Scientific Research Publishing

American Journal of Operations Research, 2022, 12, 179-193
https://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

A Heuristic Search Approach to Multidimensional Scaling

Patrick R. McMullen

School of Business, Wake Forest University, Winston-Salem, USA
Email: mcmullpr@wfu.edu

How to cite this paper: McMullen, P.R. (2022) A Heuristic Search Approach to Multidimensional Scaling. *American Journal of Operations Research*, **12**, 179-193.
https://doi.org/10.4236/ajor.2022.125010

Received: August 17, 2022
Accepted: September 12, 2022
Published: September 15, 2022

Abstract

This research effort presents an approach to accomplish Multidimensional Scaling (MDS) via the heuristic approach of Simulated Annealing. Multidimensional scaling is an approach used to convert matrix-based similarity (or dissimilarity data) into spatial form, usually via two or three dimensions. Performing MDS has several important applications—Geographic Information Systems, DNA Sequencing, and Marketing Research are just a few. Traditionally, classical MDS decomposes the similarity or dissimilarity matrix into its eigensystem and uses the eigensystem to calculate spatial coordinates. Here, a heuristic search-based approach is used to find coordinates from a dissimilarity matrix that minimizes a cost function. The proposed methodology is used over a variety of problems. Experimentation shows that the presented methodology consistently outperforms solutions obtained via the classical MDS approach, and this approach can be used for other important applications.

Keywords

Optimization, Search, Heuristic

1. Introduction

It is often important to visualize data when the data is presented in a primitive form. The visual perspective enables more of an intuitive understanding of the data and interrelationships between data points. Consider several viral DNA sequences to be compared—it would be helpful to see the degree of similarity (or difference) between them all. Of course, some statistics might be available to assist, but placing them into some spatial arrangement (to enable visualization) would be most helpful. To do this, however, some effort is required.

Consider another example where we are given a matrix of mileages between cities. This data might make more sense if it were converted into the form of a two-dimensional map, so that we could visualize the distances between all cities. Again, this would also take some work.

These example scenarios are good candidates for the use of Multidimensional Scaling. Multidimensional Scaling (MDS hereafter) is a technique used to convert similarity (or dissimilarity) data into spatial form, so that the data set can be visualized [1]. Visualizing data simplifies understanding. The mathematics required to perform multidimensional scaling are somewhat intense, as Eigen decomposition of the given matrix is required. Additionally, results in terms of accuracy can be sub-optimal. As such, other alternatives are worth exploring.

One such general alternative is a search heuristic. A search heuristic usually starts with a feasible, randomly-generated solution with the intent of eventually finding an optimal (or near-optimal) solution to an optimization problem. The optimality of a solution is determined via an objective function measure. During the search process, feasible solutions are modified to a minor degree with hopes of improvement ultimately approaching global optimality. While this search process is ongoing, newly obtained solutions replace incumbent solutions. Sometimes, these incumbent solutions are replaced by relatively inferior newly obtained solutions. This might seem odd, but research has shown, that when properly implemented, search heuristics that occasionally replace incumbent solutions with relatively inferior solutions can help avoid being trapped at local optima [2]. Tabu Search [3] [4], Simulated Annealing [5] and Genetic Algorithms [6] are three types of search heuristics that fit the description above. When properly constructed and implemented, search heuristics can provide near-optimal results with less computational effort than other solution approaches.

For this research effort, a Simulated Annealing approach is used to address an MDS problem with a distance matrix as input that outputs a set of Cartesian Coordinates. Specifically, the following sections: describe classical multidimensional scaling; describe a simulated annealing based alternative to the MDS problem, outline an experiment for implementation of the proposed methodology; detail experimental results; and draw conclusions.

2. Classical Approach to Multidimensional Scaling

Classical Multidimensional Scaling is a technique used to convert similarity (or dissimilarity) data into m-dimensions. This conversion is done so that a spatial representation of the data can be pursued. The given data is typically provided in n x n matrix form. Each element in the matrix is some pairwise measure of the similarity or dissimilarity between data points. The conversion of the data from matrix to spatial form is done for a variety of reasons: market segmentation, graphing, matching, sequencing, etc. Any attempt to visualize observations when given similarity data can be thought of as an application of MDS.

In classical MDS, an $n \times n$ dissimilarity matrix is provided, which is referred to as D. This matrix contains elements d_{ij}, which shows the Euclidean distance between points *i* and *j*. For further analysis, the matrix $\boldsymbol{D}^2$ is constructed, which is the square of the values of d_{ij}. Centering matrix $\boldsymbol{C}$ is then used, which is $\boldsymbol{I} - ((1/n)(J_n))$, where $\boldsymbol{I}$ is the identity matrix (values of "1" on the main diagonal, "0" elsewhere), and J_n is a matrix of "1". These elements form the double-centering matrix $\boldsymbol{B}$. This double-centered matrix is constructed so that the means of all row and column vectors are zero [7]. This matrix is as follows:

