World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

Comparative analysis of self-organizing maps and genetic algorithms for distance-minimizing representation of 2D and 3D point distributions

Mohammad Y. Saadeh [1,*] and Cris Koutsougeras [2]

[1] *Department of Engineering Technology, Northern Illinois University, DeKalb, IL 60115, USA.*
[2] *Department of Computer Science, Southeastern Louisiana University, Hammond, Louisiana 70402, USA.*

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(03), 2443-2454

Publication history: Received on 16 April 2025; revised on 21 June 2025; accepted on 24 June 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.3.1172

Abstract

The problem addressed in this work is how to position a small set of receivers among a much larger set of transmitters while optimizing some distance metrics which would lead to optimal communications especially in underwater exploration where signals degrade rapidly with distance. The set of transmitters are assumed to be in relatively slow motion, therefore the receivers must be constantly positioned while they follow the transmitters' changing distribution in space. As transmitters move, they are not statically allocated to receivers; they may change allocations. Thus, this problem entails dynamic clustering of the transmitters into clusters associated with the receivers and thus there is a need for determining the optimal placement of receivers. The problem is complex since the placement determines the clustering and the clustering determines the geometric medians which would be the optimal positions within each cluster. To address this complexity, this work compares a heuristic approach of Self Organizing Maps (SOMs) which maps the distribution density of the transmitters and a Genetic Algorithms (GAs) approach to optimize a distance based metric. The method was deployed on data points distributed within a 2D plane, then the search space was expanded to 3D cases and the clustering was carried out using SOM and GA. Overall, simulation results showed that GA outperformed SOM in optimizing the cost function, especially in 3D space.

Keywords: Self-Organizing Map; Genetic Algorithms; Clustering; Optimization

1. Introduction

SOM algorithm is an unsupervised learning approach that is designed to represent high dimensional datasets with a smaller and concise subset representation [1]. This approach was first introduced by Kohonen [2]. GA, on the other hand, was first introduced by Holland [3]. It is an evolutionary algorithm that mimics natural selection of the most fitted individuals for survival and reproduction [4]. Many researchers incorporated both SOM and GA either collaboratively or individually as search methods in clustering data sets. Su and Chang [5] proposed a method of applying GA to form a Self-Organizing Feature Map (SOFM) through providing estimates of the number and the locations of clusters from a data set. They adopted a two-stage training procedure to accelerate the search procedure. Jin et al [6] developed an integrated self-organizing map (ISOM) for the traveling salesman problem (TSP) while implementing three mechanisms: the local optimality of the traditional SOM, the global optimality of the expanding SOM, and the elastic force constraint in the elastic net. According to the researchers, the suggested learning rule enabled the ISOM to generate near optimal solutions. Amor and Rettinger [7] proposed an exploration technique for GAs that uses SOMs to analyze data of the evolution process which they utilized to enhance the search strategy. According to Kan et al [8], they proposed that SOM enhanced the search performance of a real-coded RCGA and found that that the use of the sub-population search algorithm improved the local search performance of the RCGA, which suggested that SOM-GA was able to find better solutions in shorter CPU time than RCGA. Other works considered GA-SOM integration in various

* Corresponding author: Mohammad Y. Saadeh.

Copyright © 2025 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0.

computing-applications. For instance, Kuo et al [9] proposed an automatic computer color separating system for printed fabrics that integrated a GA and a SOM network to automatically separate printed colors to eliminate the time-consuming manual color segmentation and registration that is typically done in the industry. While Prabhakar et al [10] performed a comparative study GA and SOM to detect the botnet network traffic to cluster network traffic from input dataset as normal and malicious. They found that SOM performed the clustering faster than GA. Although GA took longer time to complete the clustering process, it gave more accurate results. In Ahmed et al [11], the researchers clustered research papers based on titles and keywords a using the SOM Algorithm with GA-optimized parameters and on a Word2vec pre-trained model to identify the similarity between keywords.

A weight design method based on SOM that can be integrated with decomposition-based evolutionary algorithms to solve many-objective optimization problems was proposed [12]. While in [13], they introduced an optimization approach that included a clustering phase by the SOM and two GA phases to solve the TSP problem. They found that GA allowed for the whole problem to be solved all at once.

