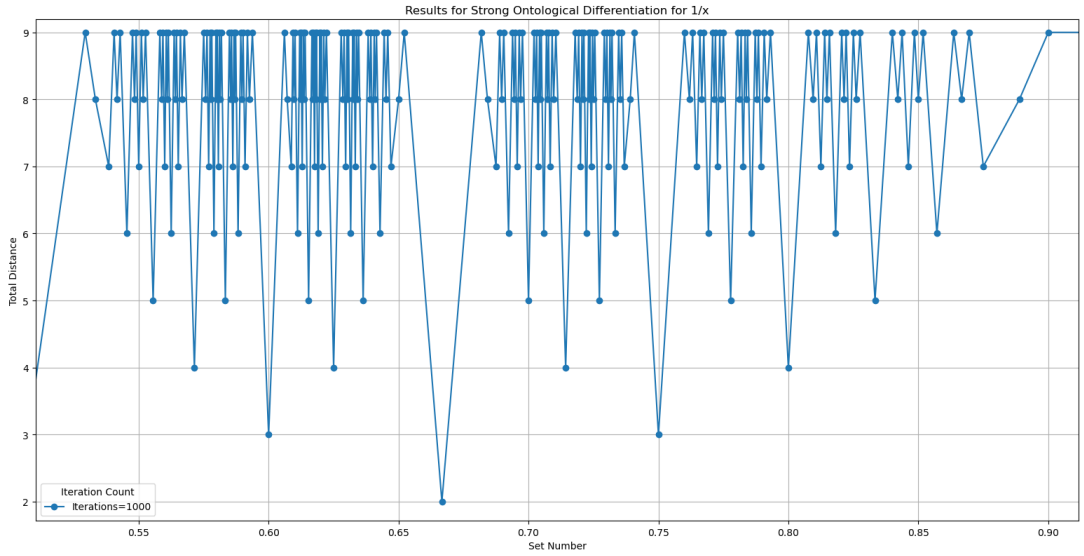

Figure 45: SOD for $1/x$ for iteration 1000, zoomed in further.

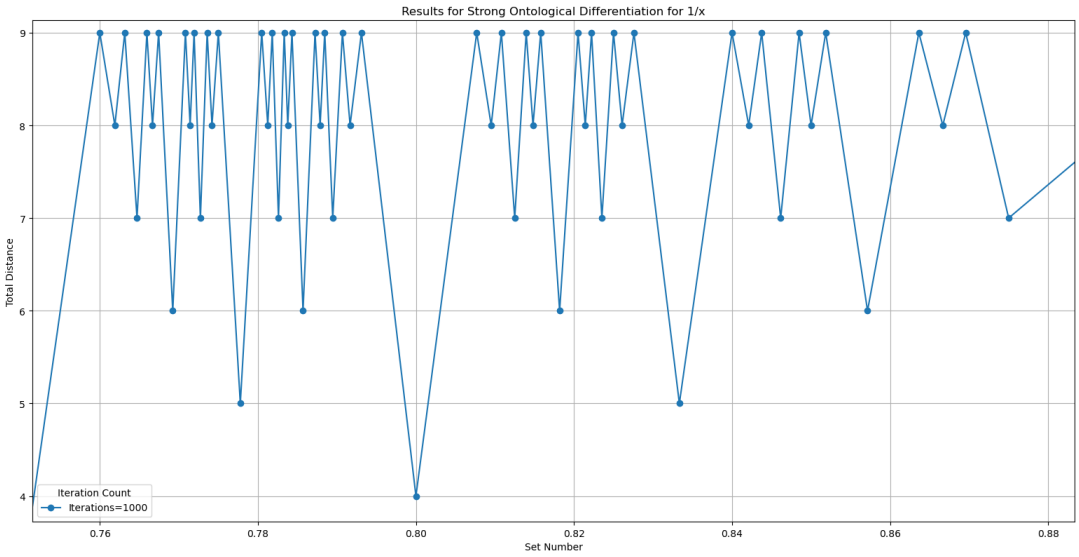

Figure 46: SOD for $1/x$ for iteration 1000, zoomed in further.

As one can see, a structure of progressive fractality appears. As the number of iterations grow, the structure of each of the "towers" gets more self similar, and the tower structure starts reproducing itself on the right. This can perhaps be better perceived if we display the plots as scattered points. In the next plots we will show the results this way so one can easily check that the aforementioned principle of progressive fractality is present.

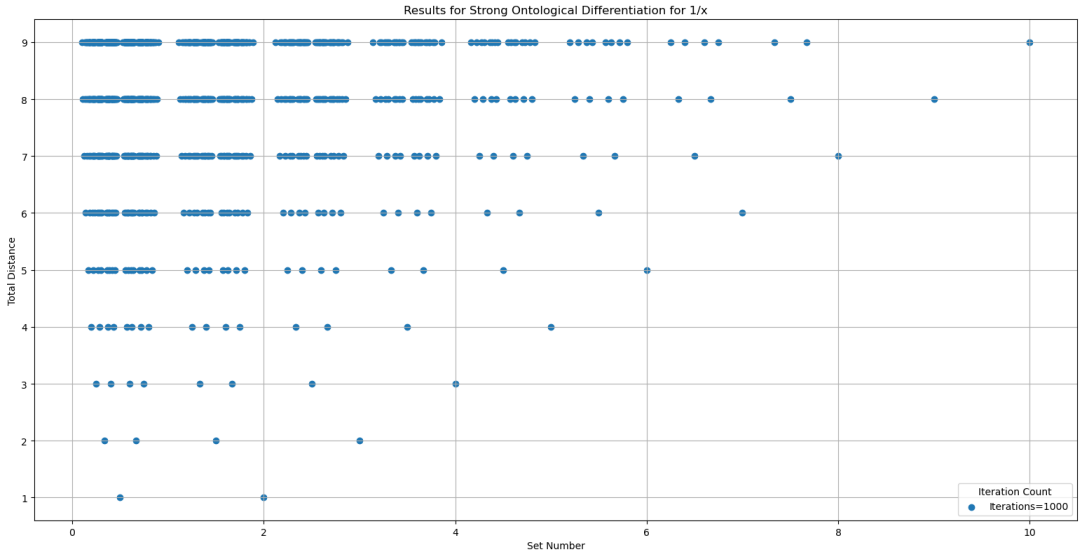

Figure 47: SOD for $1/x$ for iteration 1000, scattered.

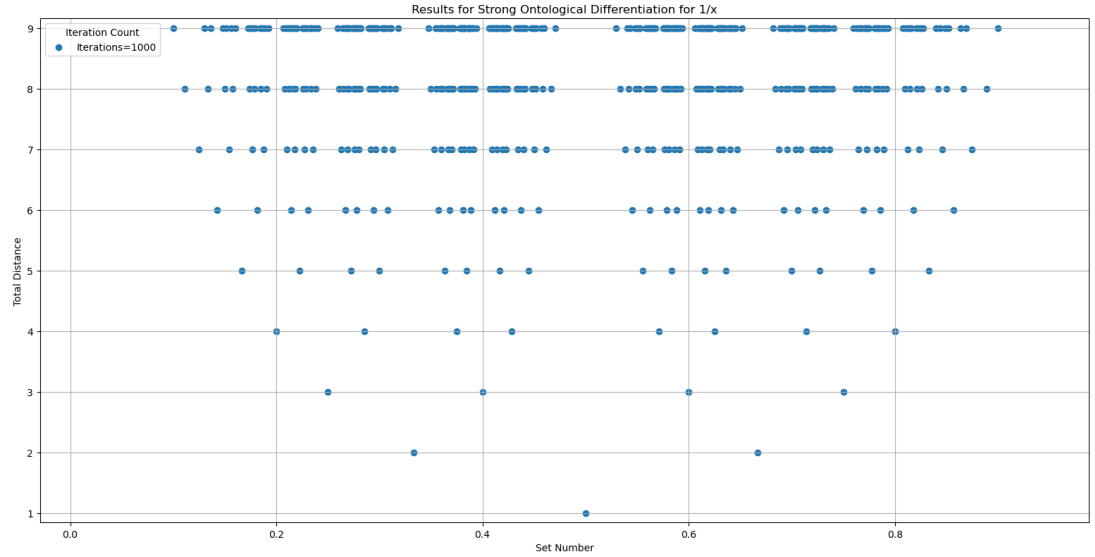

Figure 48: SOD for $1/x$ for iteration 1000, scattered, zoomed in.

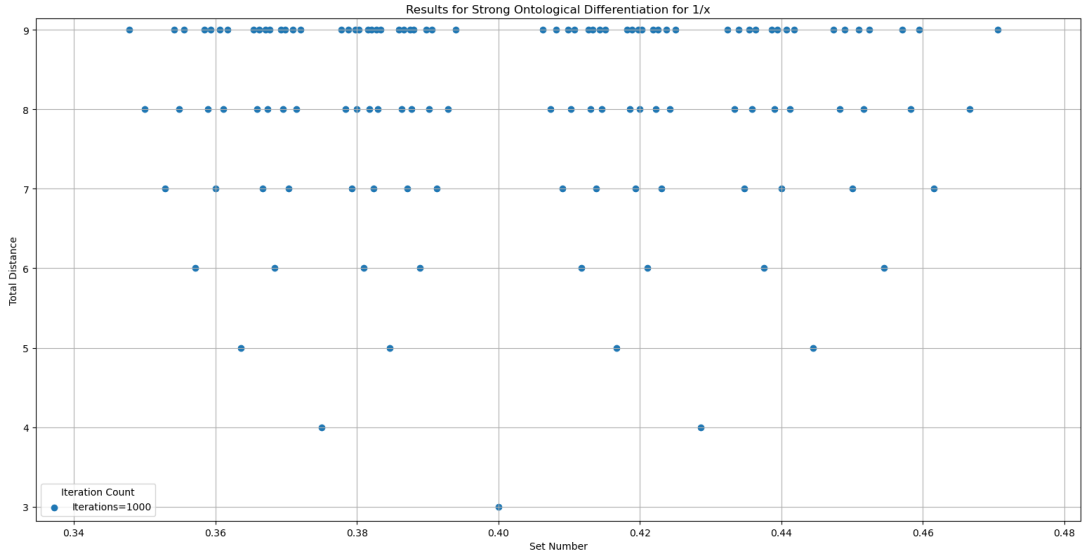

Figure 49: SOD for $1/x$ for iteration 1000, scattered, zoomed in further.

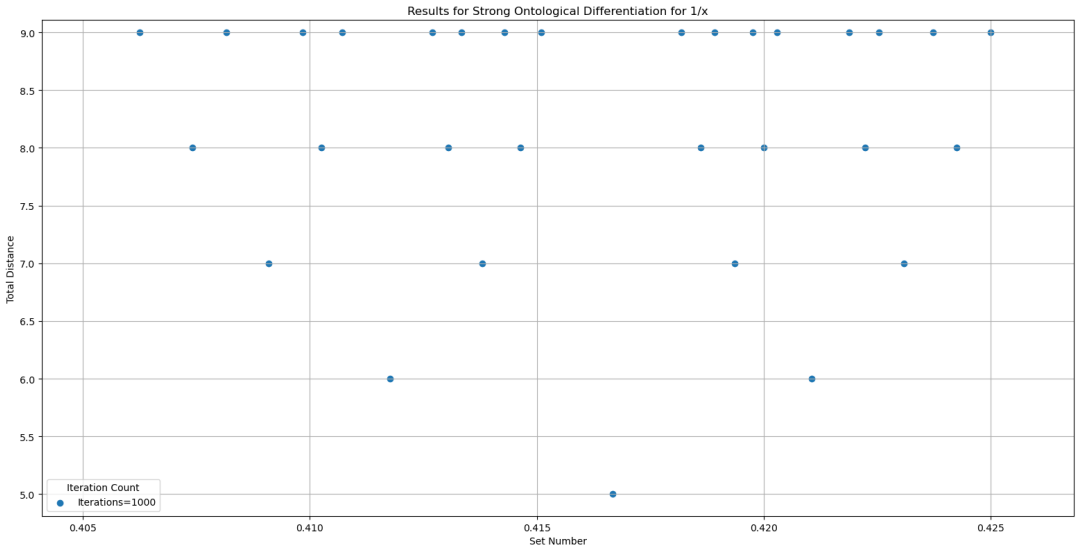

Figure 50: SOD for $1/x$ for iteration 1000, scattered, zoomed in further.

5.1.2. Results for WOD

The WOD case for this function exhibits a rather interesting contrast to the SOD case. Let us first investigate the WOD alone.

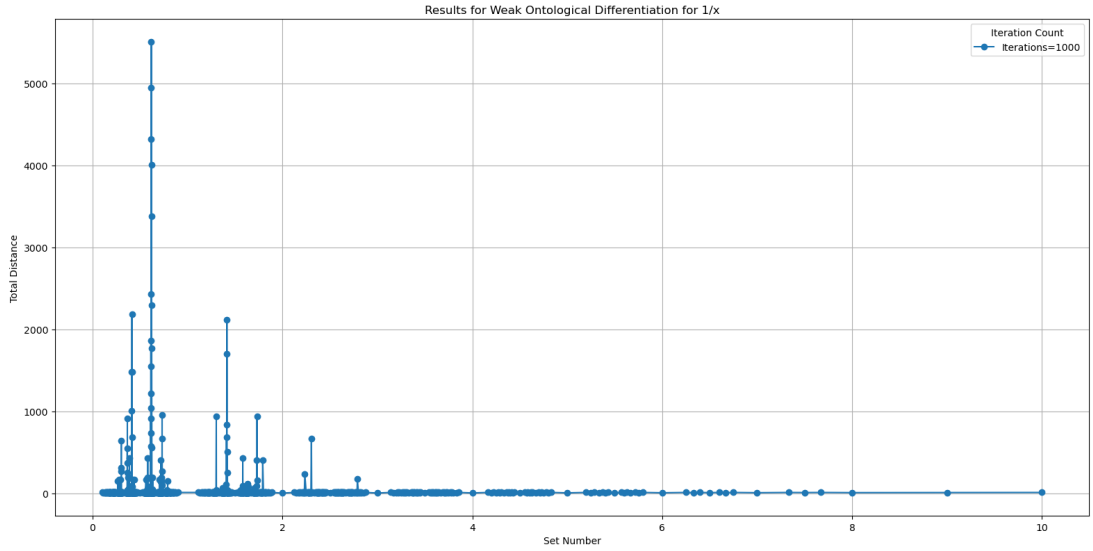

Figure 51: WOD for $1/x$.

As we can see, the values range here is much higher than in the SOD case.

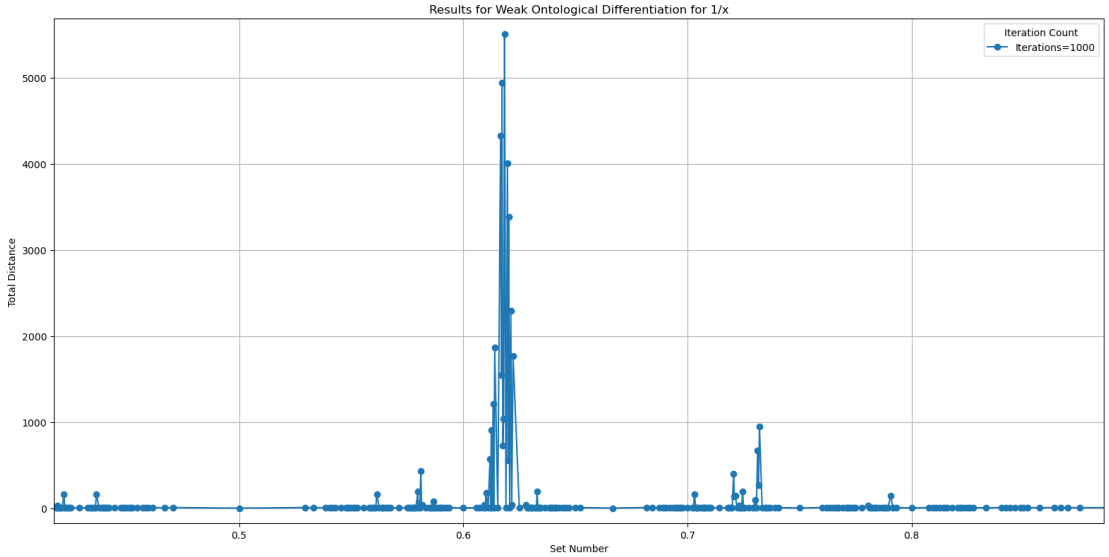

Figure 52: WOD for $1/x$, zoomed in.

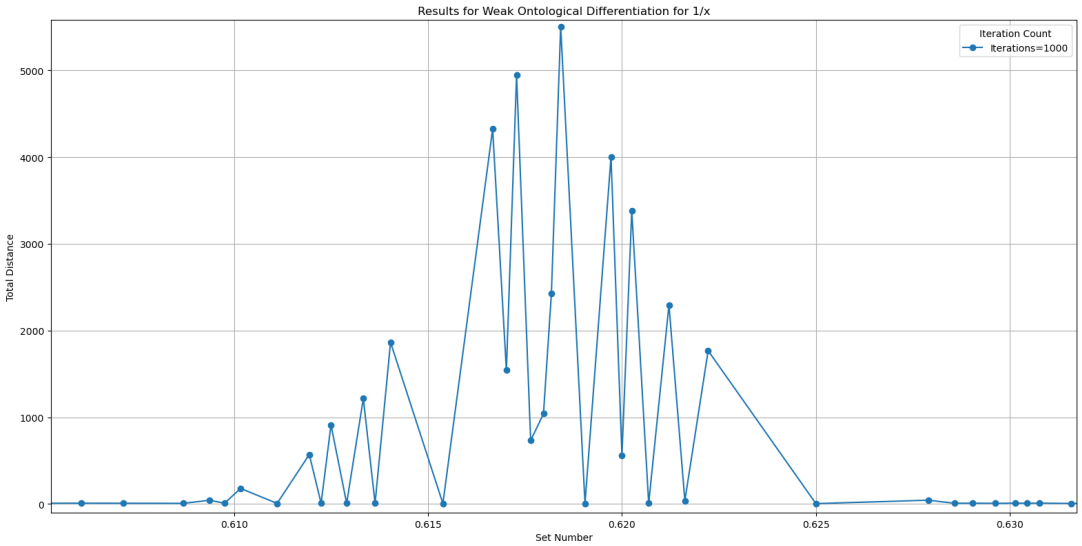

Figure 53: WOD for $1/x$, zoomed in further.

We see that the highest peak consists of an ensemble of other peaks going back and forth between the highest and lowest values. We have not zoomed in the lower regions of the graph to see what that accumulation of points look like, we will do that now as we do the comparison between SOD and WOD.