$$\boldsymbol{B} = -(1/2)\boldsymbol{C}\boldsymbol{D}^2\boldsymbol{C} \tag{1}$$

The $\boldsymbol{B}$ matrix is then decomposed into its eigensystem. Specifically, the *m*-largest eigenvalues and eigenvectors are captured. A spatial representation of the data is provided by the *n* by *m* matrix $\boldsymbol{X}$, which is determined as follows:

$$\boldsymbol{X} = \boldsymbol{E}\boldsymbol{\Lambda}^{1/2}, \tag{2}$$

where $\boldsymbol{E}$ is an $n \times m$ matrix of the *m*-largest eigenvectors, and $\boldsymbol{\Lambda}^{1/2}$ is the corresponding $m \times m$ matrix of the square-root of the *m*-largest eigenvalues on the main diagonal, with values of "0" elsewhere [8] [9].

This eigenpair approach exploits the dissimilarity matrix $\boldsymbol{D}$ to obtain spatial coordinates ($\boldsymbol{X}$).

$$Z = \sqrt{\sum_{i=2}^{n}\sum_{j=1}^{i-1}\left(d_{ij} - \sqrt{\left(x_{i1} - x_{j1}\right)^2 + \left(x_{i2} - x_{j2}\right)^2}\right)^2} \tag{3}$$

In Equation (3) above, the x_{ij} values are simply values of the $\boldsymbol{X}$ matrix. The above equation assumes that a 2-dimensional spatial solution is desired. The value Z can be thought of as an objective function value. While this equation might appear intimidating, it is not—the objective is simply to find a solution (values of x_{ij}) resulting in an aggregate minimized distance equal to the given distance matrix. When the x_{ij} values result in distances that equal the d_{ij} distances, the objective function value has been minimized to a value of "0".

Classical MDS decomposes an $n \times n$ matrix and outputs an $n \times 2$ (or $n \times 3$). This means that $n - 2$ eigenpairs are lost for a two-dimensional spatial solution, or $n - 3$ eigenpairs are lost for a three-dimensional spatial solution. These "lost" eigenpairs limit our ability to explain the variation in the distance data, as each eigenpair assists in explaining the variation in the distance data. This, of course, is unfortunate. Additionally, decomposing a matrix into its eigenpairs is computationally expensive. As such, a viable alternative is worth exploration.

3. Heuristic Approach to Multidimensional Scaling

Regardless of the approach used to perform MDS, the objective is the same—minimization of the objective function shown in Equation (3). In order to present the methodology used to solve this problem, the following parameters are defined in Table 1.

As stated before, Simulated Annealing is the chosen search heuristic to address the MDS problem at hand. Simulated Annealing gets its name from "annealing," which is the heating of a solid (usually a metal) to a high temperature and then the slow, subsequent cooling of the solid. During the cooling process,

Table 1. Search parameters.

Sym	Description	Sym	Description
$Iter$	number of iterations	x_{ab}	bth coord. value for point a (current sol.)
Sim	number of simulations	$(x_{ab})_t$	bth coord. value for point a (test sol.)
T	current temperature	$(x_{ab})_b$	bth coord. value for point a (best sol.)
T_I	initial temperature	d_{ij}	Euclidean distance from points i to j
T_F	final temperature	c_i	Cost vector of test solution ($n \times 1$)
CR	cooling rate	q	target data point for modification
n	number of data points	Z	objective function value of current solution
m	number of dimensions	Z_t	objective function value of test solution
k_B	Boltzman constant	Z_b	objective function value of best solution

the particles arrange themselves from a chaotic state into an ordered state. This cooling process is intended to enhance some physical property of the solid. With "simulated" annealing, the objective function is analogous to the desired physical property of the solid—we start with a randomly-generated solution (analogous to a solid at a high temperature) and modify the solution until a desired condition is obtained. Early in the search, there is more liberality in replacing the incumbent solution as compared to later in the search—this is analogous to the "chaotic" state early in the annealing process compared to the more "ordered" state later in the annealing process.

The following subsections detail the Simulated Annealing heuristic search process used to find a solution to the problem at hand.

Step 1: Initialization

An initial solution is constructed by assigning random values to the x_{ab} values. Specifically, this is done as follows:

$$x_{ab} = -250 + \mathrm{U}(0,500), \forall a,b \tag{4}$$

where U(0, 500) is a uniformly-distributed random variable on the (0, 500) interval. It should be noted that our solution boundary is confined to the (0, 500) interval for both the horizontal and vertical axes. The "test" solution and "best" solution values are set equal to this initial solution. Specifically, this is as follows:

$$(x_{ab})_t = x_{ab}, \forall a,b \tag{5}$$

$$(x_{ab})_b = x_{ab}, \forall a,b \tag{6}$$

The initial solution is also used to determine the objective function value, Z. This is done as follows:

$$Z = \sqrt{\sum_{i=2}^{n}\sum_{j=1}^{i-1}\left(d_{ij} - \sqrt{\left(x_{i1}-x_{j1}\right)^2 + \left(x_{i2}-x_{j2}\right)^2}\right)^2} \tag{7}$$

The objective function values for the "test" and "best" solutions are assigned this same value as follows:

$$Z_t = Z \tag{8}$$

$$Z_b = Z \tag{9}$$

Step 2: Modification

In order to improve the current solution such that it results in an optimal (or at least "near optimal") condition, it must be "modified". Specifically, the "test" solution must be modified as the first step toward improving the current solution. This modification is done to one of the *n* data points. The modification is a "minor" modification—large modifications are avoided such that a controlled improvement process is followed.