This work deploys two AI search methods (SOM and GA) to cluster a group of slowly-moving transmitting explorers (data set) using a fixed set of receiving followers which imply a spatial clustering of the distribution of the explorers. An example case of this is in underwater exploration where multiple explorer vessels moving underwater need to communicate with service vessels. Due to the nature of the application, the explorers are allowed to move freely within the service area, so the service vessels must be constantly repositioned to optimize certain aggregate distance metrics as the signals degrade rapidly with distance. Two methods (SOM and GA) are assessed on their ability to continue to cluster them optimally in quasi-real time. Assuming service vessels are surface bound, the above problem is addressed in the 2D plane of the sea surface, but then the work is generalized and data points in 3D space are included. The same simulation is carried out after all data points are allowed to roam the space freely in incremental motion.

1.1. Problem Formulation

To describe the problem, consider the following example. There are N explorers (i.e. transmitters or, in the case of seabed exploration, remotely operated vehicle ROVs) that are dispersed in an area and their locations are known to K followers (service vessels). The K followers need to be placed around that area with the following goal. Each explorer transmitter will only tune into the receiver that is physically closest to it. Any placement of the receivers automatically clusters the transmitters into K sets, one for each of the receivers. The set corresponding to a particular receiver Ri consists of those transmitters which will be closest to Ri than to any other receiver. In other words, the set corresponding to Ri would be those transmitters which fall within Ri's Voronoi field. With any particular placement of the receivers, the corresponding Voronoi fields of the receiver locations imply a clustering of the transmitters. Further, it is desired that each of the receivers is placed so that the sum of its distances from all the transmitters in its cluster is minimal; i.e. it is the geometric median (Fermat-Weber point) of its cluster. The problem at hand is, given the distribution of the transmitters, how to disperse the receivers so that ideally each is the geometric median of its cluster and the total sum of these distances in all clusters is minimized. The complexity of the problem lies in that the placement defines the clustering, so moving a receiver would change the clustering but a receiver must be moved to minimize the total distance within its cluster. Thus, the placement affects both the clustering and the distance sum.

Consider the case of underwater exploration where submerged vessels explore underwater and need to communicate data to receiver vessels. Because of the degradation of signals underwater, the receivers need to be as close as possible to the exploration vessels with which they communicate. Receivers have a one to many relationships with explorers; one receiver would be assigned to a set of explorers. As the explorers move about, the receivers must also move in a way that minimizes their distances from their allocated explorers. However, explorers may get reallocated new receivers if they move to an area which can be better served by a different receiver.

To express the full scale problem: Given a distribution of N data points (e.g. locations of transmitters), place $K << N$ markers (e.g. receivers), so that each original point is assigned to its nearest marker (through the Voronoi partition associated with the positioning of the markers) with the goal to minimize the total distance between each point and its assigned marker. To formally describe the problem let's define:

- Let $P=\{P_1,P_2,\ldots,P_N\}\subset R^d$ be the set of N original data points.
- Let $M=\{M_1,M_2,\ldots,M_K\}\subset R^d$ be the set of K markers we wish to place (where $K \ll N$).
- Each point P_i is assigned to its nearest marker M_j.

The objective is to optimize the distance metric:

$$Cost\ Function = \ min \sum_{i=1}^{N} min_{j=1}^{K} \|P_i \ - \ M_j\| \quad \text{.....(1)}$$

This problem would manifest itself in various applications like facility location, communications, vector quantization, etc.

This is a problem where a *K-means* method can be applied but this method is sensitive to initial placement as well as the distribution. The Self Organizing Maps (SOMs) is a method which also essentially clusters data points but it is better in some respects like preserving the topology of the data. While *K-means* focuses on finding the optimal separation between clusters, SOMs focuses on the topological relationships of the data as they try to map the distribution density in the space of the data. SOMs approximate the means of the implied clusters in positioning the markers therefore they are expected to be a good heuristic for the desired optimization because they concurrently address the issues of figuring out the clustering and positioning of markers near the mean, especially since it has to be performed continuously, adaptively, and smoothly as the topology of the data (transmitters) changes. In this work, we compare the SOMs method against a more standard optimization method, the Genetic Algorithms (GAs) approach which targets the optimization of the distance metric more directly.