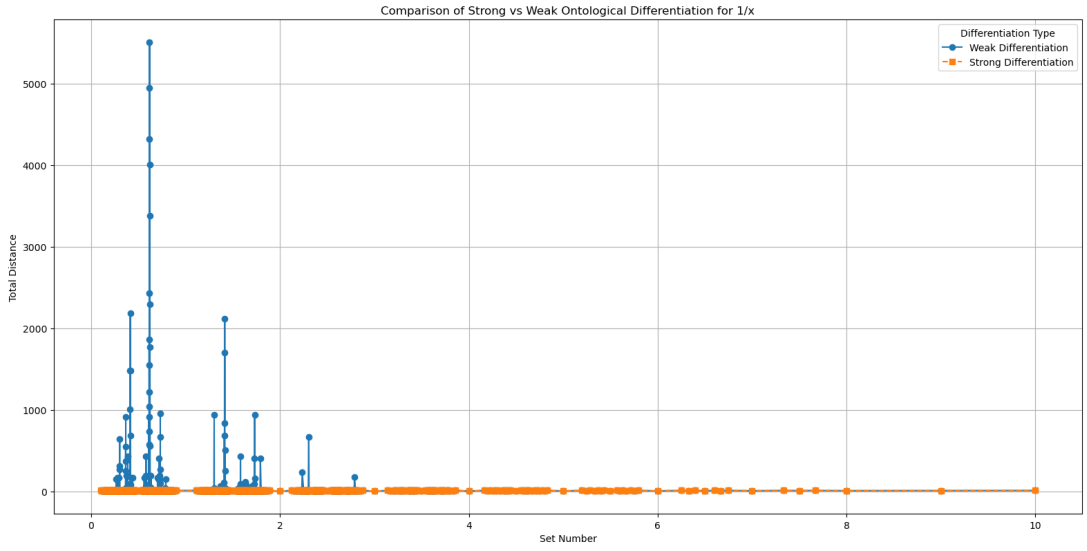

Figure 54: WOD and SOD for $1/x$ for iteration 1000.

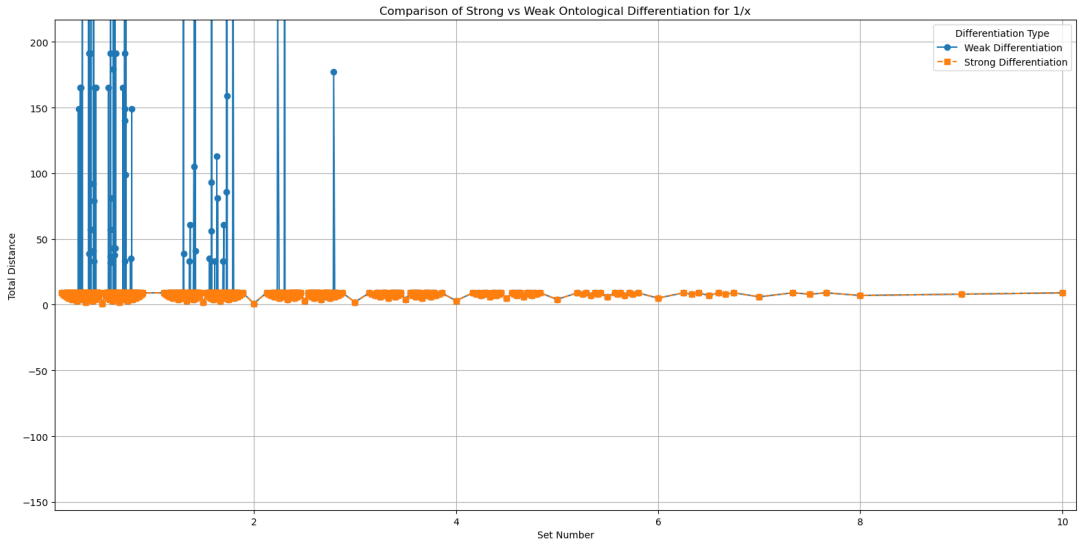

Figure 55: WOD and SOD for $1/x$ for iteration 1000, zoomed in.

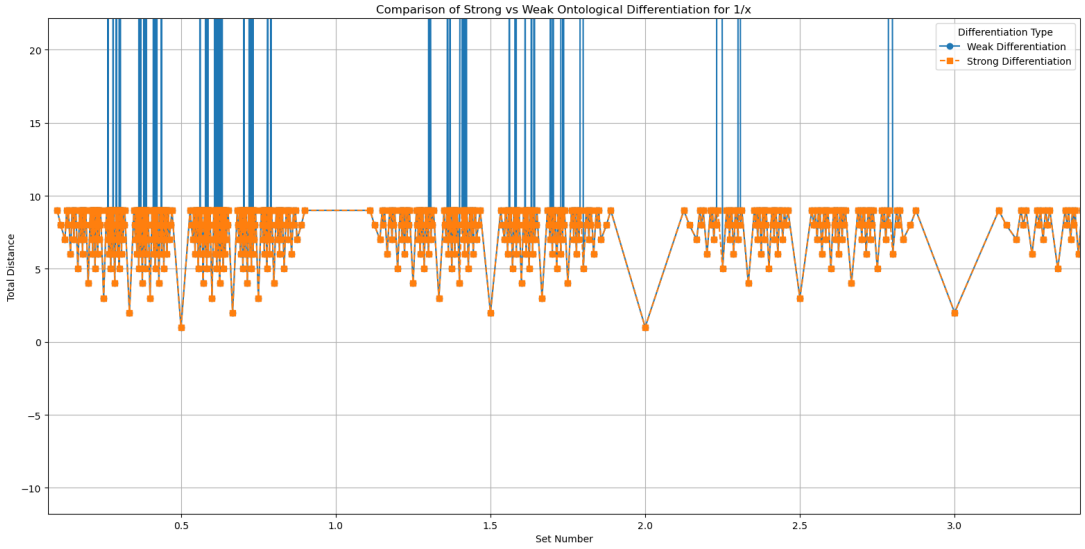

Figure 56: WOD and SOD for $1/x$ for iteration 1000, zoomed in further.

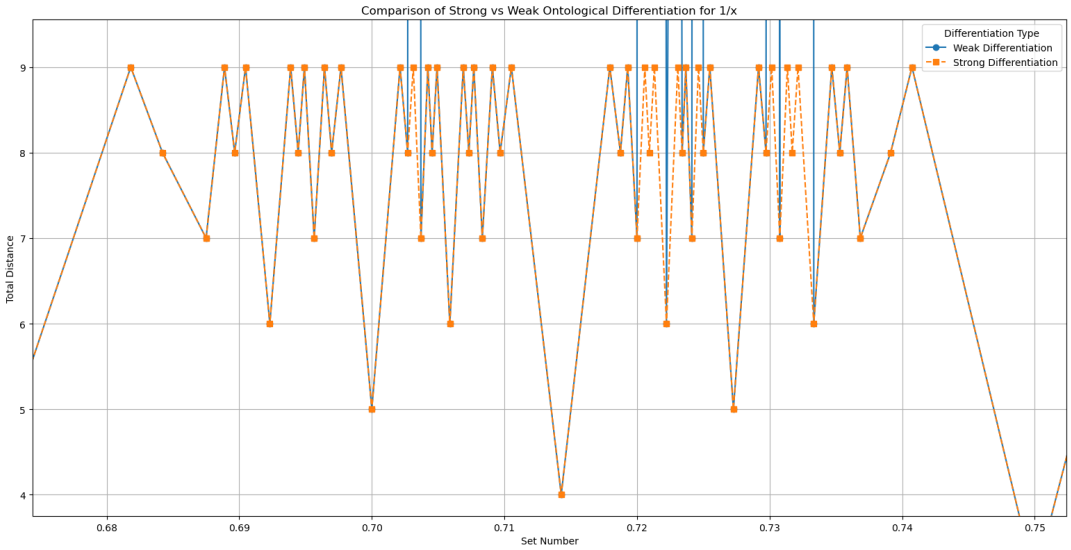

Figure 57: WOD and SOD for $1/x$ for iteration 1000, zoomed in further.

Now we see that the lower region of the WOD corresponds to the SOD. It is now clear that the WOD for $\frac{1}{x}$ is its SOD with occasional bursts of considerable size.

5.2. $f(x) = \sin x$

We will investigate the OD's applied to a recursive/infinite $\mathbf{U}$, ordered for the given structure S, and the function $f(x) = \sin x$, where our variable parameters will only be the size of $\mathbf{U}$, the size of the elements will be restricted to 2.

5.2.1. Results for WOD

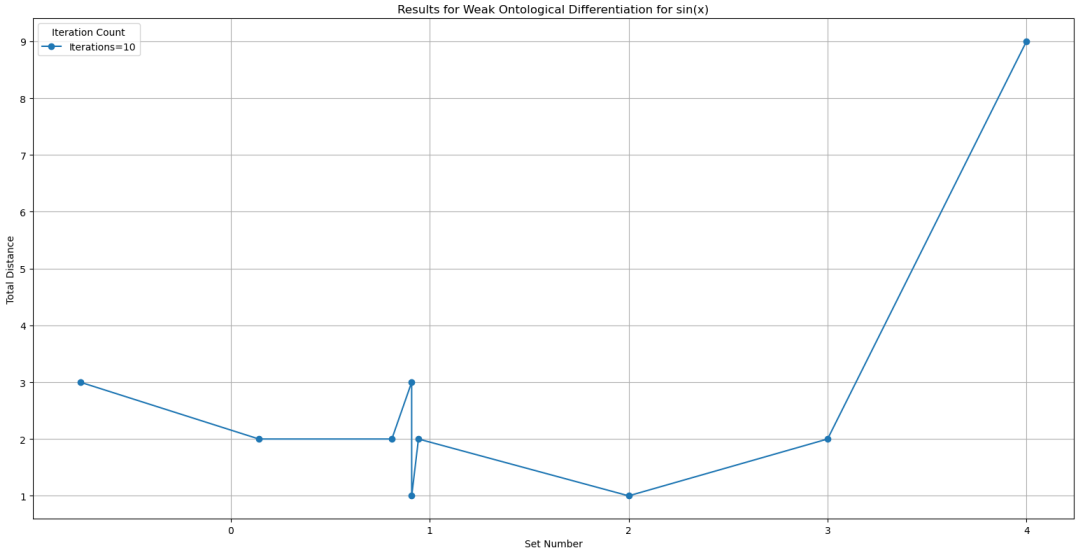

Figure 58: WOD for $\sin(x)$ for iteration 10.

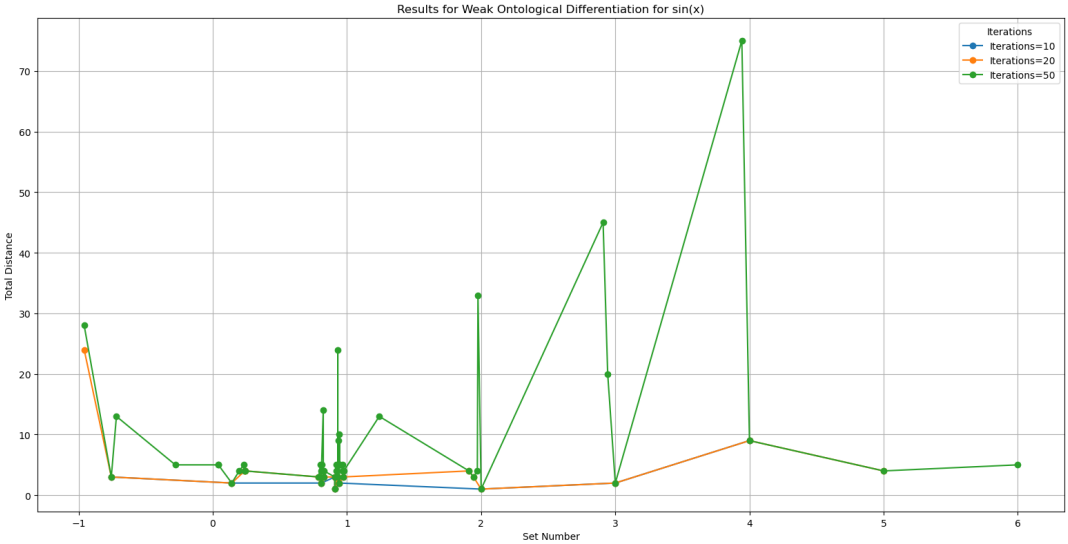

Figure 59: WOD for $\sin(x)$ for iterations 10, 20, 50.

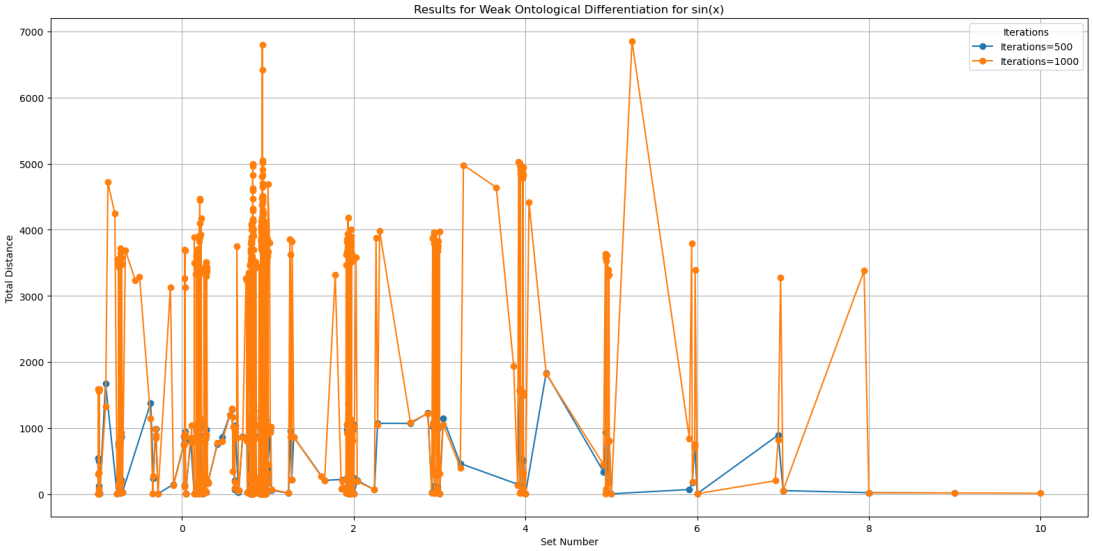

Figure 60: WOD for $\sin(x)$ for iterations 500, 1000.

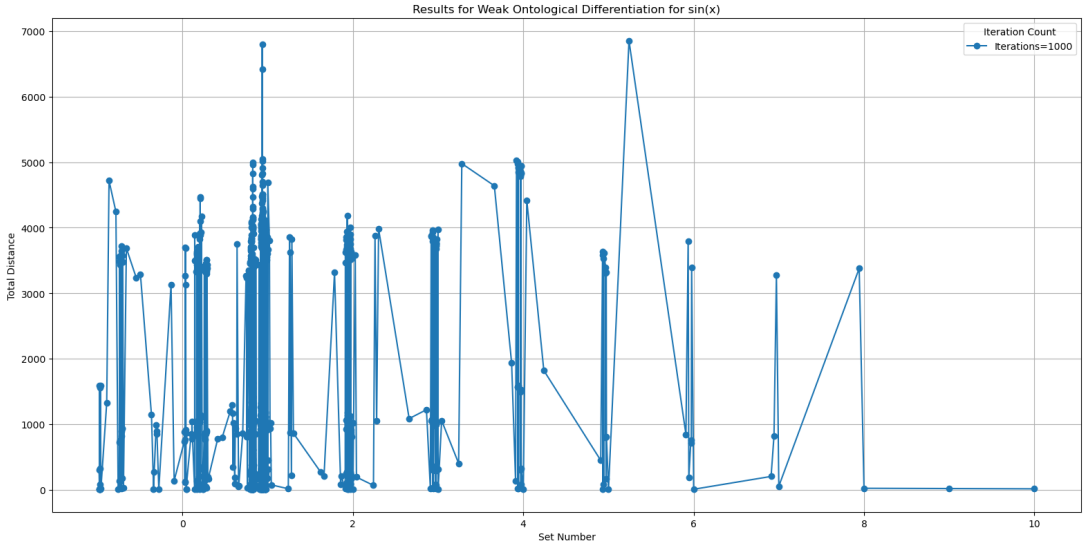

Figure 61: WOD for $\sin(x)$ for iteration 1000.

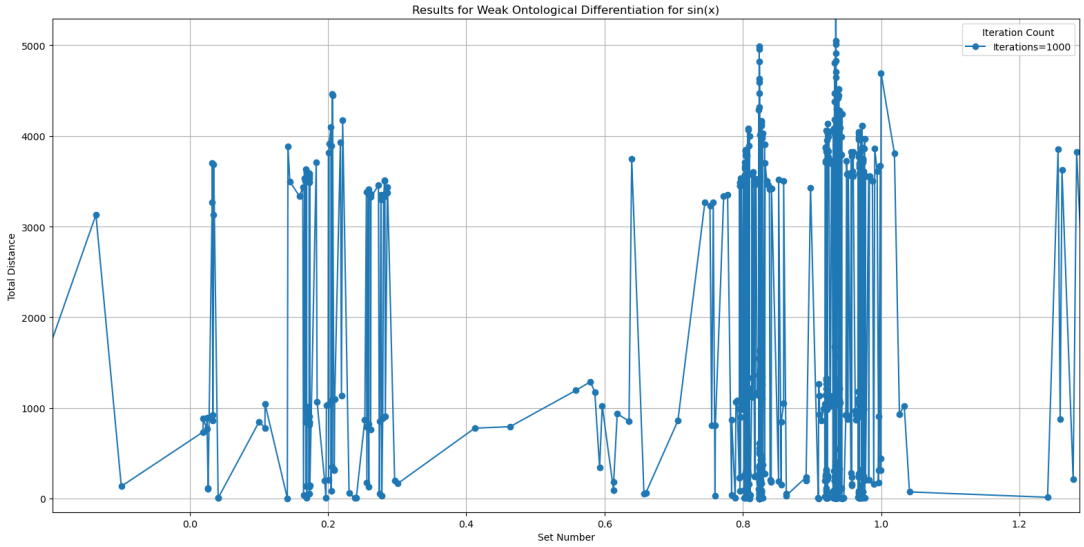

Figure 62: WOD for $\sin(x)$ for iteration 1000, zoomed in.

5.2.2. Results for SOD

The calculation of the SOD for the $\sin(x)$ function had unexpectedly an immensely expensive computational cost, which was manifested in extremely high values as we will see, therefore it was only calculated up to iteration 50. This is a rather peculiar phenomenon if we understand that it was not the case for its WOD and it was also no the case for the $\cos(x)$ function.

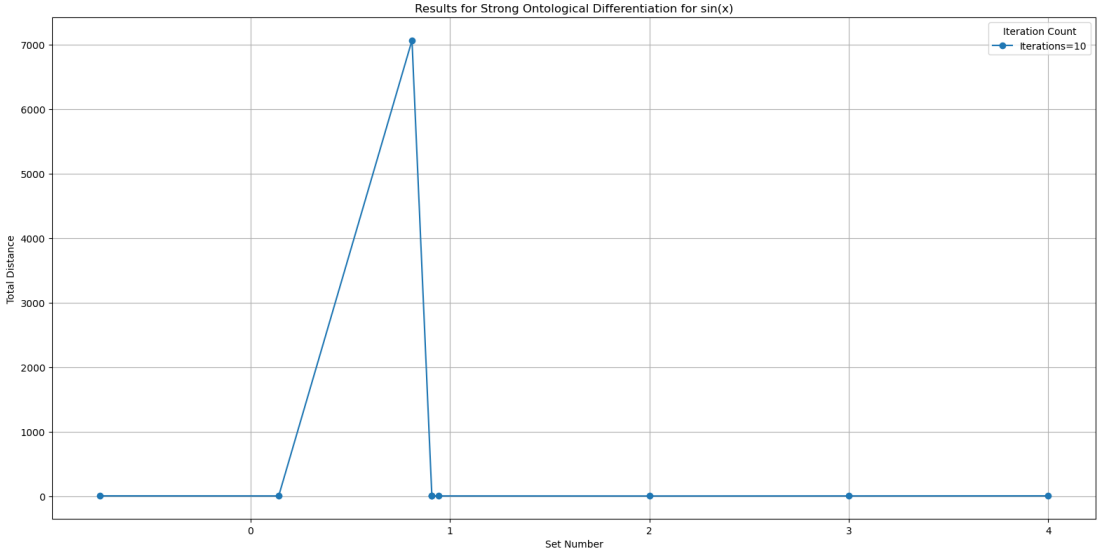

Figure 63: SOD for $\sin(x)$ for iteration 10.