First, the cost vector is obtained via the following:

$$c_i = \sqrt{\sum_{j=1}^{n}\left(d_{ij} - \sqrt{\left(x_{i1_t} - x_{j1_t}\right)^2 + \left(x_{i2_t} - x_{j2_t}\right)^2}\right)^2}, \forall i \tag{10}$$

The cost vector shows the amount of "error" each of the *n* locations contributes to the objective function value. Higher cost values contribute more to sub-optimality. Each potential target data point (*q*) for the modification has the following probability of being selected:

$$P(q) = c_i \Big/ \sum_{i=1}^{n} c_i \tag{11}$$

Monte-Carlo simulation is used to select the target value *q*. Cities with higher cost vectors are more likely to be selected as targets for modification, the actual modification of the selected data point (*q*) is as follows:

$$\left(x_{q1}\right)_t = \left(x_{q1}\right)_t + T * 25 * \left(1 - \mathrm{U}(0,1)\right) \tag{12}$$

$$\left(x_{q2}\right)_t = \left(x_{q2}\right)_t + T * 25 * \left(1 - \mathrm{U}(0,1)\right) \tag{13}$$

As stated, this modification is done for all *m* dimensions of the targeted data point (*m* = 2 is used here). The current temperature value (*T*) is used in this calculation so that the aggressiveness of each modification decreases proportionally to the value of *T*.

Step 3: Objective Function

The modified test solution is then used to determine the objective function value (Z_t) as shown below:

$$Z_t = \sqrt{\sum_{i=2}^{n}\sum_{j=1}^{i-1}\left(d_{ij} - \sqrt{\left(x_{i1} - x_{j1}\right)^2 + \left(x_{i2} - x_{j2}\right)^2}\right)^2} \tag{14}$$

Step 4: Solution Comparison

Test 1:

If the objective function value associated with the test solution is less than the objective function associated with the current solution, the test solution replaces the current solution. Specifically, the following applies:

$$x_{ij} = \left(x_{ij}\right)_t \tag{15}$$

$$Z = Z_t \tag{16}$$

Otherwise, the difference between the current solution and test solution is determined:

$$\delta E = (Z_t - Z)/Z \tag{17}$$

This difference is shown in relative form, with the value being represented by δE. This value is always positive, since $Z_t > Z$. This value is used to determine whether-or-not the inferior test solution should replace the current solution, via a probabilistic condition. Specifically, the value P_A represents the probability of replacing the current solution with the test solution. This probability is as follows:

$$P_A = \exp(\delta E/k_B T) \tag{18}$$

The value k_B is referred to as the Boltzman Constant [10], and is user-determined such that a certain degree of relative inferiority (δE) results in the test solution having a specific probability of replacing the current solution.

A uniformly-distributed random number on the U(0, 1) interval is then generated. If this random number is less than P_A, the test solution replaces the current solution, in accordance with Equations (12) and (13).

Test 2:

If the objective function value associated with the test solution is less than the objective function associated with the best solution, the test solution replaces the best solution.

$$(x_{ij})_b = (x_{ij})_t \tag{19}$$

$$Z_b = Z_t \tag{20}$$

Otherwise, no action is taken.

Step 5: Incrementation

Steps 2, 3 and 4 are repeated *Iter* times. The value of T is then updated as follows:

$$T = T * CR \tag{21}$$

This continues while $T > 1$. When $T \le 1$, the simulation has concluded. When the simulation has concluded, the current solution is replaced by the best solution, and the value of T is set to T_1. Mathematically, this is done as follows:

$$x_{ij} = (x_{ij})_b \tag{22}$$

$$Z = Z_b \tag{23}$$

$$T = T_1 \tag{24}$$

This is repeated *Sim* times, and the best solution found is reported.

To gain a better understanding of the presented methodology, it is presented in pseudocode via **Figure 1**.

As a point of clarification, it should be noted that repeated simulations in an effort to obtain an optimal solution may present some confusing terminology. As such, **Figure 2** below shows the hierarchy of terms used here.

```
Initialize();
for(g=1;g<=Sim;g++){
while(T> 1){
    for(h=1;h<=Iter;h++){
Modification();
ObjectiveFunction();
If($Z_t$<$Z$)CurrentSolution←TestSolution;
      Else {
$\delta E$ = ($Z_b$ - $Z$)/$Z$;
$P_A$ = exp($\delta E/k_B T$);
If(U(0, 1)<$P_A$) CurrentSolution←TestSolution;
}//end else
If($Z_b$<$Z_t$)BestSolution←TestSolution;
}//end h
}//end while
CurrentSolution←BestSolution;
$T$ = $T_1$;
}//end g
ReportBestSolution();
```

Figure 1. Search heuristic in pseudocode.