For the remaining of this work, we assume that there are 200 data points randomly distributed within a 200 x 200 Cartesian coordinate. Also, we assume that five clusters are needed to represent these data points. Then, both SOM and GA were deployed to search for the optimal clusters' positions to minimize the cost function in Eqn.1. In the case of seabed explorations and similar applications, these data points (explorers) will be freely roaming their working space. The motion is assumed slow enough to be able to do sufficient iterations of the algorithms for convergence in-between movements. In this work, we use five iterations to measure the effectiveness of both: SOM and GA in responding to small movements in the data points' distribution. Each data points movement was capped at <10% of the operating space. The scope of the work is then expanded to include data points in the space (3D) which is 200 x 200 x 200. We repeated the exact procedure by capturing the optimal clusters' positions in SOM and GA, then the data points were allowed to roam the space freely and five iterations of movements (changes in data points' distribution) were captured and assessed.

2. Methods

2.1. Self-organizing maps

Self-Organizing Maps (SOMs) is an elegant method to map the distribution of a data set with a much smaller set of markers. It starts with a random (but usually uniform) distribution of markers over the space of the data to be mapped and iterates with small movements of the markers until convergence (i.e. no significant further change). Specifically, each iteration over the entire data set, focuses on one data item at a time, identifies the marker that is closest to that item (called the winner marker) and repositions this winner marker by moving it closer to the data item in a step that is a small percentage of their distance as shown in Figure 1:

$$\boldsymbol{M_{new} = M + \alpha(S\text{-}M)} \quad \text{where } 0 < \boldsymbol{\alpha} < 1 \quad \boldsymbol{\ldots\ldots\ldots\ldots} \quad (2)$$

This iterative process continues until there are no longer any significant changes in the positions of the markers.

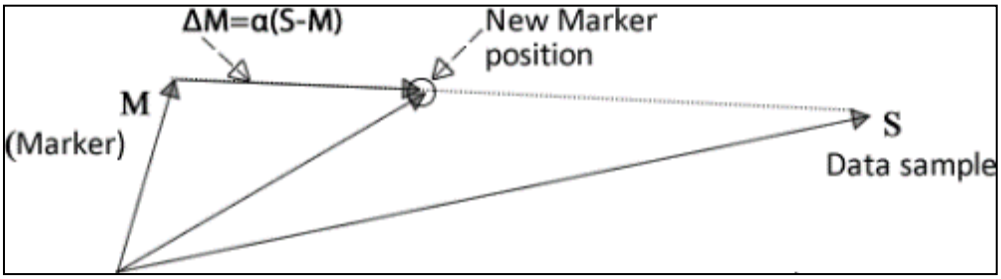


Figure 1 Marker movement towards the sample

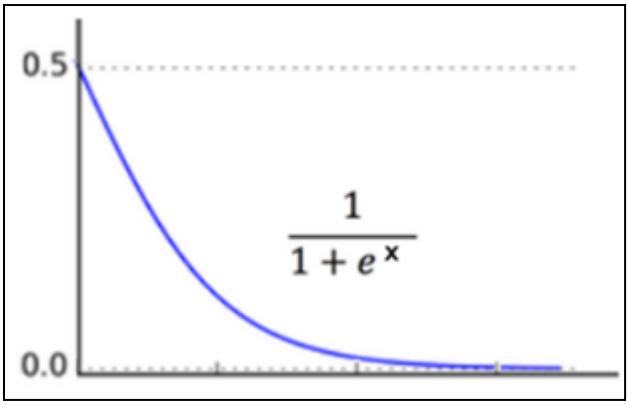


Figure 2 Sigmoid tapering off

The variant that we use here allows other markers which are neighbors of the winner to also move towards the focused data point but at a smaller fraction that is a function of their distance from the winner (i.e. how close neighbors they are). Figure 2 shows a coefficient used for a neighbor marker which tapers off with the Euclidian distance x between the winner and the corresponding neighbor; so a neighbor marker N is moved by:

$$N_{new} = N + \alpha(d_{NM})(S-N) \quad \text{..... (3)}$$

where $\alpha()$ is a function of the distance d_{NM} and defined as:

$$\alpha(d_{NM}) = \beta/(1 + e^{-d_{NM}}) \quad \text{..... (4)}$$

$$\text{with } \beta = 0.99^{(i-1)} \quad \text{..... (5)}$$

where β is a correction factor that is set to diminish gradually to stabilize the search process as it converges (i.e. followers begin to establish their territories), and i is the iteration number. At every iteration, every marker will be using Eqn.3 where α which is a function of each distance from the marker to the winner is computed according to Eqn.4.