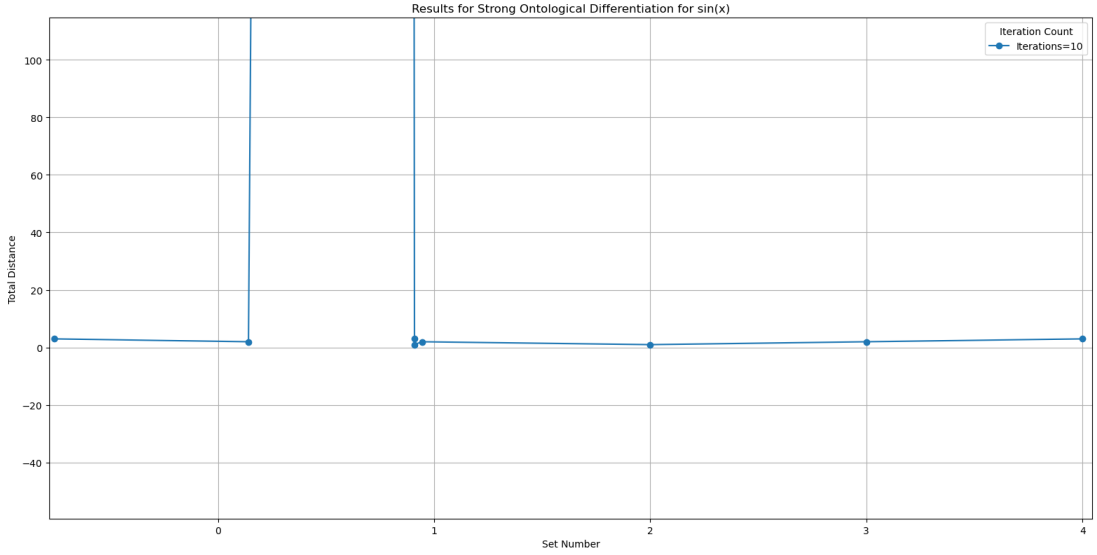

Figure 64: SOD for $\sin(x)$ for iteration 10, zoomed in.

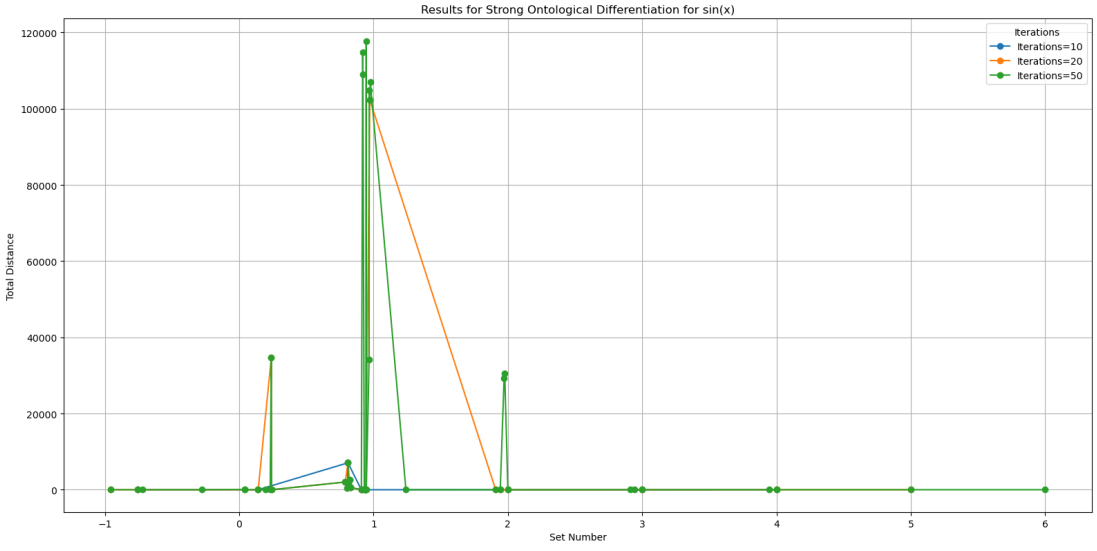

Figure 65: SOD for $\sin(x)$ for iterations 10, 20, 50.

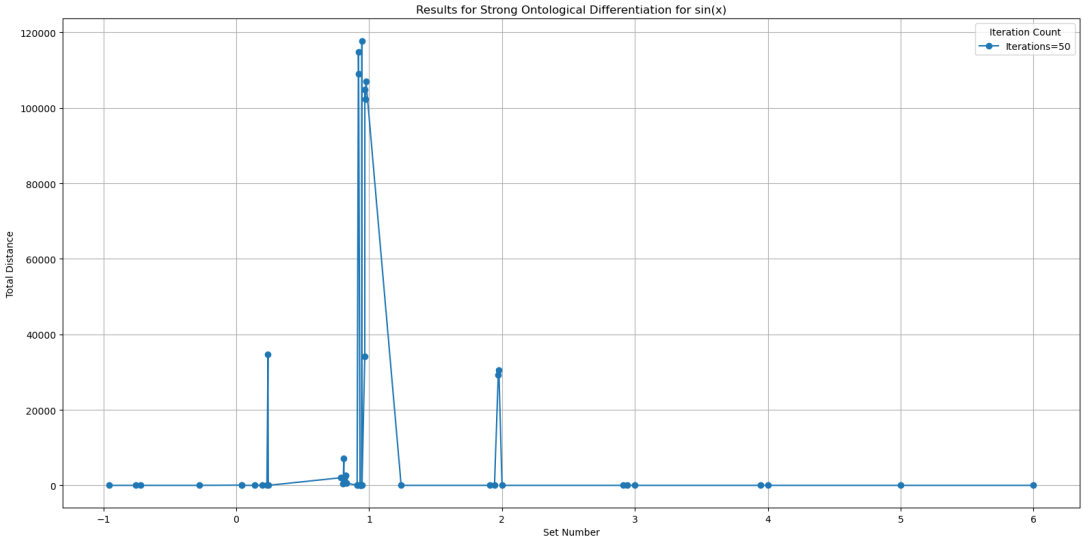

Figure 66: SOD for $\sin(x)$ for iteration 50.

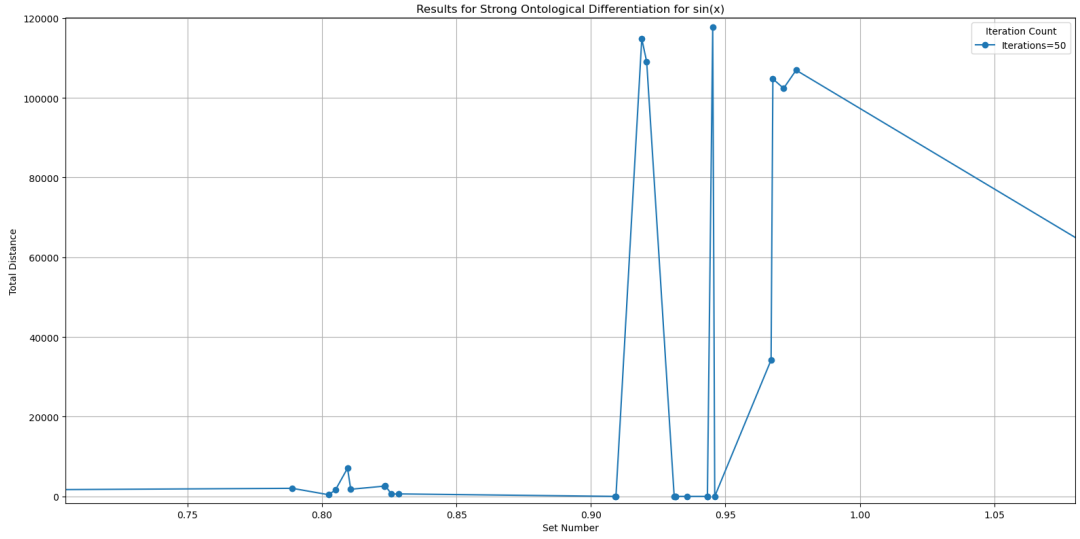

Figure 67: SOD for $\sin(x)$ for iteration 50, zoomed in.

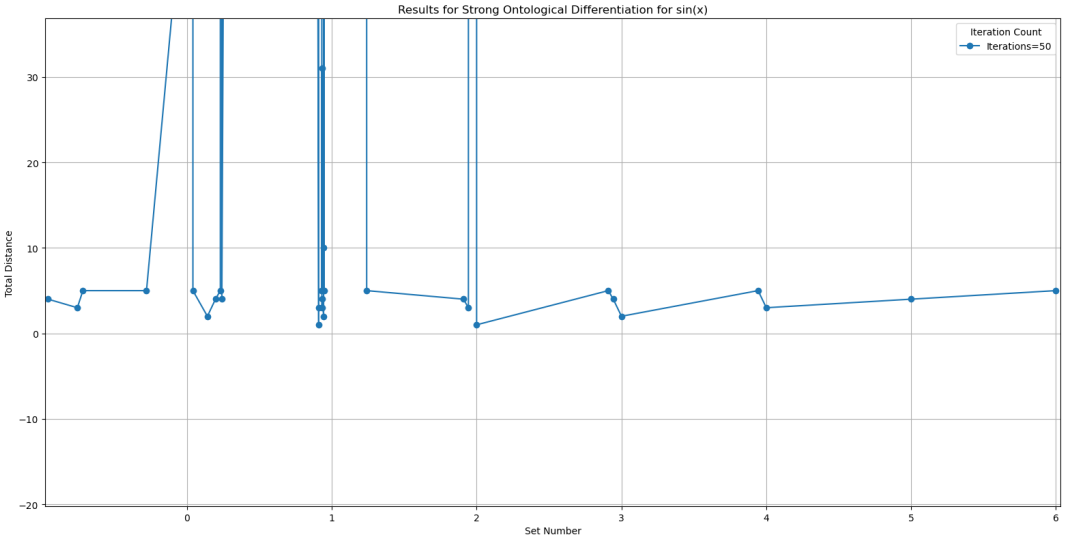

Figure 68: SOD for $\sin(x)$ for iteration 50, zoomed in the lower region.

5.3. $f(x) = \cos x$

5.3.1. Results for WOD

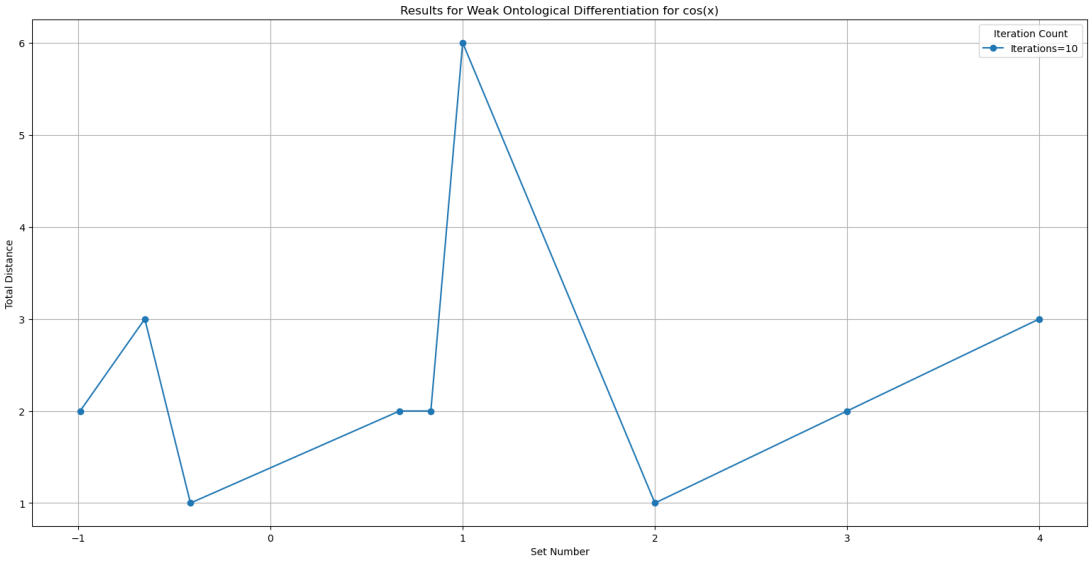

Figure 69: WOD for $\cos(x)$ for iteration 10.

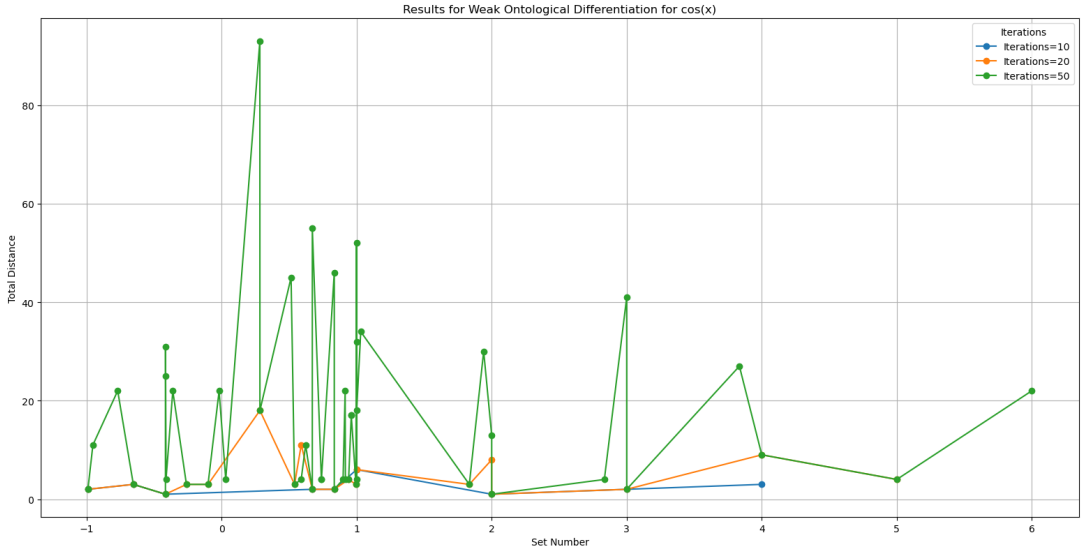

Figure 70: WOD for $\cos(x)$ for iteration 10, 20, 50.

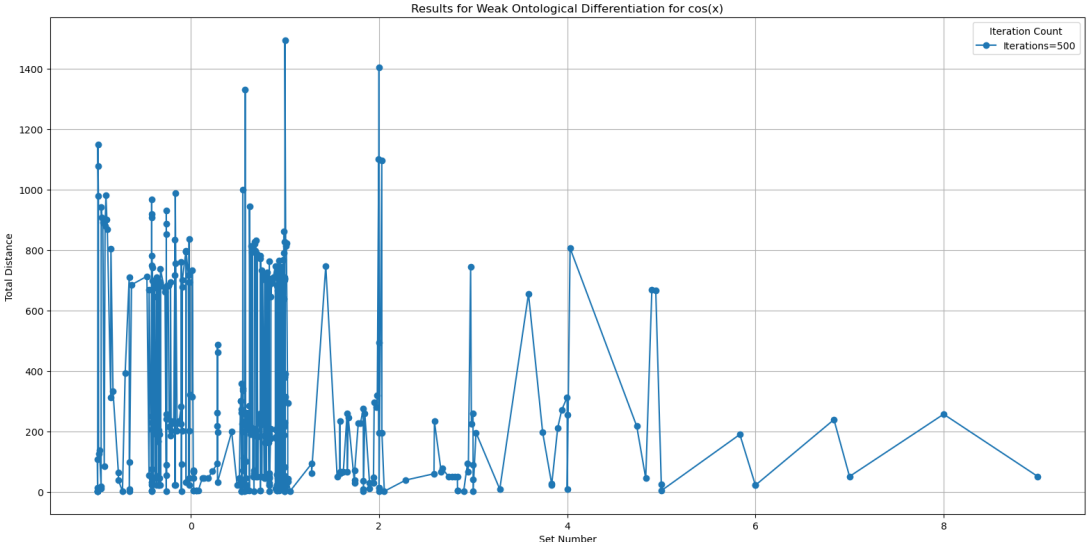

Figure 71: WOD for $\cos(x)$ for iteration 500.

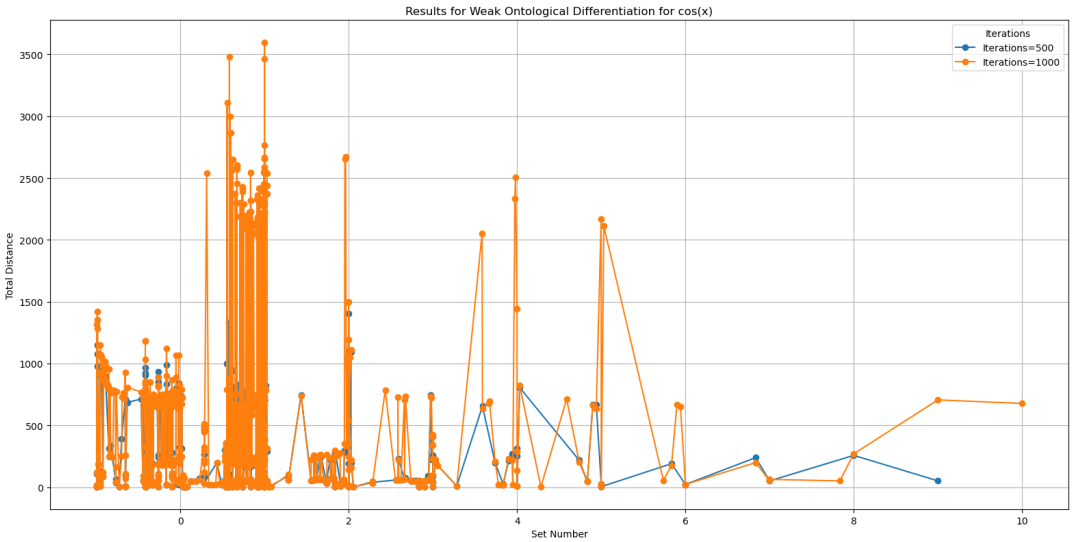

Figure 72: WOD for $\cos(x)$ for iteration 500, 1000.

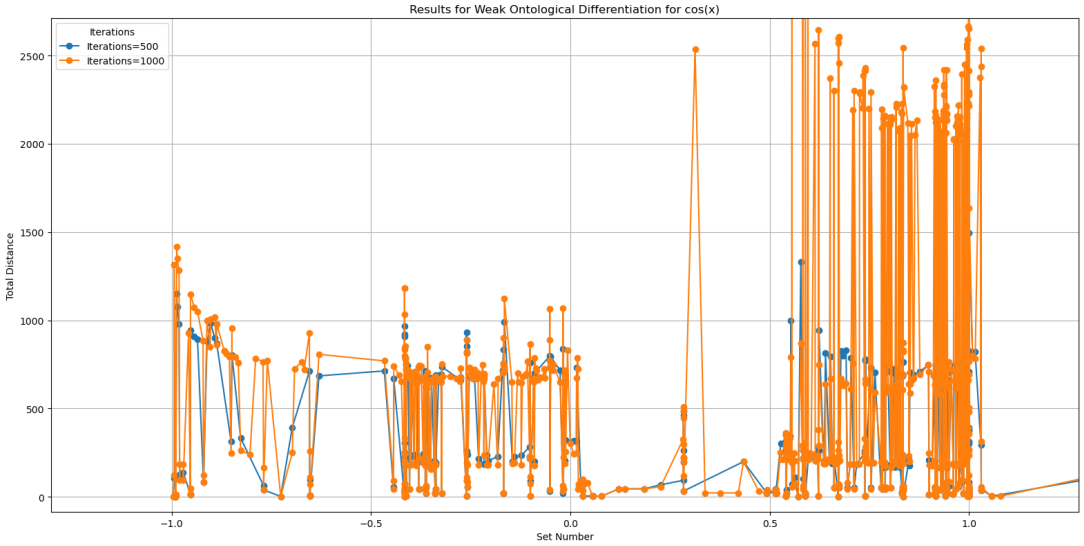

Figure 73: WOD for $\cos(x)$ for iteration 500, 1000, zoomed in.