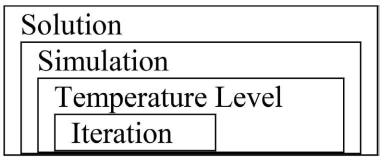

Figure 2. Hierarchy of simulation elements.

A simulation is a subset of a specific solution. Specifically, there are *Sim* simulations performed for each solution. This is one of the defenses against being "trapped" at local optima. At the conclusion of each simulation (*Sim*) during the search process, the "current" solution is replaced by the "best" solution. This provides the next simulation (*Sim*) with the best possible starting point in the attempt to find the optimal solution. If this were not done, the search process is likely trapped at a sub-optimal solution. A single instance of executing the presented methodology is classified as a "solution". A "simulation" in this context is only a subset of the solution process.

4. Experimentation

An experiment is designed to evaluate the effectiveness of the presented methodology. A distance matrix was obtained for 36 metropolitan areas in the continental United States. These are "air-distances". These values were obtained via the website https://www.airmilescalculator.com/. These distances can be thought of as "links" in the parlance of networks. The number of links in a network of n locations is $n(n-1)/2$. These links can also be thought of as pairwise distances.

4.1. Implementation

The methodology presented is coded via the NetLogo software package (https://ccl.northwestern.edu). NetLogo is a Java-based programming environment, particularly suited to enable agent-based simulation [11] [12]. Agent-based simulation is desired here, as the n cities can be thought of as agents.

The distance matrix is used to address problems ranging from four cities (n = 4) to 36 cities (n = 36). All values of n between 4 and 36 are used for assessment. As such, 33 (1 + 36 − 4) unique problems are addressed. Each problem is solved five times so that reliable estimates of performance are available. An example solution is detailed in **Figure 3**.

Figure 3 shows a solution for a problem with n = 26 cities. The "best" solution found for this example has an objective function value of $Z_b = 24.13$. This value quantifies the square root of the sum of squared differences between the actual (given) distance matrix and the distance matrix associated with the "best" solution $((x_{ab})_b)$. In this case, there is a total difference of 24.13 US miles between the two matrices. A value of $Z_b = 0$ is considered optimal—which would imply no difference between the actual distance matrix and the distance matrix associated with $(x_{ab})_b$.

Note from **Figure 3** the solution resembles the map of the continental United States if the image is rotated about 45 degrees in the counter-clockwise direction. The search is not concerned with the proper "orientation" of the solution—it is only concerned with the proper location of the cities with respect to the other cities.

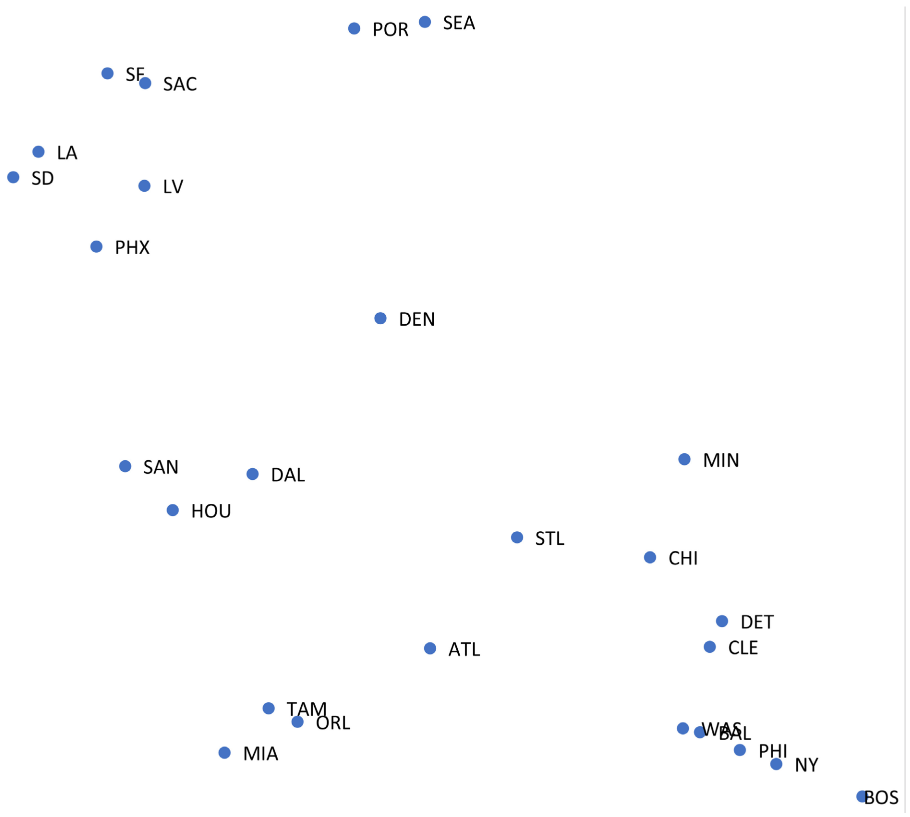

Figure 3. Example problem solution.