SOMs is a good candidate for this problem not only because it essentially maps the distribution density of the data, but also because when a marker has settled after being continuously "pulled" by the data items in its cluster, it rests at or near the centroid of its cluster and thus it is a good candidate for a solution to the problem at hand. However, because it is not a direct optimization method, we compare it with a search-based optimization method and specifically a Genetic Algorithm approach.

2.2. Genetic Algorithms

This work attempts to compare the SOM method to a heuristic search-based optimization method (Genetic Algorithm) to study the performance of each method and the speed and accuracy of the resulting solutions. The GA agent performs a heuristic search to identify the spatial location of the clusters and iteratively replaces lower fitting chromosomes with higher fitting ones in a process that mimics natural selection. The overall goal of the GA is to minimize a cost function; thus, identifying the correct function is crucial to the success of the GA.

The GA proposed in this work begins with a suggested generation of *C* chromosomes with all parameters (i.e. Cartesian coordinates of the Nodes) coded within each chromosome. Each chromosome is then decoded and a fitness function is used to rank the chromosomes. The same cost function that was implemented in the SOM is used here as well. After ranking them, the GA propagates the best 50% of chromosomes to the next generation to ensure the successful traits are preserved within the population. The remaining 50% of the population is pruned and replaced with fresh population that is the result of the reproduction between the successful chromosomes that were preserved. This process is explained in more detail later.

Each chromosome holds information about the spatial coordinates of each suggested cluster which consists of many variables. Thus, a large population is needed in order to encompass a globally optimal solution. Hence, each population consists of 200 chromosomes and each chromosome contains independent variables (genes) that are coded within each chromosome, as follows:

- In the Cartesian coordinate space, two genes are needed to code the coordinates of each cluster origin. Thus, each chromosomes contains 2**K* genes (x and y coordinates of each marker).
- In the XYZ space (discussed in Section 3.5), three genes are needed to code the spatial coordinates of each cluster origin. Thus, each chromosomes contains 3**K* genes (x, y and z coordinates of each marker).

These genes are arranged in chromosomes to form the initial population; each population undergoes pruning, crossover, and mutation to generate the next generation.

2.2.1. Evaluation of Generations

Each chromosome is evaluated by extracting the data points of its genes and decoding them into the different parameters (X, Y and Z coordinates of each cluster origin). Then, the distances between each data point and the current cluster origins are computed to determine the cluster origin with the shortest distance. The shortest distances are then added together to determine the closeness of the suggested cluster origins to the data points they represent.

In the Planar System:

$$C = \left[M_{x1}, M_{y1}, M_{x2}, M_{y2}, \ldots., M_{xm}, M_{ym}\right]$$

$$\boldsymbol{D_i} = \sqrt{\left(x_i - M_{xj}\right)^2 + \left(y_i - M_{yj}\right)^2} \qquad \forall\, j \in [1, m] \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

$$Closeness = \sum_{i=1}^{n} \min\ (\boldsymbol{D_i}) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (7)$$

where M_{xj} and M_{yj} are the x and y coordinates of cluster origin *j*, respectively. $\boldsymbol{D_i}$ is the vector representing the distances between data point *i* and all *j* cluster origins.

2.2.2. Pruning

The chromosomes are ranked according to their closeness (fitness) function in Eqn. 7. Half of the chromosomes (the least fitted ones) are removed from the next generation's population while the most fitted 50% of the current population will propagate to the next generation. The remaining 50% of the next generation will be replaced with offspring from the surviving chromosomes (the most fitted ones), using crossover and mutation as described below.

2.2.3. Crossover

The offspring consists 50% of the next generation population and allows the GA to continue to combine successful traits (genes) from different chromosomes (parents) into newer ones (offspring) in a heuristic and natural selection process.

Two parents are selected randomly from the pool of parents to undergo a crossover process. The chromosomes with the highest fitness values are weighed higher. Thus, they have higher chances of getting selected for the reproduction process. This work uses random two-point crossover to exchange genes across the two chromosomes, as explained in the following illustration.