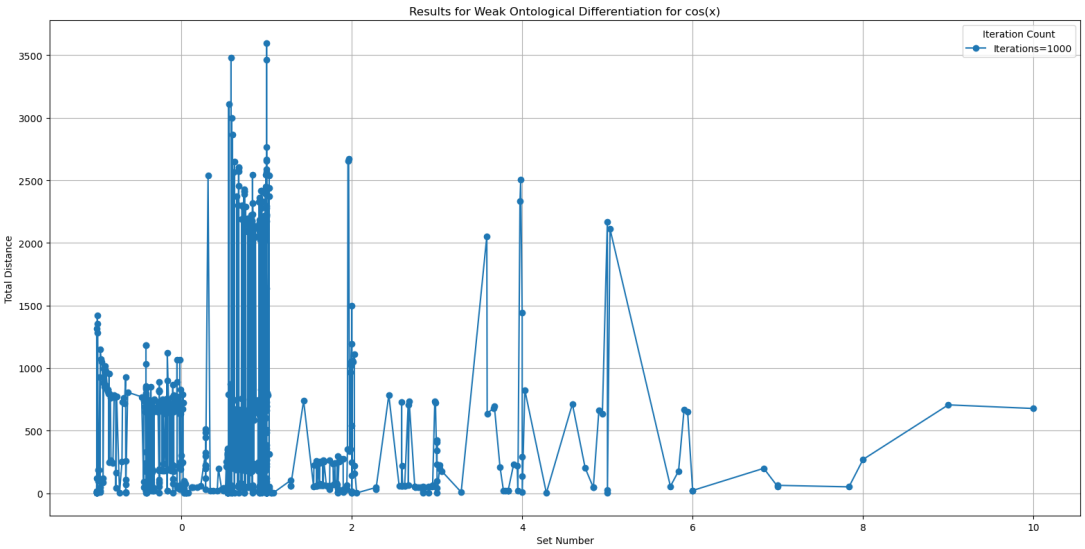

Figure 74: WOD for $\cos(x)$ for iteration 1000.

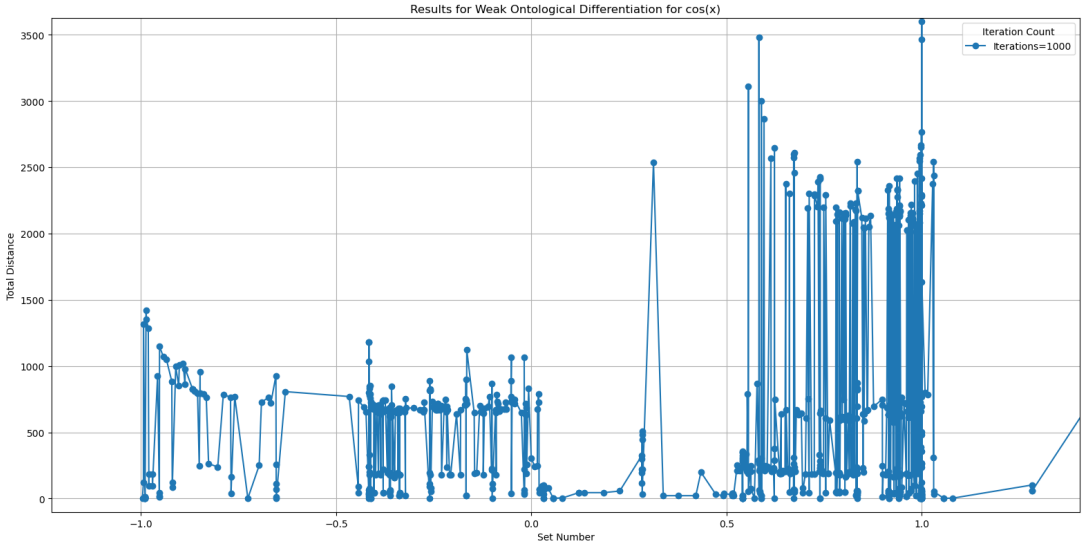

Figure 75: WOD for $\cos(x)$ for iteration 1000, zoomed in.

5.3.2. Results for SOD

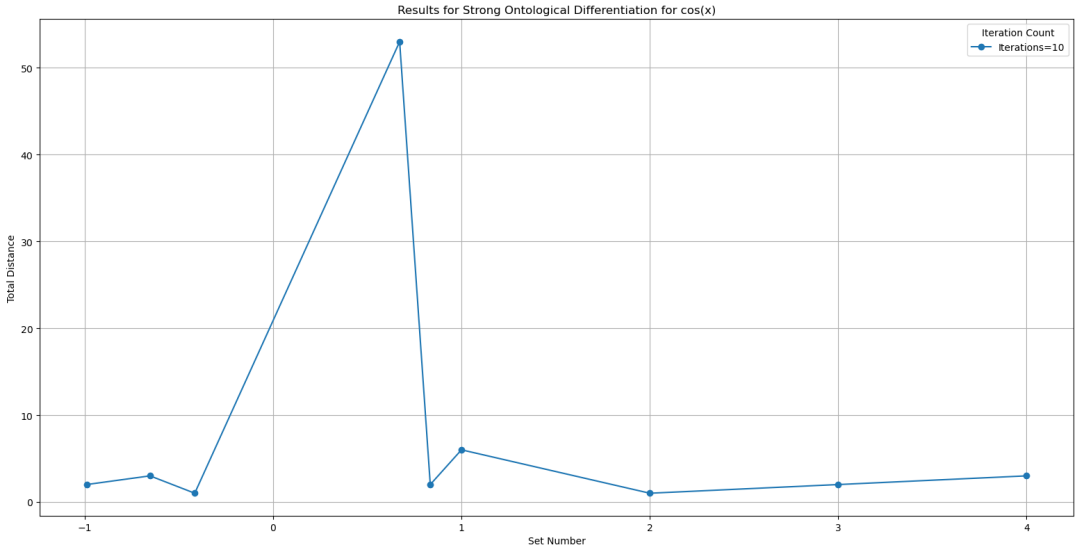

Figure 76: SOD for $\cos(x)$ for iteration 10.

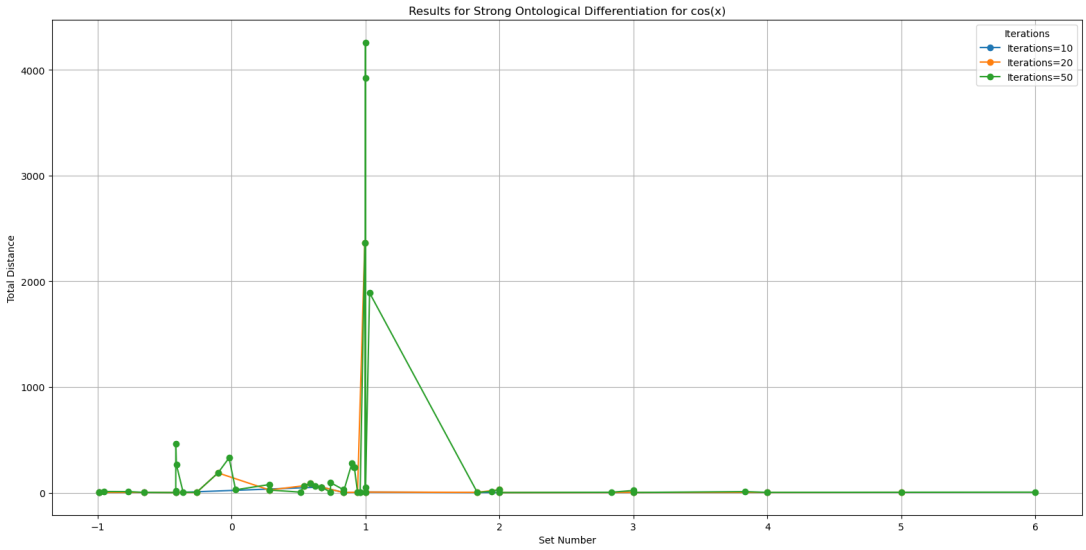

Figure 77: SOD for $\cos(x)$ for iterations 10, 20, 50.

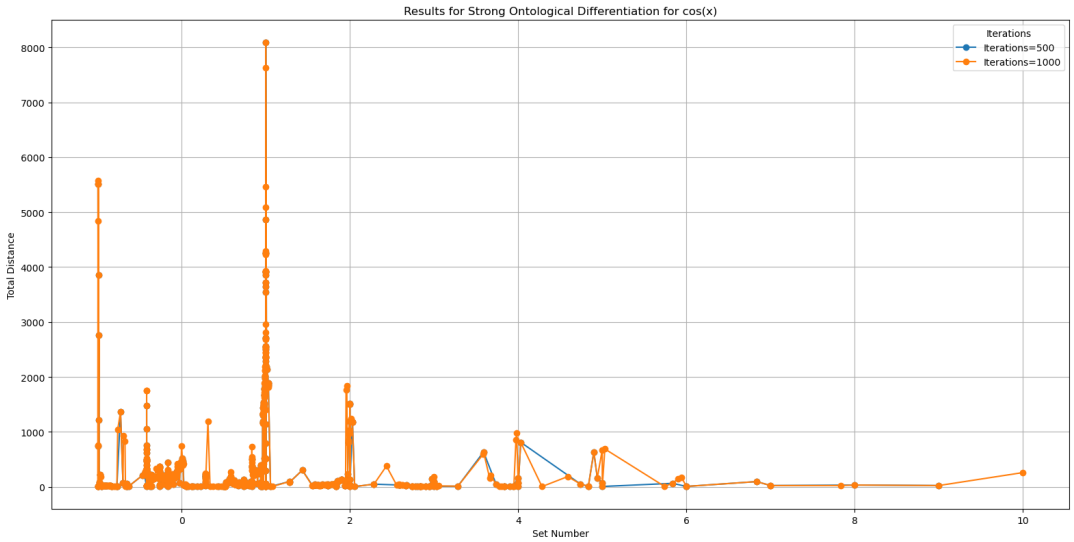

Figure 78: SOD for $\cos(x)$ for iterations 500, 1000.

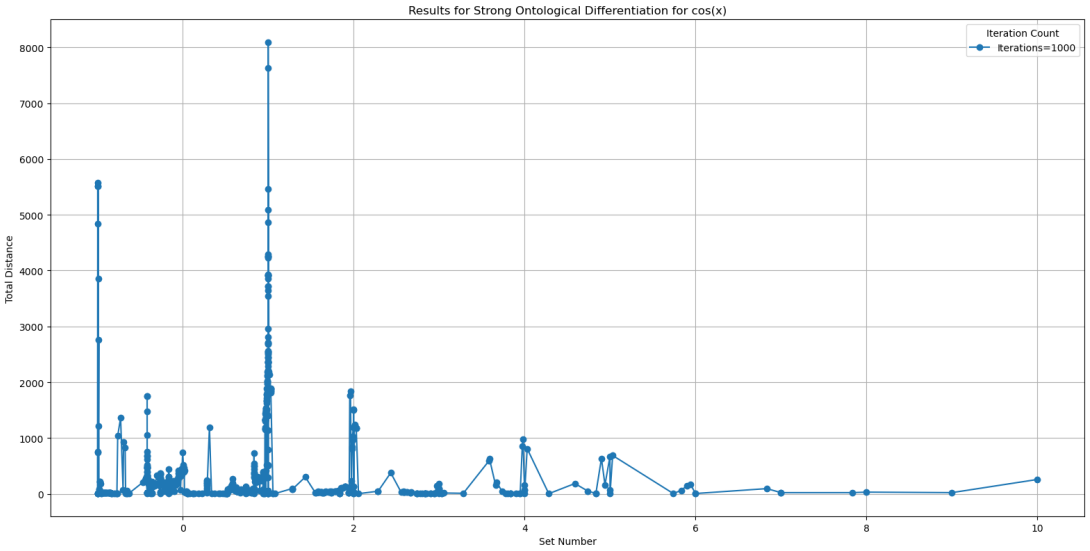

Figure 79: SOD for $\cos(x)$ for iteration 1000.

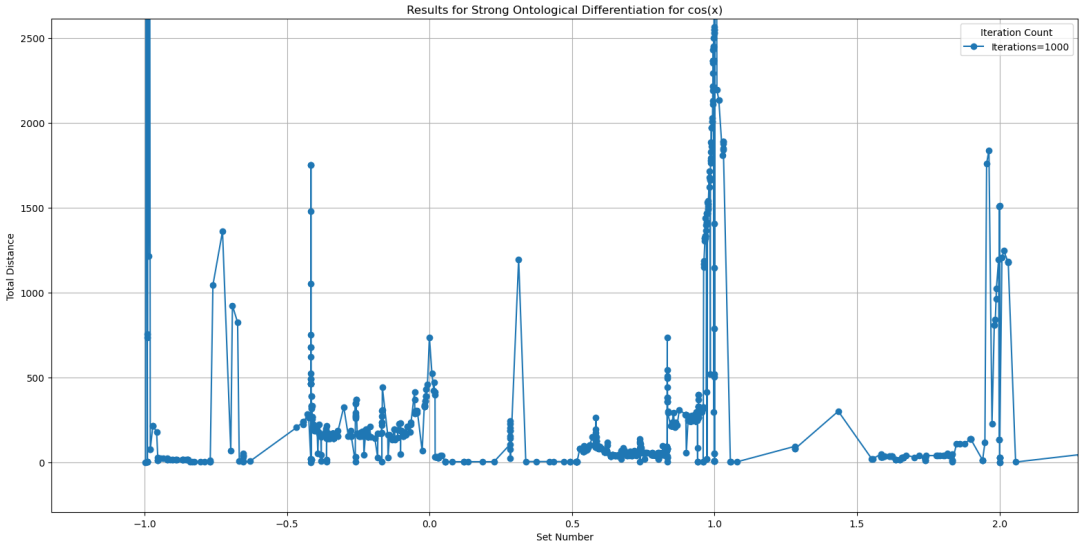

Figure 80: SOD for $\cos(x)$ for iteration 1000, zoomed in.

Now let us do a quick comparison of the results between the WOD and SOD:

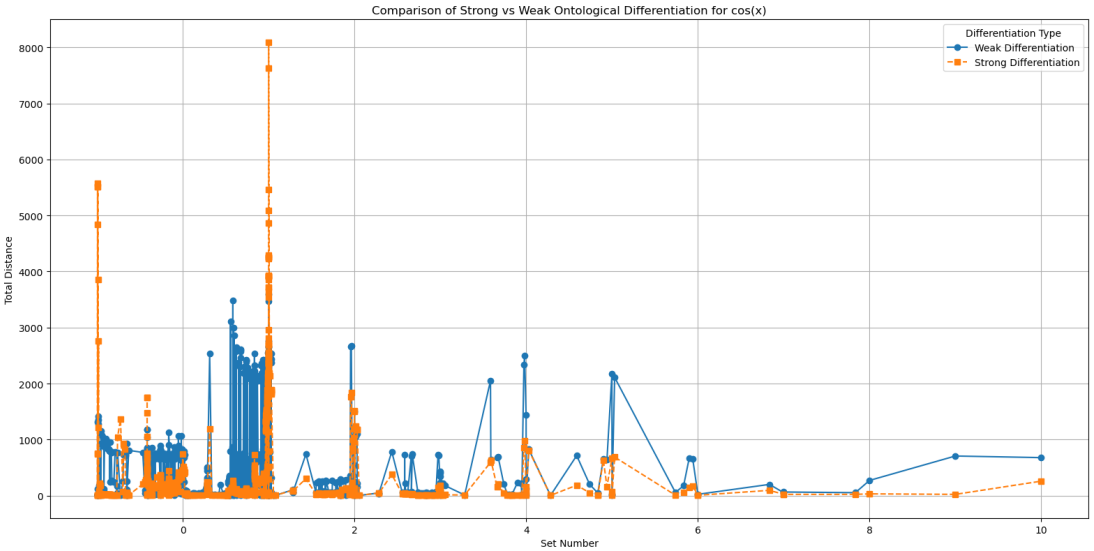

Figure 81: WOD and SOD for $\cos(x)$ for iteration 1000.

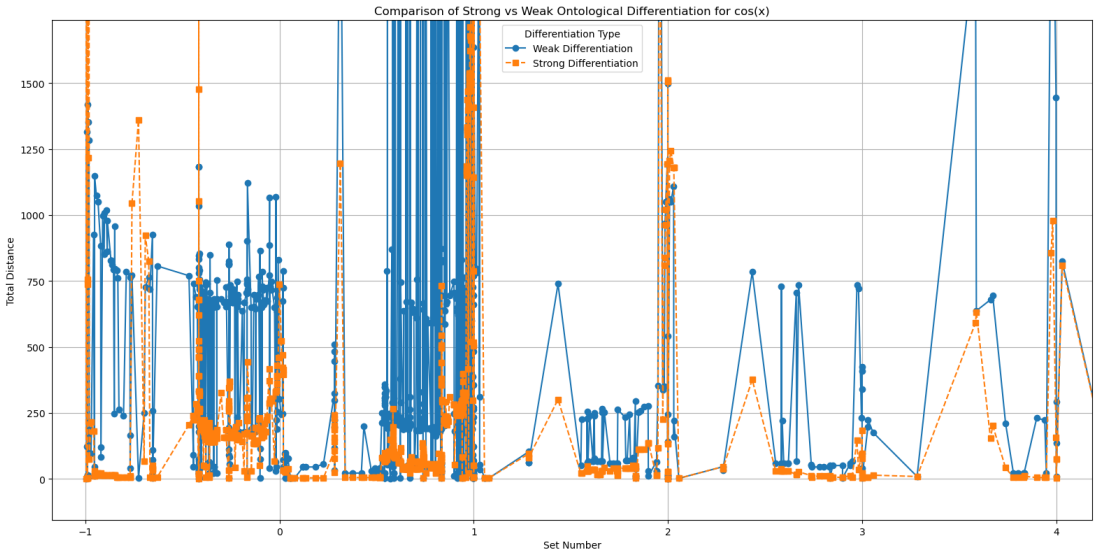

Figure 82: WOD and SOD for $\cos(x)$ for iteration 1000, zoomed in.