4.2. Computational Experience

The program was run on a Windows machine, with an Intel Core 8565U processor, at 4.6 GHz. The user-chosen parameters were chosen via trial and error, so that the "best" solutions could be found repeatedly. This is detailed in Table 2.

For all solutions, a cooling rate (CR) of 0.99 was used, along with 75 iterations per temperature level ($Iter$). The starting temperature (T_1) was 25 and the final temperature (T_F) was 17.5. The k_B value was set such that a solution with 1% relative inferiority would replace the incumbent solution with a 10% probability.

4.3. Stagnation

During the development of the search process, it was noticed that the solution was getting trapped at local optima, despite efforts to prevent this. This local optima trapping phenomenon is referred to as "stagnation" hereafter. Specifically, improvement was impeded by cities being out of place. That is, cities located in places where minor modifications are not aggressive enough to improve the objective function value, resulting in solutions becoming stagnant. There are two types of stagnation requiring treatment—single city stagnation and two-city stagnation.

4.3.1. One-City Stagnation

Figure 4 shows another solution for $n = 26$ cities.

Proper bearings might be realized if the solution is rotated about 150 degrees in the counter-clockwise direction. An unfortunate development then becomes apparent. Miami, Florida (MIA) is "out of place". MIA should be closer to the other Florida locations of Tampa and Orlando (TAM and ORL). Other cities seem relatively properly placed. Unfortunately, the modification process described above will not correct the problem, as this modification process can only accomplish minor modifications—as such, MIA will not be moved anywhere that is helpful. Instead, the MIA location must be moved in a more aggressive fashion. The general procedure for this lies with the cost vector (c). The maximum value in the cost vector displays the most "offensive" city in terms of contribution

Table 2. Simulation settings.

Problem Size (n)	Simulations
4 - 8	12
9 - 15	25
16 - 24	50
25 - 29	150
30 - 32	200
33 - 35	250
36	300

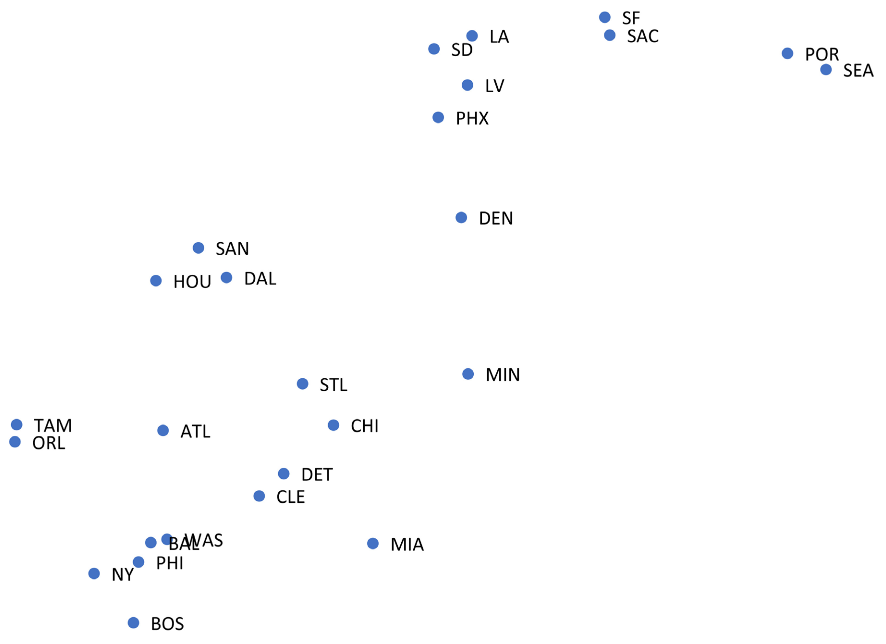

Figure 4. Example of one-city stagnation.

to the objective function value. The city associated with the highest value of the cost vector is then selected as the target. This target city is referred to as "q". The modification to city q is as follows:

$$\left(x_{q1}\right)_t = \left(x_{q1}\right)_t + 500 * \left(1 - \mathrm{U}\left(0,1\right)\right) \tag{25}$$

$$\left(x_{q2}\right)_t = \left(x_{q2}\right)_t + 500 * \left(1 - \mathrm{U}\left(0,1\right)\right) \tag{26}$$

This modification is similar to the one done in normal circumstances detailed above with two key differences. The modification to combat the stagnation is not sensitive to the value of T. Additionally, the modification here is much more aggressive than the modification performed under normal circumstances, so that the "offending" city can be moved further that normal.

4.3.2. Two-City Stagnation

Figure 5 shows yet another solution. If the image is rotated about 45 degrees in the counter-clockwise direction, proper perspective might be realized.