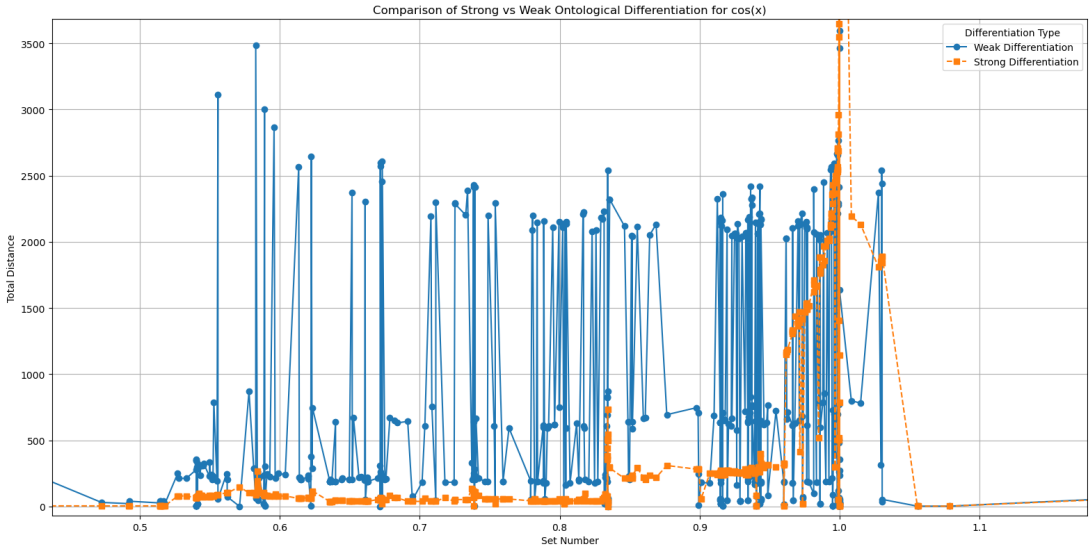

Figure 83: WOD and SOD for $\cos(x)$ for iteration 1000, zoomed in.

Finally, it would also be interesting to compare the results for $\cos(x)$ and $\sin(x)$, therefore let us have the two following plots to see a direct comparison:

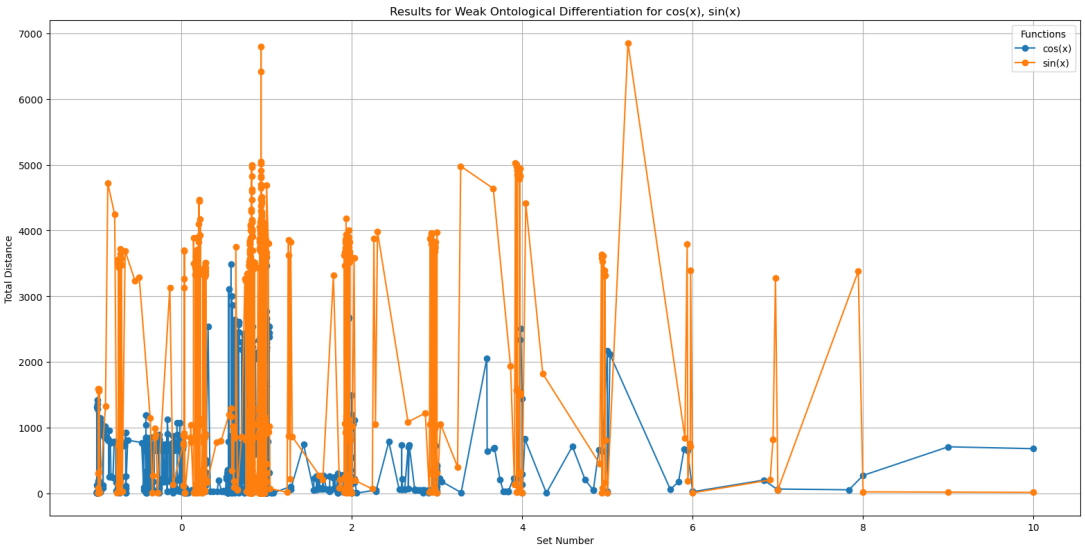

Figure 84: WOD for $\cos(x)$ and $\sin(x)$ for iteration 1000.

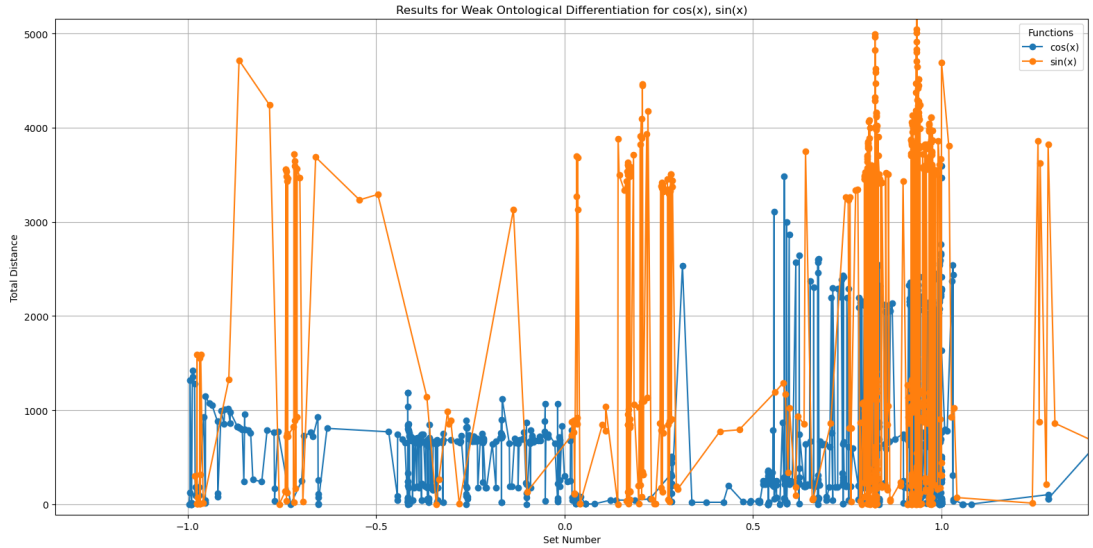

Figure 85: WOD for $\cos(x)$ and $\sin(x)$ for iteration 1000, zoomed in.

5.4. $f(x) = x + k$

We will investigate the OD's applied to a recursive/infinite $\mathbf{U}$, ordered for the given structure S, and the function $f(x) = x + k$ (we have come back to this function for this recursive version), where our variable parameters are: k in the function, the size of $\mathbf{U}$ and the size of its elements.

5.4.1. Results for Decimals

Here we have selected the decimals for the range $[1, 2]$. It must be noted that except for $k = 1.5$, the OD diverges to infinity since the structure of $\mathbf{U}$ given from the 6th iteration onwards makes it impossible for the read function to reach the last set number from the first set.

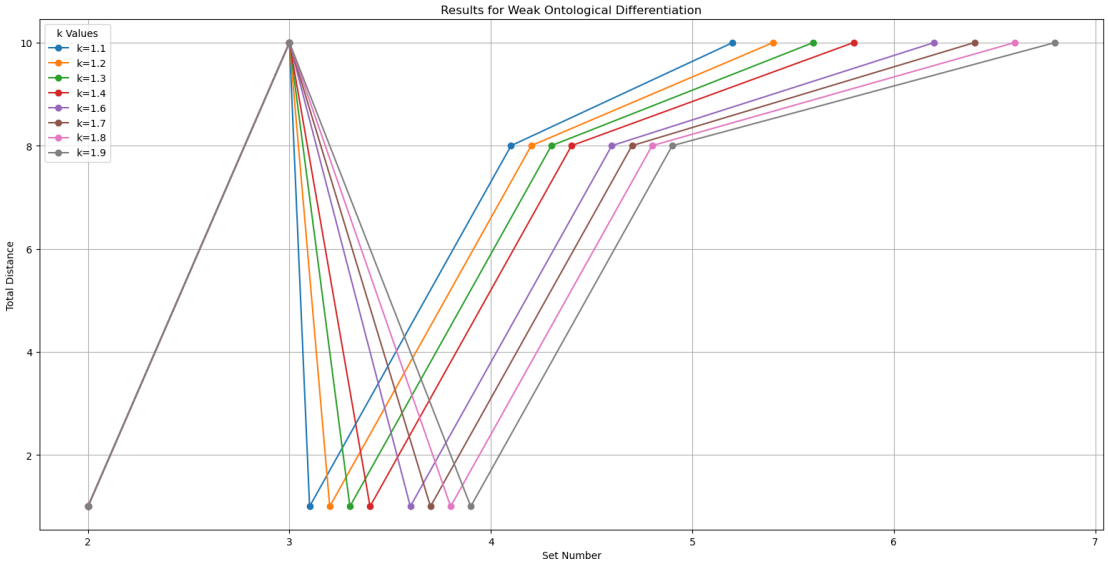

Figure 86: WOD for k as first decimals.

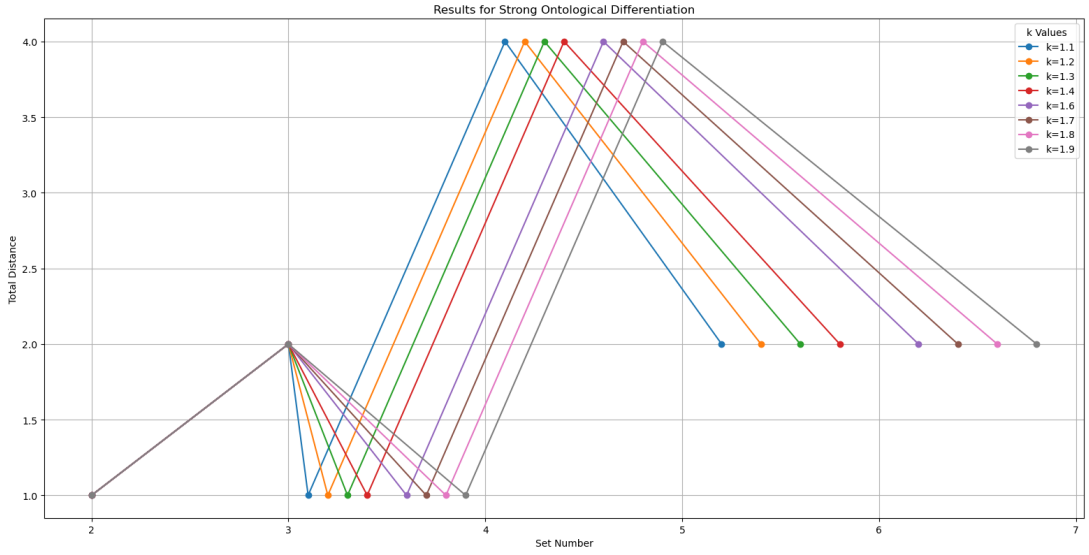

Figure 87: SOD for k as first decimals.

For the following case where $k = 1.5$, one can quickly check that the read function

will always be able to reach whichever set our set 1 is compared to, however the computational cost also will also grow as the number of iterations do.

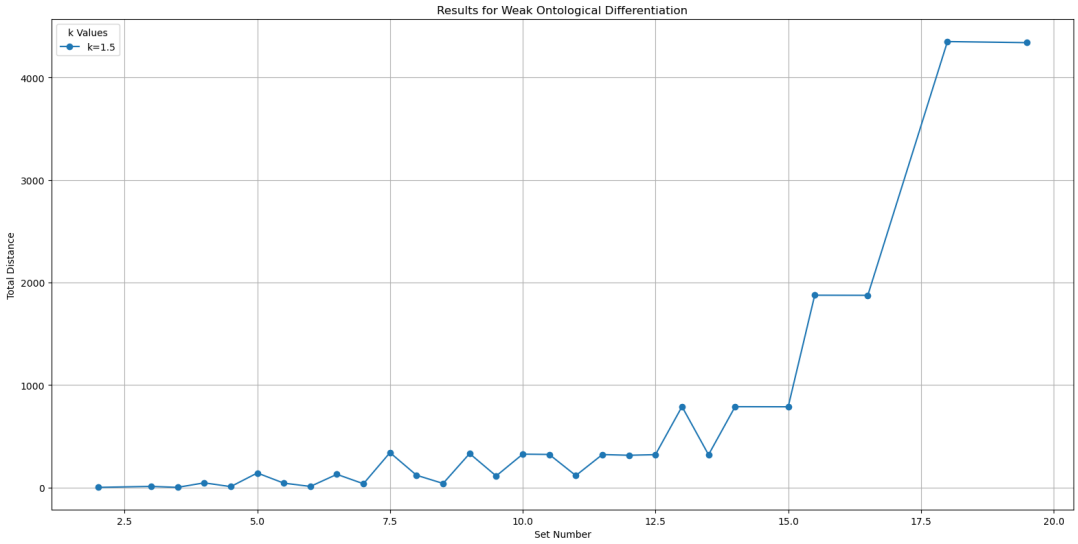

Figure 88: WOD for $k = 1.5$

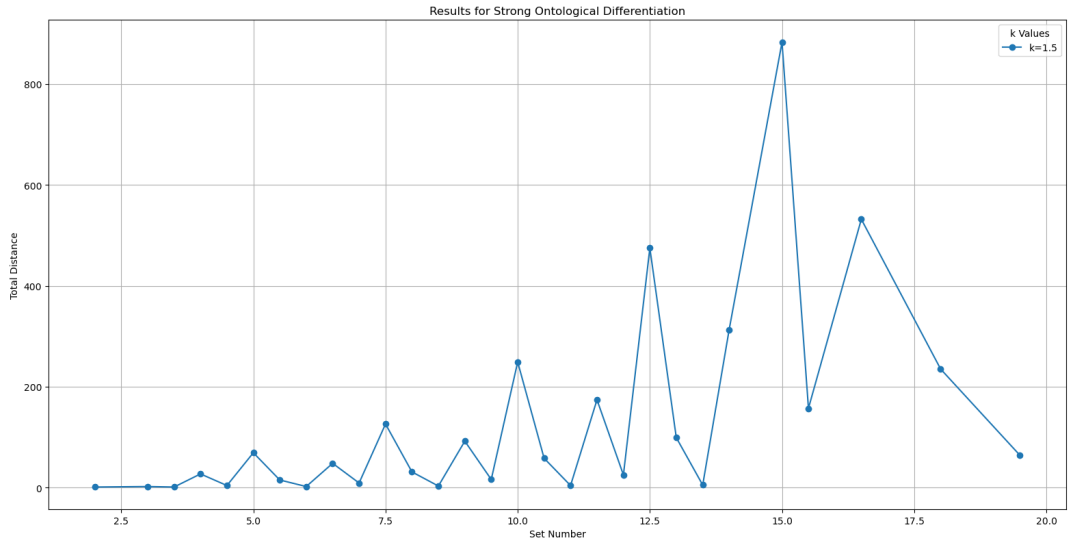

Figure 89: SOD for $k = 1.5$

To illustrate as an example why the 6th iteration is a limit for convergence of any OD for the aforementioned decimals, let us take $k = 1.1$ and show the generated sets and the last SOD for the 6th iteration:

Example 15. *The generated sets, with their SOD results, are:*

- *Set 1 vs Set 2: 1*

- *Set 1 vs Set 3: 2*
- *Set 1 vs Set 3.1: 1*
- *Set 1 vs Set 4.1: 4*
- *Set 1 vs Set 5.2: 2*

For the 7th iteration, the next set to deal with will be 6.3, for which we will have the following:

Example 16. *Set 1 with Set 6.3:*

- *Level 1:* $[2, 3.1]; [7.3, 8.4]$
- *Level 2:* $[3, 4.1, 4.1, 5.2]; [8.3, 9.4, 9.4, 10.5]$
- *Level 3:* $[4, 5.1, 5.1, 6.2, 5.1, 6.2, 6.2, 7.3]; [9.3, 10.4, 10.4, 11.5, 10.4, 11.5, 11.5, 12.6]$

As one can see, no expansion of the read function of the left side, set 1, will completely cancel one side of any level for set 6.3, not even level 0, for in order for the read function to produce 6.3 one must expand 4.2 and this element is not (and obviously will not) be present in any level of the expansion. Therefore we have that the SOD will diverge.

This can of course be easily solved by changing the structure and have only already appeared sets to do the comparison with, but as we said, in this recursive function section we are just following one single structure for better coherence. One can change the structure as much as one wishes to obtain different results and aim at different goals.

5.4.2. Results for Integers

Here we have selected the first five integers as values for k. It must be noted that while in this case it is possible for the read function to reach any set number our set 1 is compared to, the computational cost is very expensive as the number of iterations grow, so the iterations here shown are restricted to modest numbers.

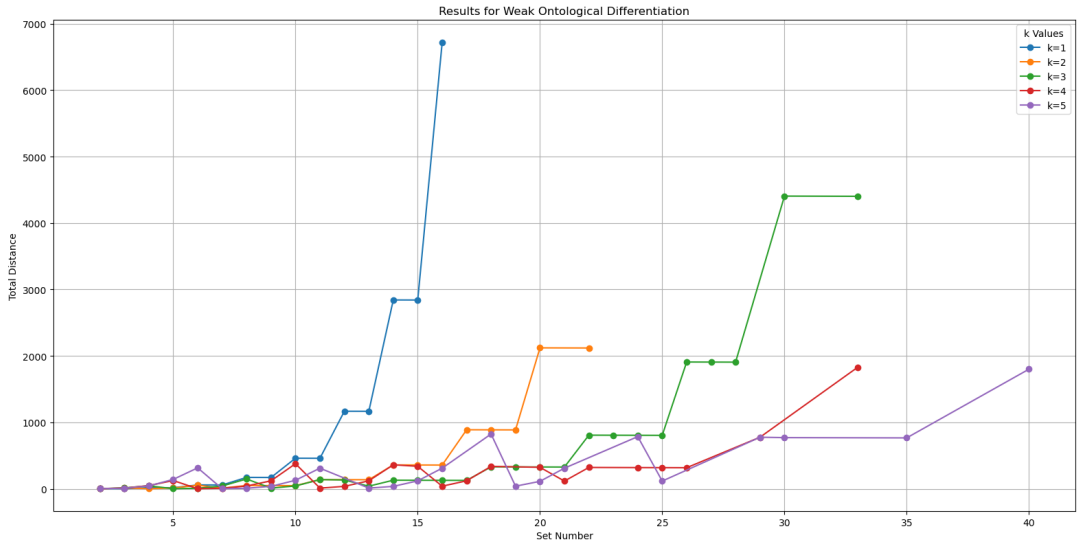

Figure 90: SOD for k as first integers.

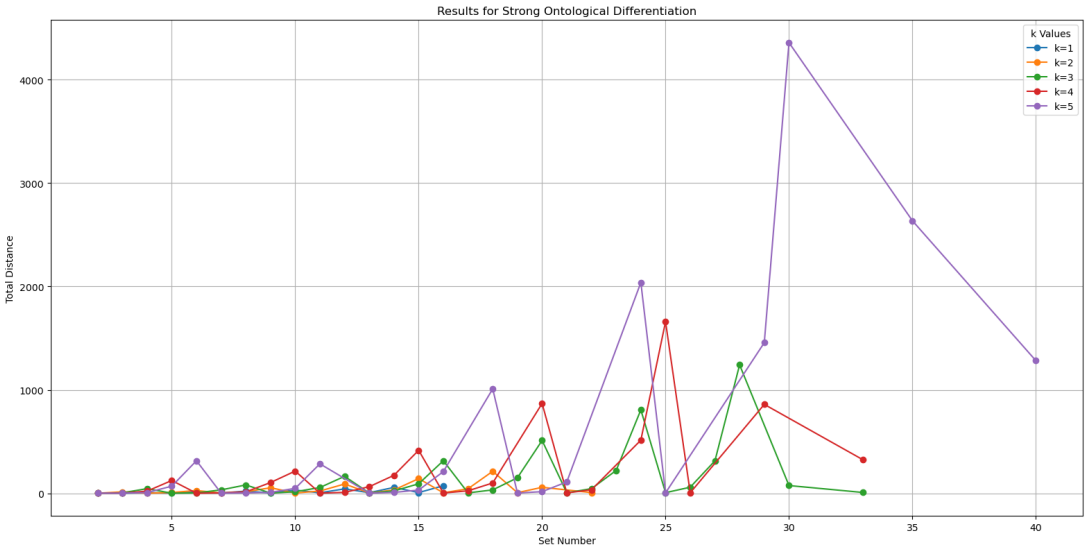

Figure 91: SOD for k as first integers.