This solution, at first, might appear to be "reasonable". Upon further inspection, however, one might notice a "disconnect" between the eastern cities and western cities. Specifically, the western cities seem to be inverted when compared to the eastern cities. Seattle and Portland (SEA and POR) are shown very much out of place. Unfortunately, this condition cannot be remedied via the single-city stagnation just described. The two "offensive cities" are properly oriented with respect to each other and other nearby cities. This prevents us from moving, for example, SEA to its proper general location because the result would still result in a suboptimal condition—a large distance between SEA and POR would be harmful. Moving both offending cities simultaneously by the same amount is

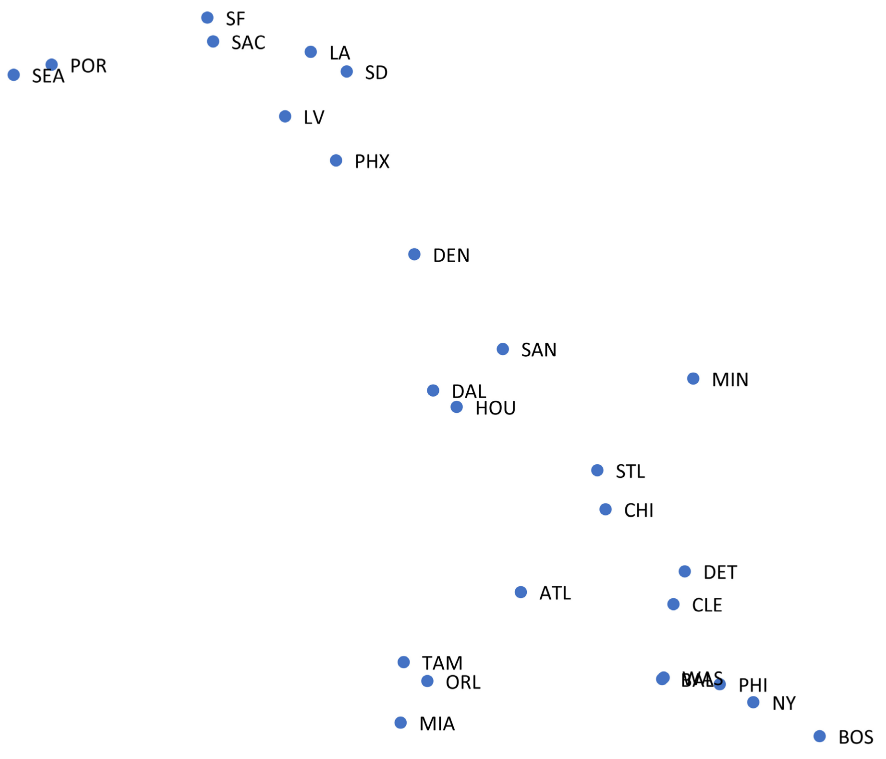

Figure 5. Example of two-city stagnation.

found to alleviate the problem. This two-city movement is done by re-examining the cost vector *c*.

Specifically, the following value Y is calculated:

$$Y = \max\left(c_i + c_j\right) \tag{27}$$

The row and column resulting in the maximum value Y are the two cities contributing most to sub-optimality. Here "*r*" is used as row index "*i*" and "*s*" is used as column index "*j*". These two targets are moved in an aggressive fashion. The following shows the movement process. The dx and dy terms will show horizontal and vertical movements, respectively.

$$dx = 500 * \left(1 - \mathrm{U}\left(0,1\right)\right) \tag{28}$$

$$dy = 500 * \left(1 - \mathrm{U}\left(0,1\right)\right) \tag{29}$$

The following movements are then made for cities r and s.

$$\left(x_{r1}\right)_t = \left(x_{r1}\right)_t + dx \tag{30}$$

$$\left(x_{r2}\right)_t = \left(x_{r2}\right)_t + dy \tag{31}$$

$$\left(x_{s1}\right)_t = \left(x_{s1}\right)_t + dx \tag{32}$$

$$\left(x_{s2}\right)_t = \left(x_{s2}\right)_t + dy \tag{33}$$

Cities r and s are moved the exact same amount and in the same direction. The distance between them will not change. It is desired that their simultaneous movement will at some point induce a condition that ends stagnation and move towards an optimal solution.

It should be noted that the search heuristic performed modification to address both forms of stagnation on a periodic basis during the search.

4.4. Convergence

With stagnation addressed, attention is turned to how well the solutions approach their desired objective function values. It is, of course, desired that the objective function value finds its minimum as efficiently as possible. Figure 6 below shows three types of convergence noticed during the development process. The horizontal axis represents time (the "age" of the search), while the vertical axis shows the "best" objective function value found at the associated time. For example, at the start of the search, the objective function value is high (undesirable) because of randomly-generated initial solution is not motivated in terms of the objective function value. Improvement then commences because the search process is motivated to find minimal values of the objective function value.

The blue line in Figure 6 shows the first type of convergence, which is classified here as "quick convergence". This is where the objective function achieves a near-optimal value without incident. The orange line shows what is classified as "stagnation remedied". This is where during the search process, stagnation occurs, but is remedied via the approaches described above, and a near-optimal condition is eventually obtained. Note the "rapid improvement" for the orange line when stagnation is remedied. The grey line is classified as "stagnation not remedied". This is where stagnation occurs, and is not remedied. Fortunately, this research effort only experienced "stagnation not remedied" during the developmental phase—once this condition was observed during development, the search heuristic was modified such that stagnation was always remedied. The results section further details this.