6. Randomness

In this section we will study **U** manifold whose element's order is given by a random process. Their members will be generated randomly (using NumPy) restricted to a size of **U** range.

When it comes to the constraints under which these sets are generated, there are many which we will now present here, but all of them have in common that no set

can contain itself as fundamental constraint. The different types are:

-**Unconstrained**: When generating sets, there are no specific rules or restrictions applied. The elements in each set are chosen randomly without any regard for previous selections or any other criteria.

-**Regular Uniqueness**: Under this constraint, each set generated must be unique in terms of its content compared to the previously generated sets. The sets are checked to ensure that there are no exact duplicates in the order of elements. However, if the same elements appear in different orders, these are considered different sets under this constraint.

-**Strict Uniqueness**: This is a stricter version of Regular Uniqueness. It requires that each set is unique not just in terms of its exact content but also considering the order of elements. That means even permutations of the same elements are not allowed.

-**Weighted** : In this constraint, certain elements have a higher likelihood of being selected based on a predefined weight. These weights influence the random selection process, making it more likely for some elements to appear in sets than others. The weight factor determines the degree to which these elements are favored during the selection process.

-**Fixed Size**: In this constraint, all sets are generated with an identical number of elements. For example, if you specify that each set should contain 3 elements, then every set created will have exactly 3 elements, regardless of other factors. This ensures uniformity across all sets, with each one containing the same number of items.

-**Irregular Size**: Under this constraint, sets are allowed to vary in size within a specified range. Instead of every set having the same number of elements, each set can have a different number of elements, as long as they fall within a defined minimum and maximum range. This introduces flexibility, allowing for some sets to be larger or smaller than others.

-**Continuation (True or False)** : This constraint determines whether the set generation should continue to expand or stop after a fixed number of sets. When continuous=True, after the initial set of elements are generated, the function keeps expanding the sets by adding more elements according to the rules defined by the selected constraint. When continuous=False, the set generation stops after creating a fixed number of sets without further expansion. After the initial set generation, if continuous=True, the code extends the set generation process by generating additional sets (num new sets). The continuous generation function combines existing

sets with newly generated sets. The new sets are created similarly to the initial sets but are appended to the existing list. This extension allows the set generation process to evolve beyond the initial configuration. The total number of sets after this process is num sets + num new sets.

The following results are given by a num sets = 100 and num elements = 2 and for 100 iterations, that is, we have generated 100 random sets and done each specific operation each of those 100 times and then they are put together on these graphs. As for the weighted value, we have chosen sets 2 and 5 with a weight factor of 10.

6.1. Results for Unconstrained

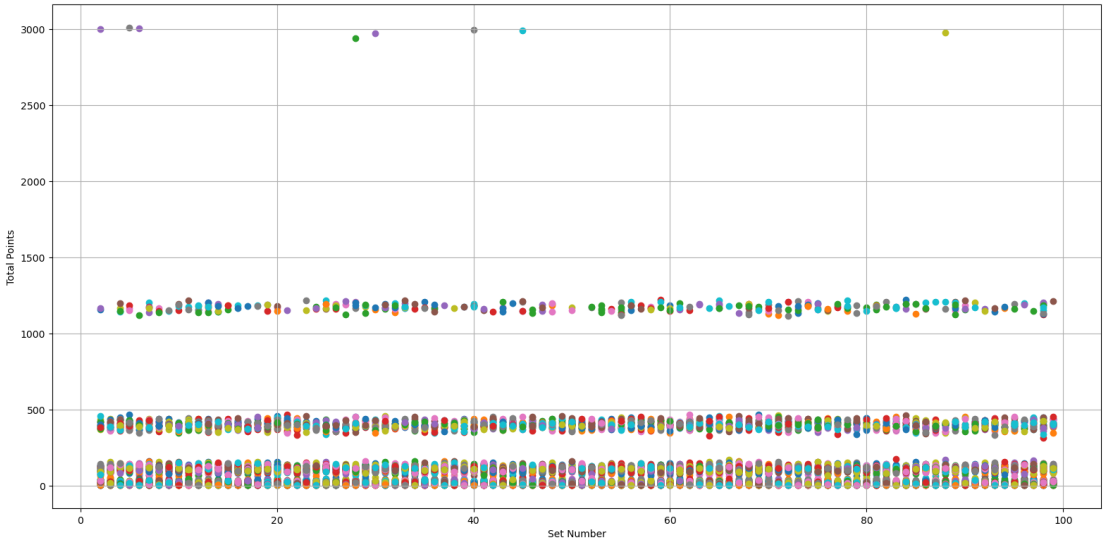

Figure 92: WOD for Unconstrained with fixed size, continous = False, one vs all, for a num sets = 100 and num elements = 2 for 100 iterations.

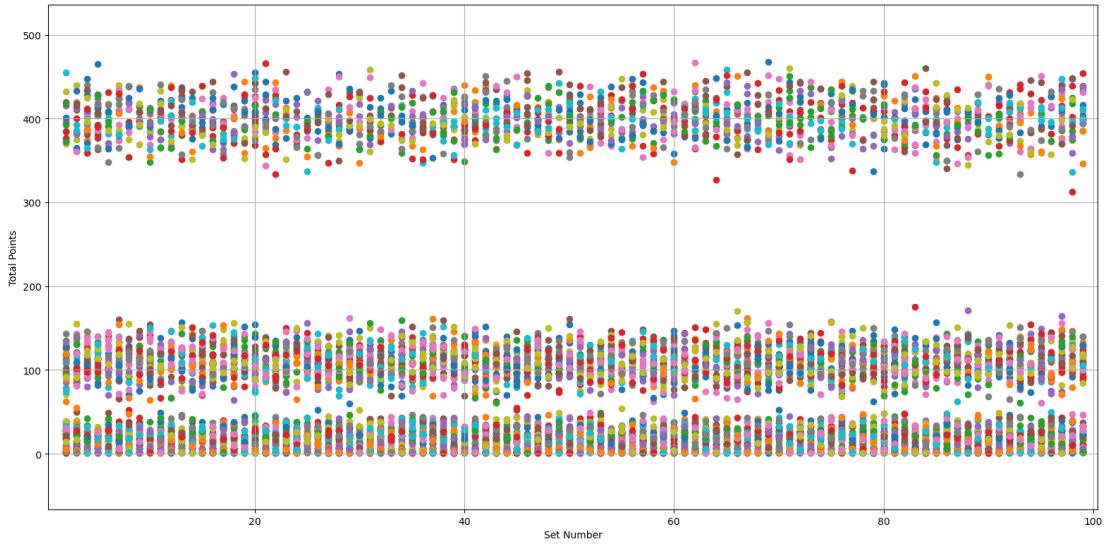

Figure 93: *WOD* for Unconstrained with fixed size, continous = False, one vs all, for a num sets = 100 and num elements = 2 for 100 iterations, zoomed in.

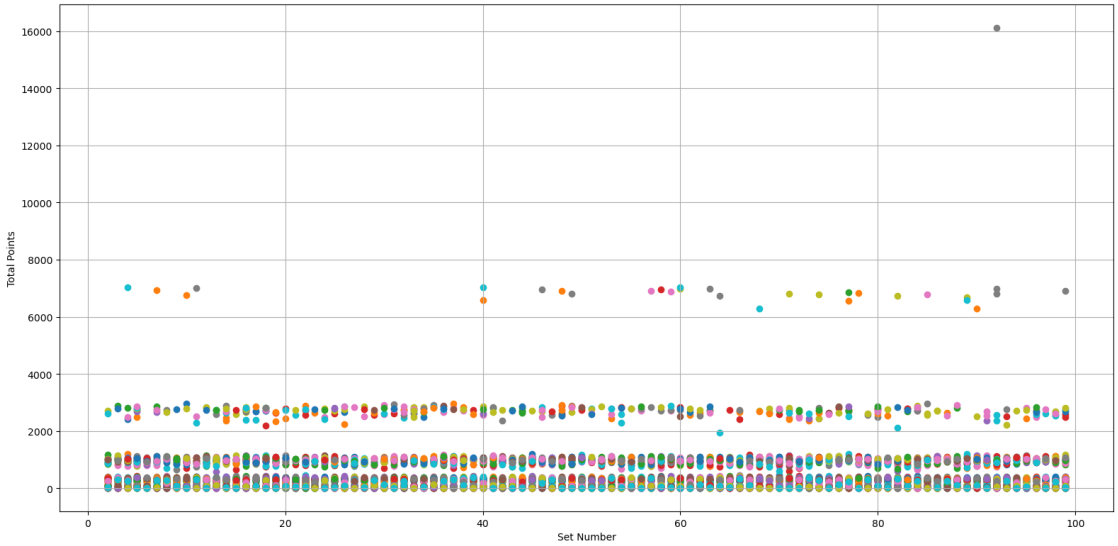

Figure 94: *SOD* for Unconstrained with fixed size, continous = False, one vs all, for a num sets = 100 and num elements = 2 for 100 iterations.

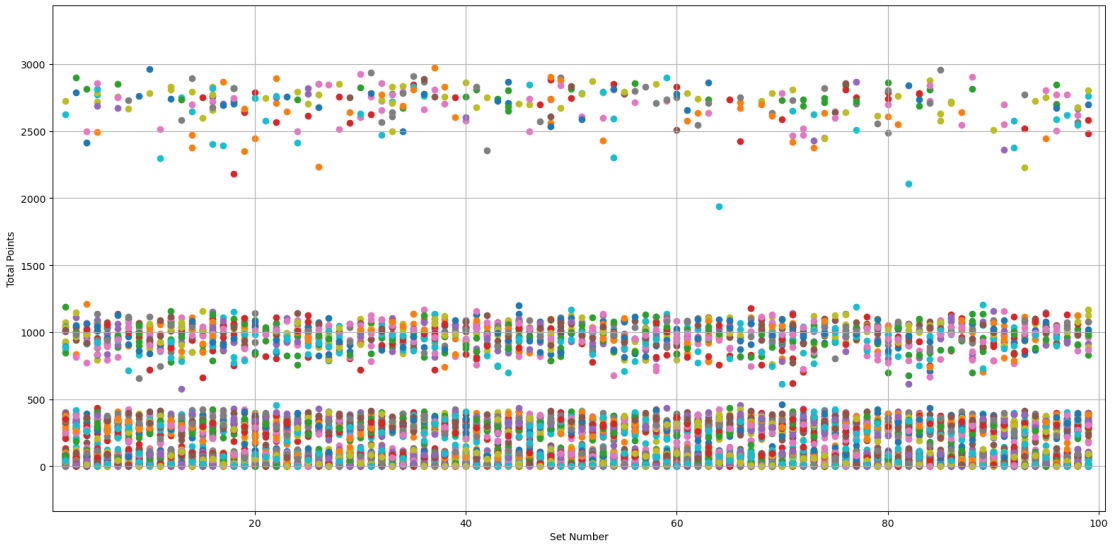

Figure 95: *SOD* for Unconstrained with fixed size, continous = False, one vs all, for a num sets = 100 and num elements = 2 for 100 iterations, zoomed in.

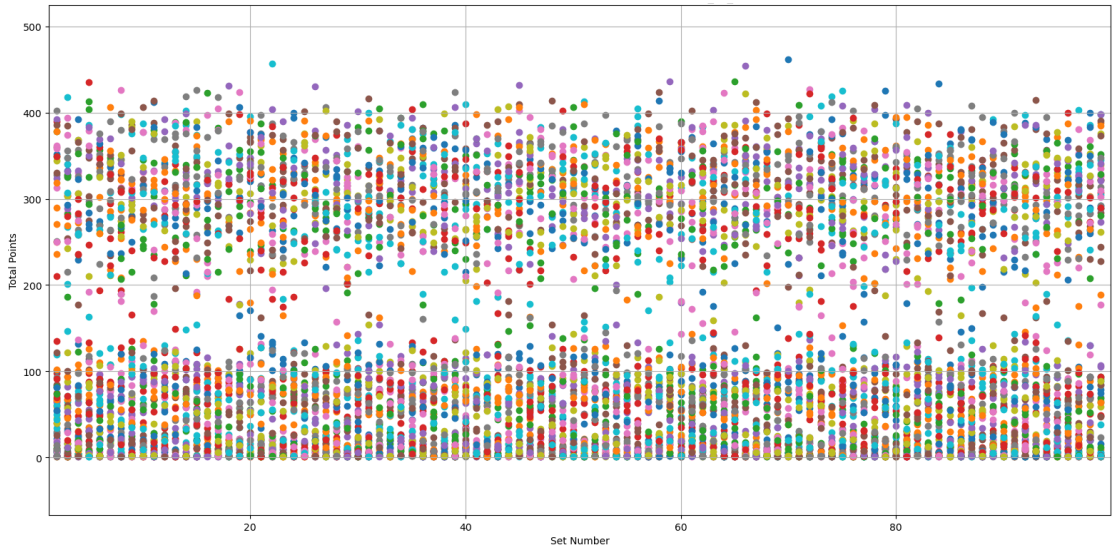

Figure 96: *SOD* for Unconstrained with fixed size, continous = False, one vs all, for a num sets = 100 and num elements = 2 for 100 iterations, zoomed in further.

As we can see, both in *WOD* and *SOD* there appears to be different levels, which we can see much more clear when we do inspect the all vs all options as in the following case.

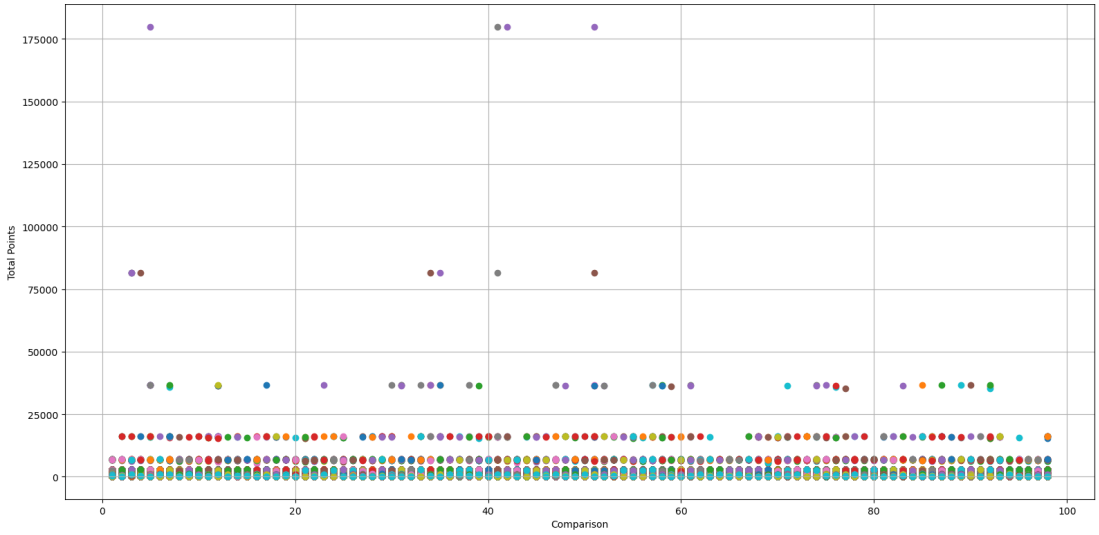

Figure 97: *SOD* for Unconstrained with fixed size, continous = False, all vs all, for a num sets = 100 and num elements = 2 for 100 iterations.

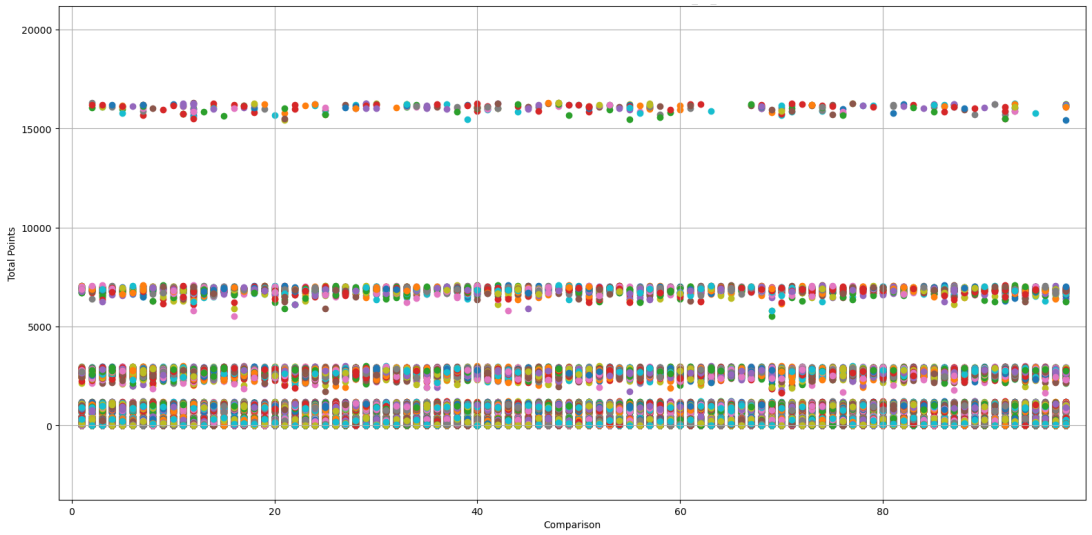

Figure 98: *SOD* for Unconstrained with fixed size, continous = False, all vs all, for a num sets = 100 and num elements = 2 for 100 iterations, zoomed in.

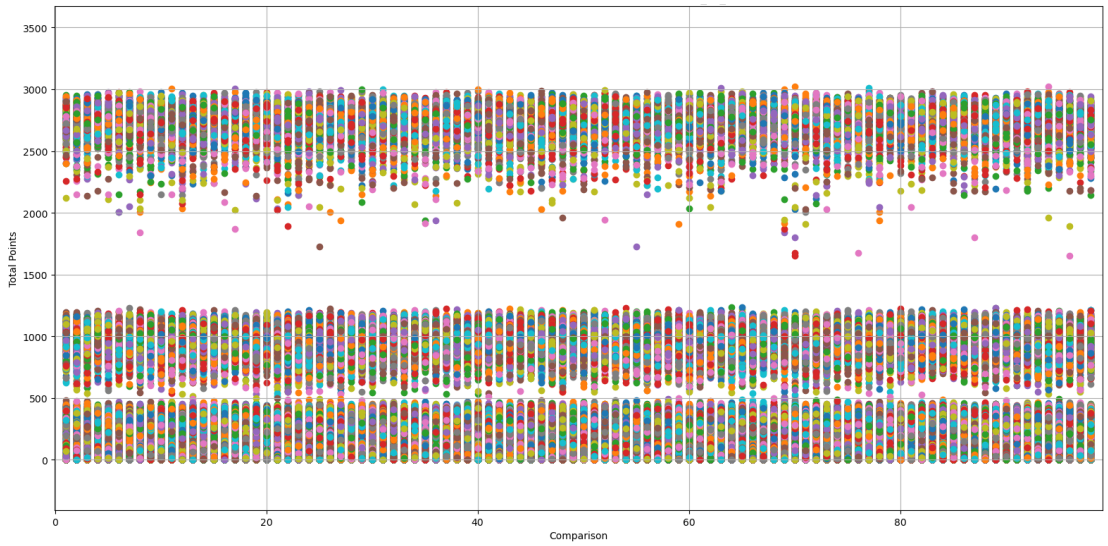

Figure 99: *SOD* for Unconstrained with fixed size, continous = False, all vs all, for a num sets = 100 and num elements = 2 for 100 iterations, zoomed in further.