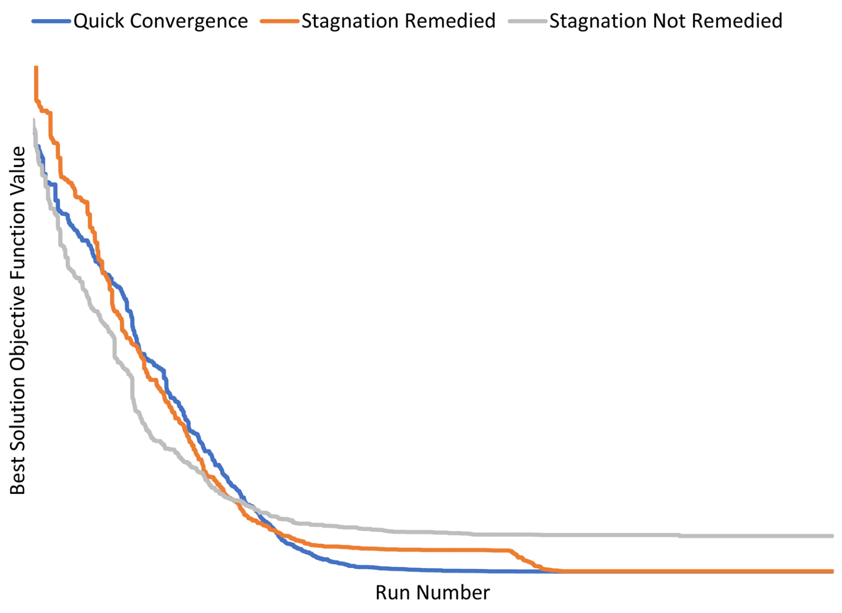

Figure 6. Convergence types.

5. Experimental Results

Table 3 below shows the results for each value of *n*. For each value of *n*, a solution was obtained (5) times. The mean objective function value is reported along with its standard deviation. The ratio of standard deviation to mean is less than 1% for each value of *n*. As such, the small variation in the mean provides a large degree of confidence in the estimates. Also reported is the objective function value when the classical MDS approach ("Eigen") is used.

As one can see, the search approach provides vastly superior results as compared to the classical approach ("Eigen"). Figure 7 details these findings.

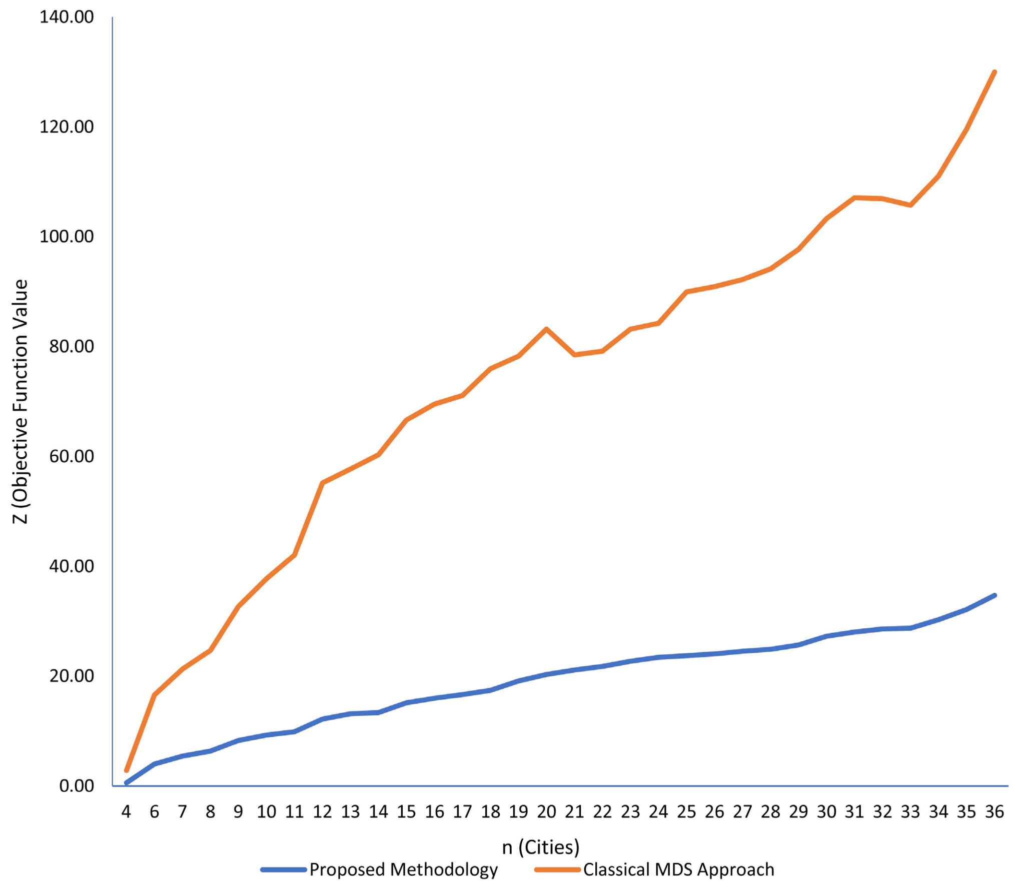

Figure 7. Number of cities (*n*) and objective function.