We will now summarize all the results in the next figure, and we will do it as well in the next subsections for each constraint and parameter, since we cannot inspect carefully every single option for it would be too lengthy. The reader is invited to investigate it herself using the code provided in the repository.

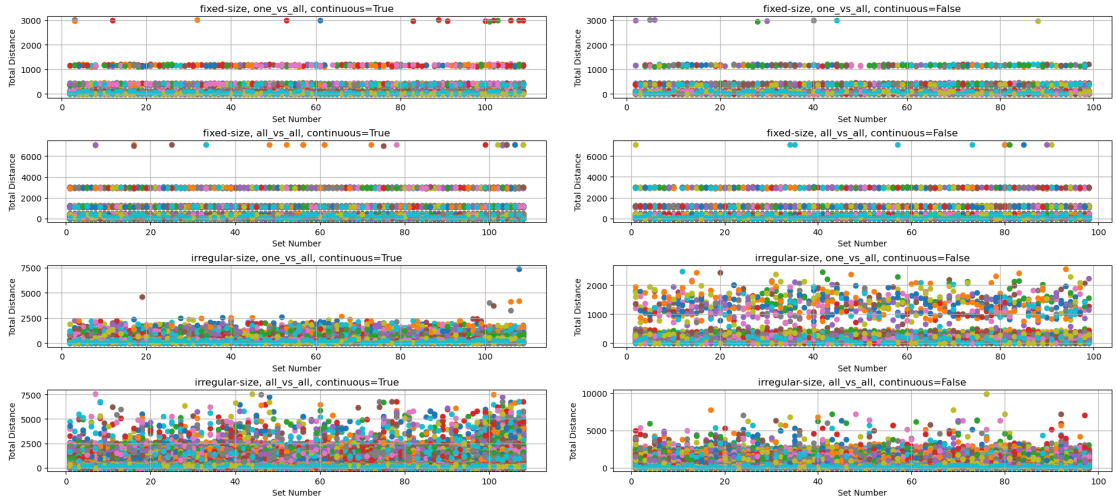

Figure 100: *WOD* of Unconstrained for a num sets = 100 and num elements = 2 for 100 iterations.

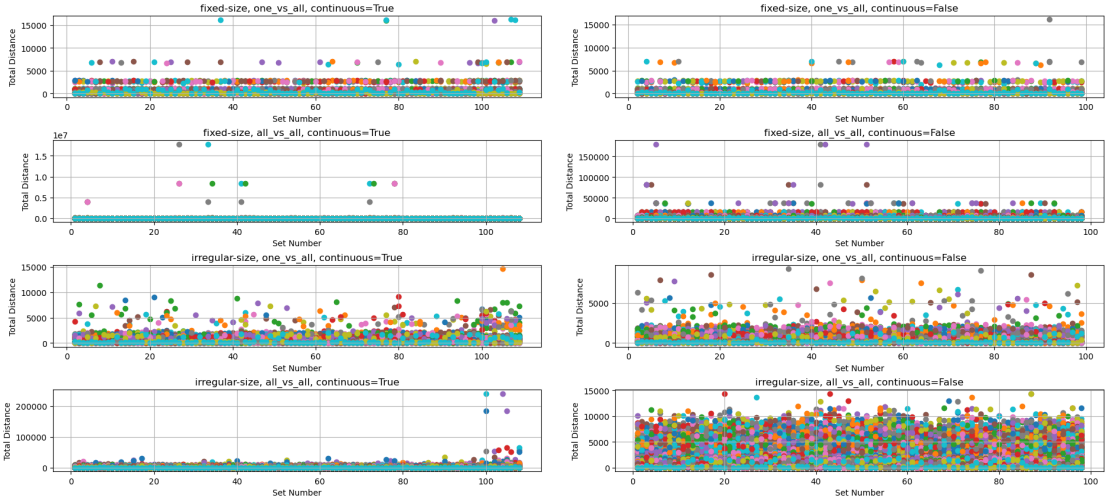

Figure 101: *SOD* of Unconstrained for a num sets = 100 and num elements = 2 for 100 iterations.

6.2. Results for Regular Uniqueness

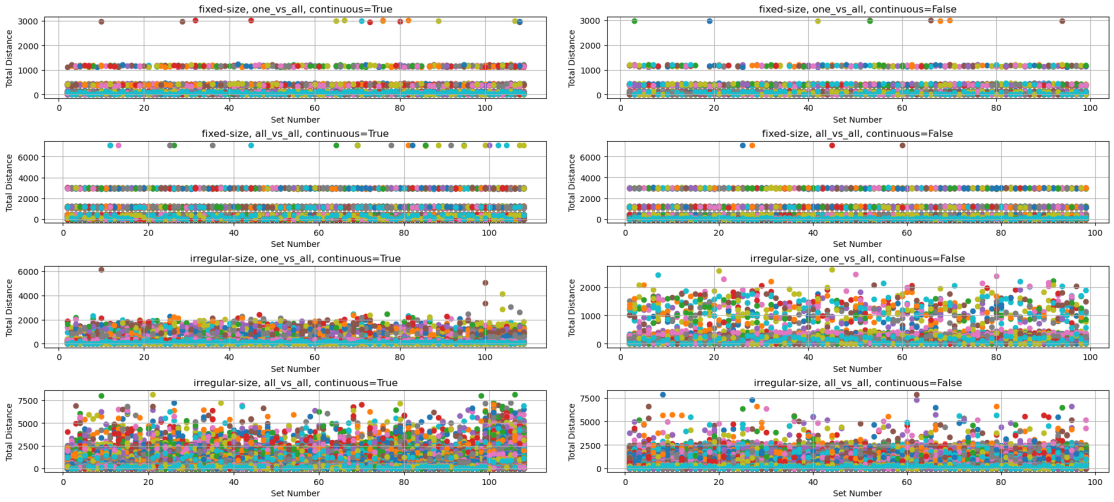

Figure 102: *WOD* of Regular Uniqueness for a num sets = 100 and num elements = 2 for 100 iterations.

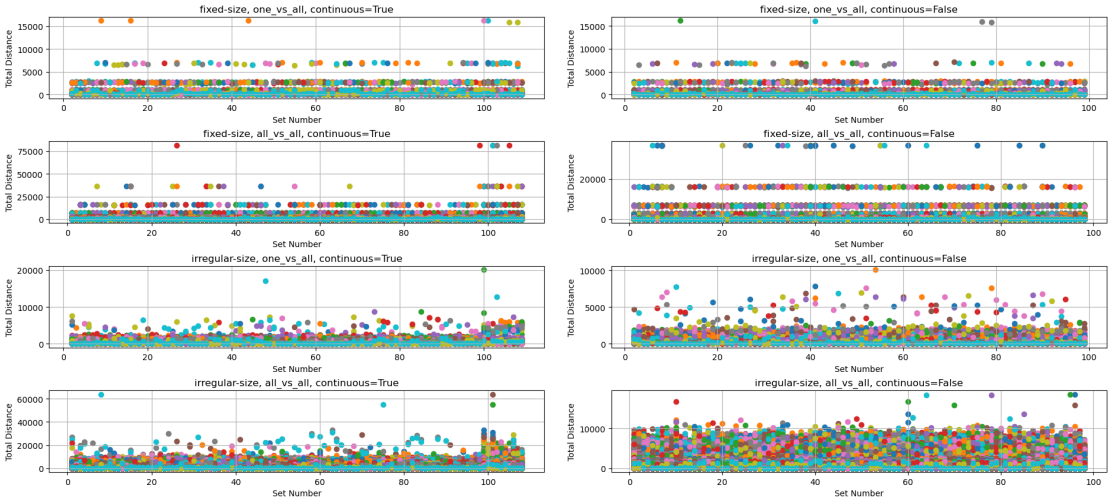

Figure 103: *SOD* of Regular Uniqueness for a num sets = 100 and num elements = 2 for 100 iterations.

6.3. Results for Strict Uniqueness

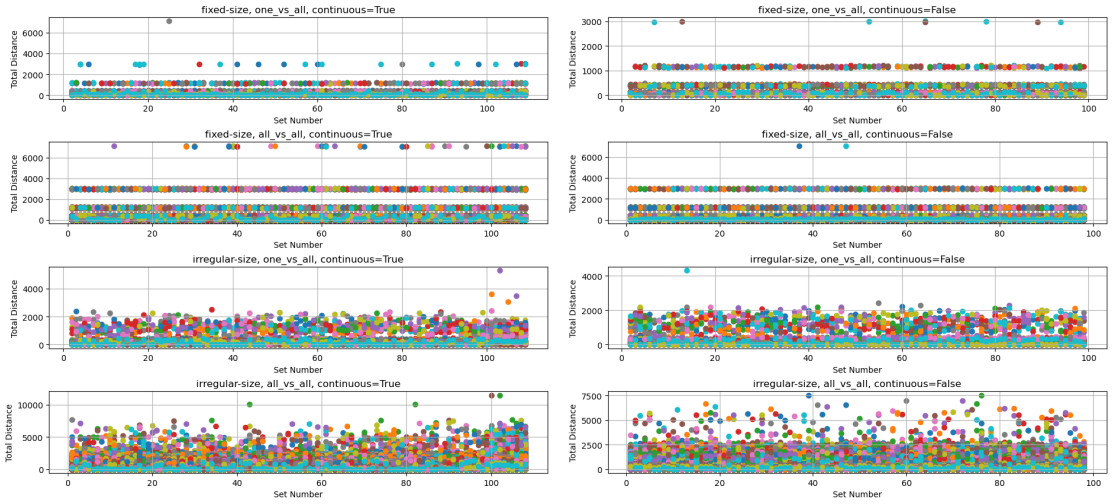

Figure 104: *WOD* of Strict Uniqueness for a num sets = 100 and num elements = 2 for 100 iterations.

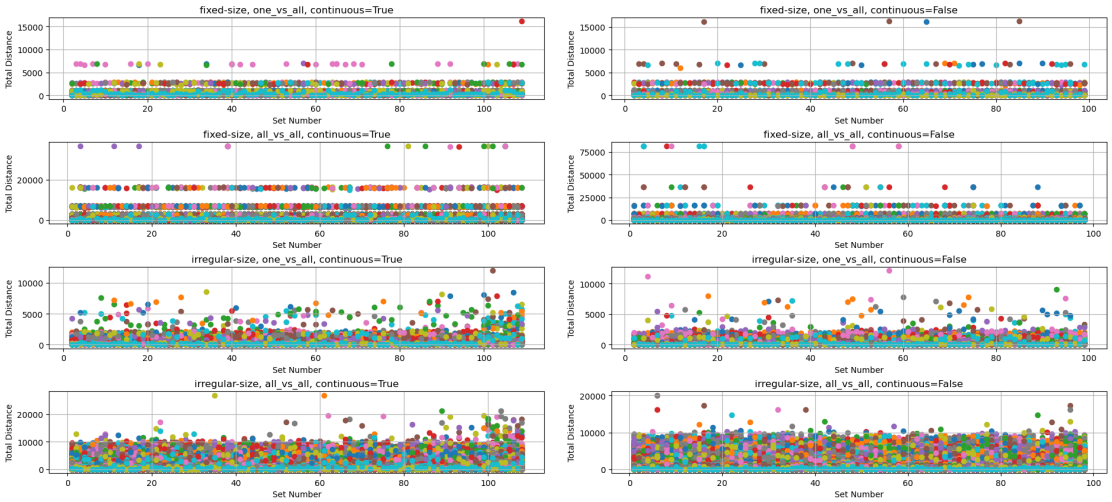

Figure 105: *SOD* of Strict Uniqueness for a num sets = 100 and num elements = 2 for 100 iterations.

6.4. Results for Weighted

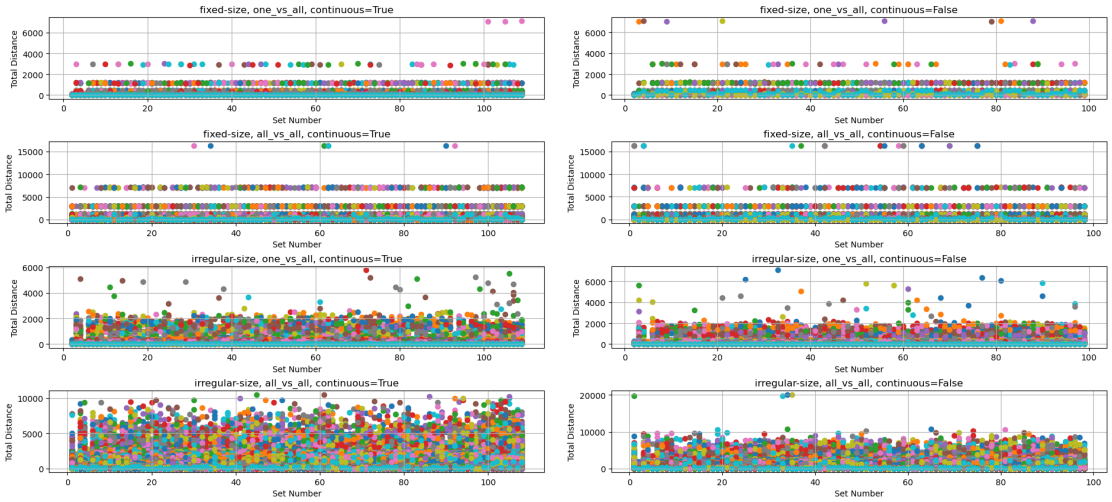

Figure 106: *WOD* of Weighted for a num sets = 100 and num elements = 2 for 100 iterations.

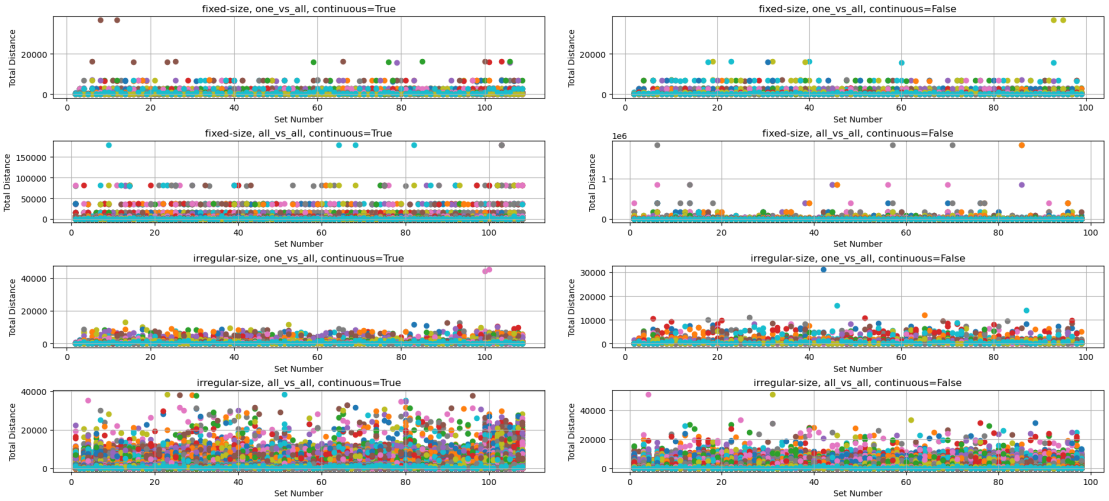

Figure 107: *SOD* of Weighted for a num sets = 100 and num elements = 2 for 100 iterations.

Part V. Applications

In this part we will apply both a **U** manifold and Ontological Differentiation to two different models. One for language and one for theoretical physics. In the first one we will obtain graphical results as we did in the functions sections and we will analyze them, and in the second one we will use the results from the functions sections along with both the notion of **U** and *OD* to be able to provide a new physical model interpretation.

7. Language Model

As anticipated, in this section we will create a **U** manifold for the English language and we will analyze it using *OD*.

7.1. Construction of U

In order to create our **U** manifold we have taken a list of the 5000 thousand most common words in English from "A Frequency Dictionary of Contemporary American English: Word Sketches, Collocates and Thematic Lists" by Mark Davies and Dee Gardner. We then have created a code to give definitions to these words. This code's main task is to generate concise and valid definitions for a list of words by following specific conditions:

- Relevance and Consistency: Each word's definition must consist of exactly three related words, which are either directly connected in meaning or are synonyms. Importantly, these related words must also be part of the original list of words being inspected.

- No Self-Reference: The word being defined should not appear in its own definition. The code actively avoids including the original word in the list of related terms.

- Part of Speech Matching: The words included in the definition should ideally match the part of speech (e.g., noun, verb) of the original word, ensuring the definition is contextually appropriate.

- Completeness: If fewer than three relevant words are initially found, the code searches for additional synonyms or related terms to fill the definition. If it's still not possible to create a valid definition that meets all the criteria, that word is removed from the final output.

- Final Output: The code then cleans up any definitions that don't meet these criteria and saves the valid ones to a text file.

The final file in our repository which contains all these elements which met the aforementioned criteria is called "final definitions" and has 4354 elements.

7.2. One vs All Results

We have chosen one element of our set **U**, the element "Word", and we have ran a comparison for its WOD and SOD with the rest of the elements in the set.

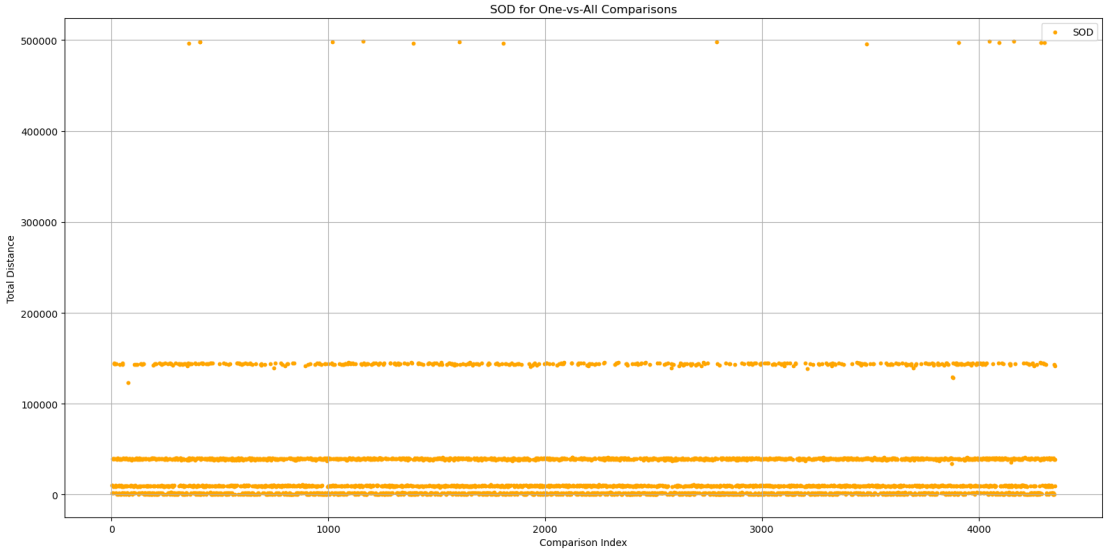

Figure 108: WOD for "Word" against rest of the set.