Table 3. Results of search heuristic.

n	Mean	Std Dev	Eigen	*n*	Mean	Std Dev	Eigen
4	0.60	0.03	2.81	21	21.11	0.09	78.49
5	1.40	0.07	5.46	22	21.78	0.10	79.15
6	3.99	0.03	16.59	23	22.70	0.11	83.17
7	5.43	0.03	21.29	24	23.42	0.10	84.26
8	6.35	0.07	24.67	25	23.71	0.04	89.93
9	8.29	0.07	32.70	26	24.05	0.05	90.91

Continued

10	9.28	0.06	37.74	27	24.52	0.14	92.21
11	9.89	0.03	42.03	28	24.88	0.16	94.15
12	12.18	0.08	55.17	29	25.68	0.25	97.70
13	13.15	0.08	57.70	30	27.26	0.06	103.29
14	13.36	0.12	60.32	31	28.04	0.19	107.08
15	15.15	0.09	66.63	32	28.60	0.20	106.93
16	15.97	0.18	69.54	33	28.73	0.11	105.73
17	16.66	0.05	71.10	34	30.27	0.09	111.02
18	17.42	0.08	75.96	35	32.11	0.07	119.55
19	19.11	0.08	78.25	36	34.72	0.15	130.00
20	20.31	0.11	83.18				

The blue line shows the mean objective function value associated with the presented search approach, as a function of the number of cities (n). The orange line shows the objective function value via the classical MDS approach as a function of n.

6. Concluding Comments

A search heuristic has been presented to convert a distance matrix into spatial coordinates. Some potential pitfalls encountered such as "stagnation" were encountered and remedied. The final results show objective function values that are near-optimal. Additionally, these results are superior to those obtained via classical MDS. Classical MDS is unable to compete with the direct search heuristic presented here.

As stated earlier, MDS can be used for a variety of applications. It so happens the application used here was to convert distance data into spatial coordinates. The intent of this research is not to just find spatial coordinates, but to illustrate an approach to obtain a good-quality solution to an MDS problem. If one just wanted spatial coordinates for locations, they could simply look them up on the internet. The example here is considered reasonable because the end-user should have a pre-conceived belief as to where the locations should be assisting with understanding the search process. This search process could be used for most any situation where it is desired to convert similarity (or dissimilarity data) into spatial data.

There are, of course, opportunities for research beyond this effort. More locations would be an interesting pursuit, as would consideration of multiple geographic regions (*i.e.*, continents). Of course, the search parameters could also be further explored. Other possible follow-up efforts could be related to other applications of MDS, such as consumer segmentation, information storage, and DNA sequence comparison.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Mboko Ibara, S. and Ossouna, D. (2021) Determinants of Multidimensional Poverty among the Under-Five: Illustration Based on Data from the Congo Multiple Indicator Cluster Survey. *Theoretical Economics Letters*, **11**, 363-380. https://doi.org/10.4236/tel.2021.112024

[2] Bianchi, L., Dorigo, M., Gambardella, L.M. and Gutjahr, W.J. (2009) A Survey on Metaheuristics for Stochastic Combinatorial Optimization. *Natural Computing*, **8**, 239-287. https://doi.org/10.1007/s11047-008-9098-4

[3] Glover, F. (1989) Tabu Search—Part 1. *ORSA Journal on Computing*, **1**, 190-206. https://doi.org/10.1287/ijoc.1.3.190

[4] Glover, F. (1990) Tabu Search—Part 2. *ORSA Journal on Computing*, **2**, 4-32. https://doi.org/10.1287/ijoc.2.1.4

[5] Kirkpatrick, S., Gelatt Jr., C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671-680. https://doi.org/10.1126/science.220.4598.671

[6] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic Publishers.

[7] Marden, J.I. (1995) Analyzing and Modeling Rank Data. Chapman & Hall, New York.

[8] Kruskal, J.B. (1964) Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, **29**, 1-27. https://doi.org/10.1007/BF02289565

[9] Johnson, R.A. and Wichern, D.W. (2018) Applied Multivariate Statistical Analysis. 6th Edition, Pearson.

[10] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953) Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, **21**, 1087. https://doi.org/10.1063/1.1699114

[11] Wilensky, U. and Rand, W. (2015) An Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo, MIT Press.

[12] McMullen, P. (2020) An Agent-Based Approach to the Newsvendor Problem with Price-Dependent Demand. *American Journal of Operations Research*, **10**, 101-110. https://doi.org/10.4236/ajor.2020.104006