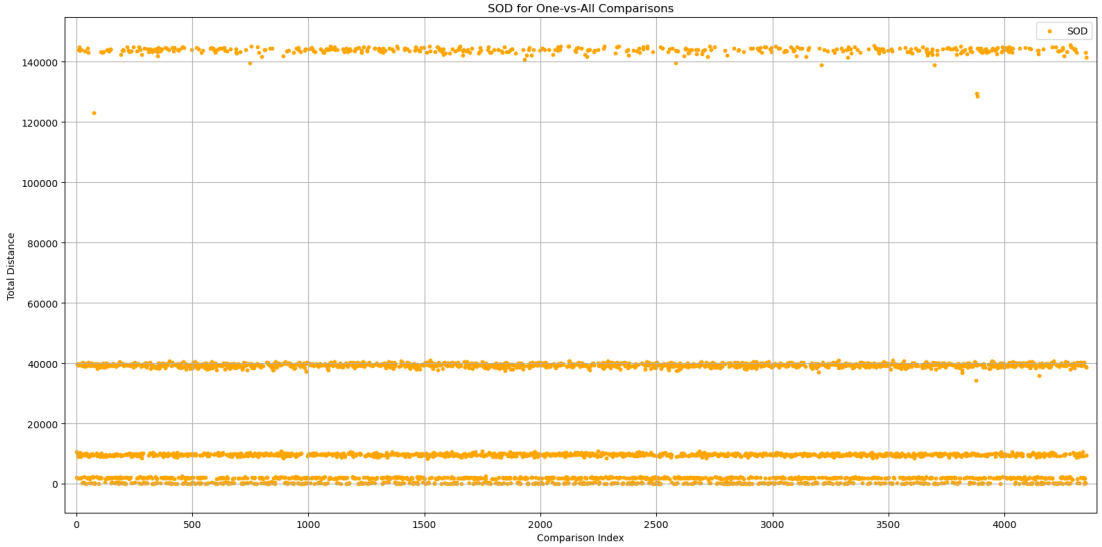

Figure 109: WOD for "Word" against rest of the set, zoomed in the lower region.

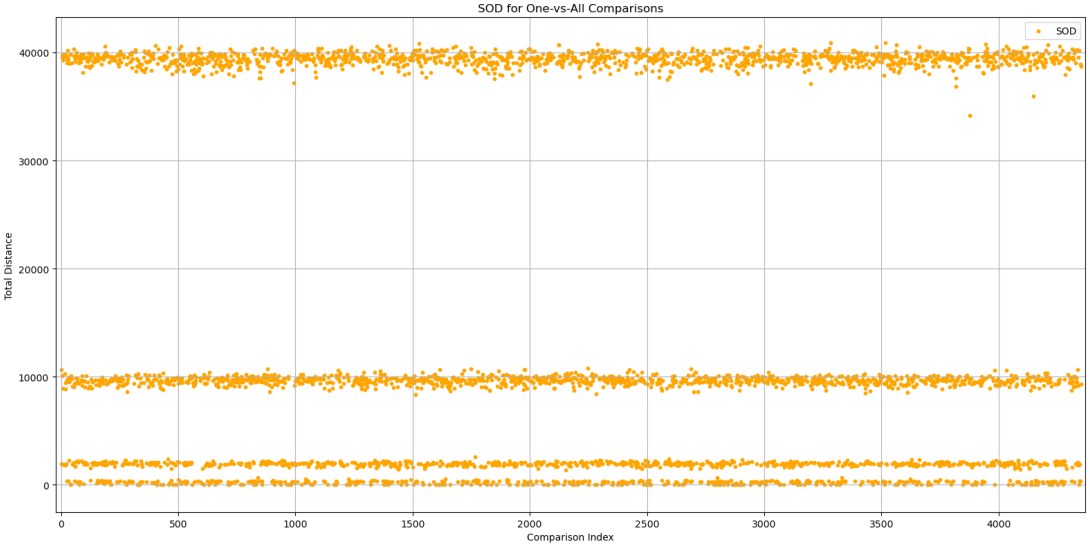

Figure 110: *WOD* for "Word" against rest of the set, zoomed in the still lower region.

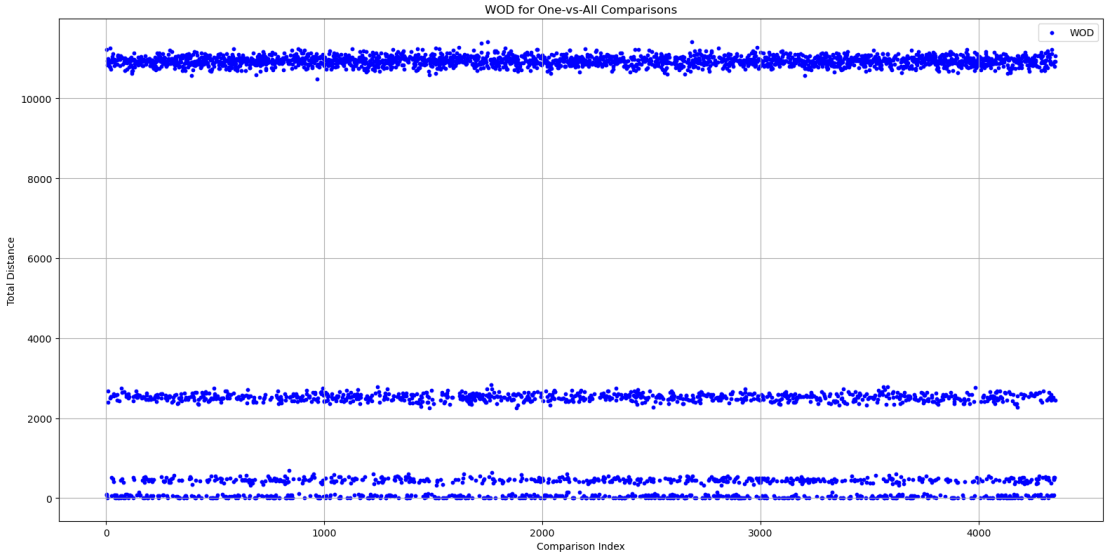

Figure 111: *SOD* for "Word" against rest of the set.

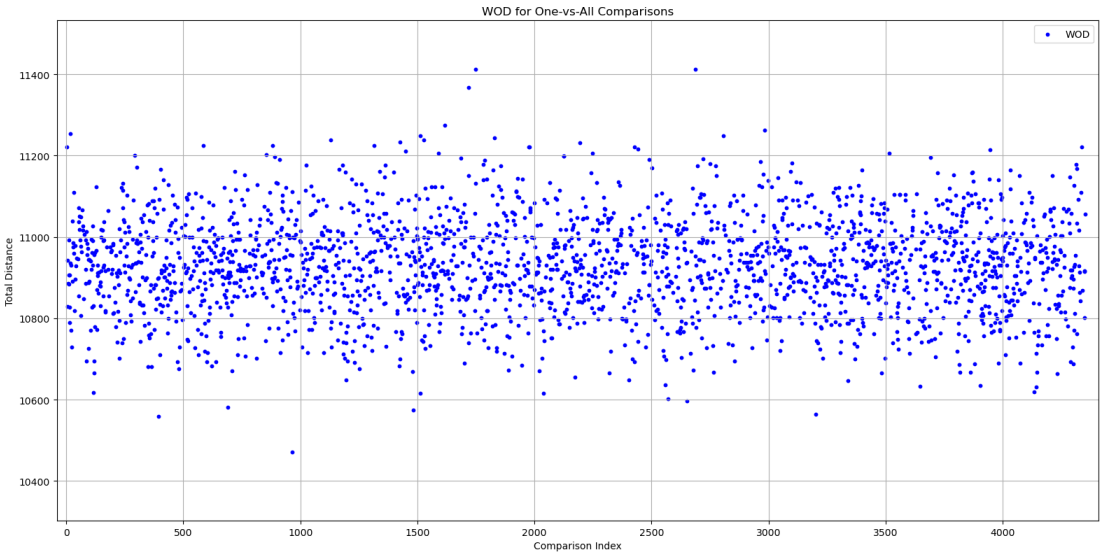

Figure 112: *SOD* for "Word" against rest of the set, zoomed in the upper cluster.

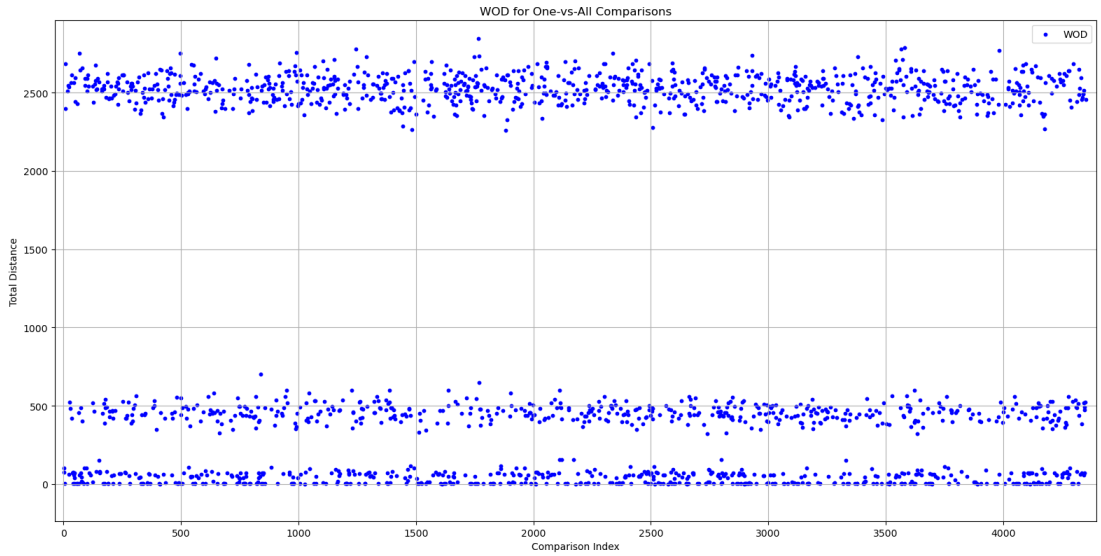

Figure 113: *SOD* for "Word" against rest of the set, zoomed in the lower region.

7.3. All vs All Results

Here we show the results for the all vs all results, which are overlapped (as regularly done for all vs all comparisons).

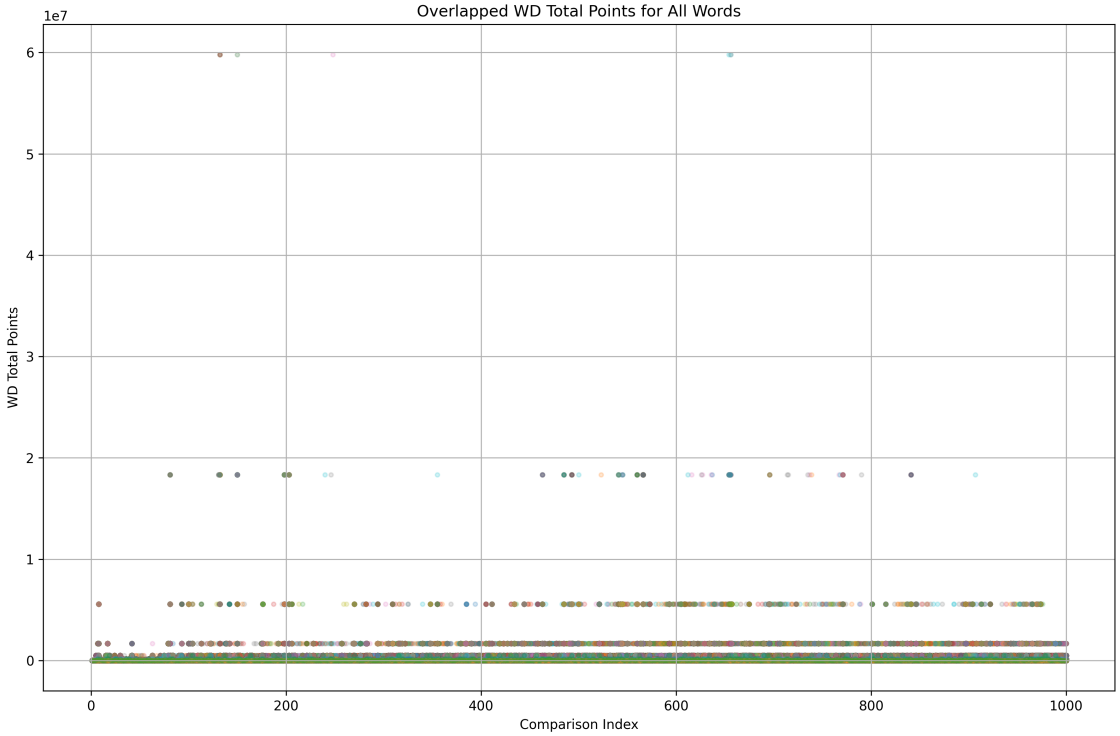

Figure 114: WOD of All vs All

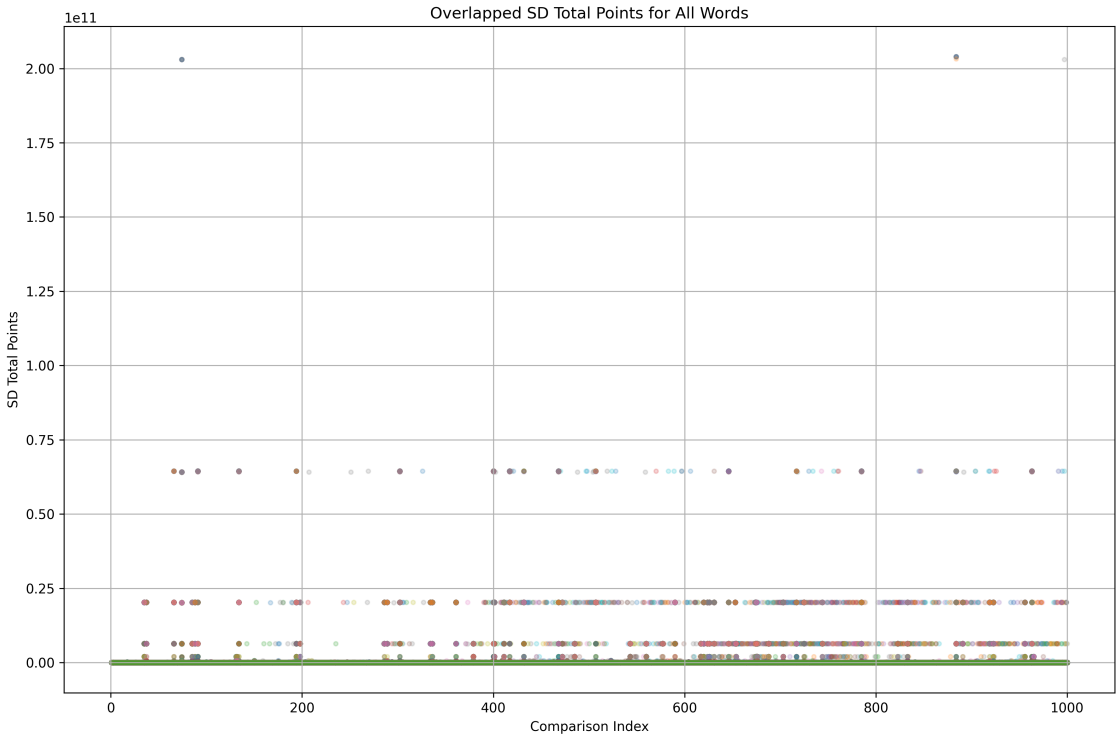

Figure 115: *SOD* of All vs All

As we can see, both from the One vs All and the All vs All results, the stratified structure observed both in random distribution sets as in the compiled plots for finite functions is present. It would be interesting to investigate which elements are closer to each other, to "Word" as in our One vs All case, as well as in the overall picture, using Islands and Sub-Islands. However, this task goes beyond the scope of this work and the computational power to investigate such a plot for All vs All is too lengthy, even though our language model is not tremendously big. The reader is invited to explore it herself in the provided online repository.

8. Physical Model

In this section we will go over the results obtained from the functions section and we will use the notions of **U** and OD to provide an approach to some of the most basic physical models.

8.1. QFT Manifold

In section 6 where **U** had a random distribution, the results showed how OD operated on this type of order produced a landscape of discrete levels (also in the compiled results of finite functions). It seems only natural to propose that a manifold **U** with a random distribution could be a model for a Quantum Field Theory

(QFT) universe.

In QFT, a point is given as x^μ where $\mu = 0, 1, 2, 3$ are the indices representing the 4-dimensional spacetime. Then we have a field $\phi(x)$ (or other field like the spinor or gauge field) which it gives some value to that point. That is, both x^μ and field $\phi(x)$ are separated. What we propose in a **U** manifold is that both point value and field value are one and the same. Points give a sense to fields, and fields give a sense to points, and the reason behind is because they are part of the same conglomerate.

A point P_i in **U** is defined by its relations with other points P in **U**. The way to tell apart different points in such a manifold is through OD, it tells you how "different" the are, how distant these points are from each other. Any function, any physical theory is made to tell points and their causes apart, to explain why the difference in points cause whatever different results.

The field properties (like energy) emerge when we start differentiating points from each other, not from applying some exotic function. The exotic function was already applied in its elements distribution, the exotic function explains the order of the elements in the manifold. The energy levels, the gaps, appear when we do a differentiation of points.

U is the spacetime, P's are just the manifold points and the point distribution (given by some structure S) creates the field. The difference between points, what makes one point be itself and not other, makes the field properties emerge. So the field properties emerge not from applying a function to points, but by telling points apart.

We have then that the reason for discreteness in Quantum Mechanics (QM) comes from how the points of the manifold are distributed, from how they have their elements ordered.

There is also one more aspect to take into account for considering **U** for a QFT manifold. The absolute non-Euclidean property that **U** can have, as shown in the violation of the triangle inequality for the Sample Set in 3.1.2 (let us contemplate though that **U** can be ordered with a function such that it fulfills the triangle inequality, it is not an intrinsic property of **U**, its possibility is) means that **U** can behave in a probability-like form. The calculation of some $OD(P_i, P_j)$ may not be known, determined from former calculations of other P in the same **U**, but it may only be known once the calculation is done. This means that the nondeterministic nature of QM, the probability interpretation, comes from the configuration given by the structure which is applied onto **U**.

As one may have noticed, when a manifold **U** has a finite size, as it was the case for the finite functions and random distribution, the OD presents gaps and levels, which it is not so much the case when **U** is of continuous nature as for recursive functions. The fact that **U**'s finiteness determines if there is quantification should be duly noted when contemplating **U** for a universal model.

8.2. Gravitation

In our description of the **U** it becomes clear that there is no place, no outer place, for particles in **U**, but particles are part of **U**, as we suggest in the previous sub-section. There is nothing defined outside the spacetime, everything is spacetime.

One of the obvious questions then is how gravity, a force so obviously related to particles, can be understood in the context of a **U** manifold. The answer relies on OD once again. Some P_i will have shorter values of OD with other points (shorter distances) if such P_i is or has elements which are densely repeated across **U**, or in the case that the elements size is irregular, and P_i has a very big size, then it will be easier for it to have closer distances to other points. If we were to understand the cardinality, the size, of the read function (e.g. at level 1) as the mass, the content of the point, then we could see it as an analogy of gravitational attraction.

In an irregular element size set up, the more mass P_i has, the shorter it will be to the other points. In a weighted set up, the denser the frequencies of the elements of P_i, the shorter the distance to the other points. That analogy of mass, of density of P_i is the gravitational attraction. It's a point mass which attracts the others closer to it.

Part VI.
Online Resources

We have created a repository on GitHub (https://github.com/pablogc1/Ontological-Differentiation) where one can find the different codes used to calculate and plot the different OD's with their different setups, as well as the application on the language model.