Chromosome_1 = [a b c | d e f g | h i]
Chromosome_2 = [j k l | m n o p | q r]

The continuous genes follow a randomly selected weighted average between 0 and 1.

For example, the example above shows a two-point crossover between Chromosome 1 and Chromosome 2. The two new offspring will be as follows:

Chromosome_1* = [a b c **x1 y1 z1 w1** h i]
Chromosome_2* = [j k l **x2 y2 z2 w2** q r]

Where:

$$x_1 = d * w + m (1-w) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (8)$$

$$x_2 = m * w + d (1-w)$$

w is a randomly selected weight between 0 and 1. The same crossover will take place to find y_1 through w_2. In this work, all genes are purely continuous with no discrete components and will undergo the same calculation as in Eqn. 8.

2.2.4. Mutation

The mutation is the process that allows the reproduction to impose random genetics into the new population that can lead to breakthroughs in the search process through the exploitation of parameters. This allows new information to be presented and may lead to accelerated convergence in a global minimum. A rate of 3% was used through this work.

Figure 3 [14] below visualizes the GA processes to identify new generations and the GA-System Identification interaction.

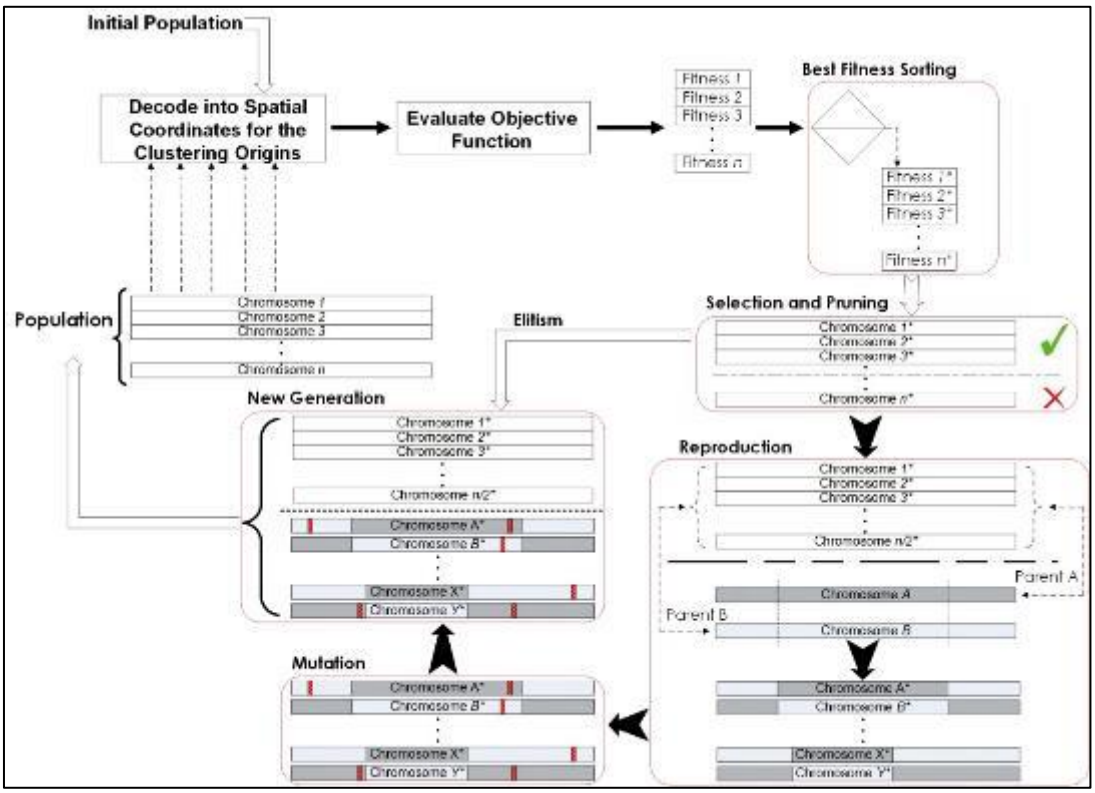


Figure 3 Depiction of the GA process [14]

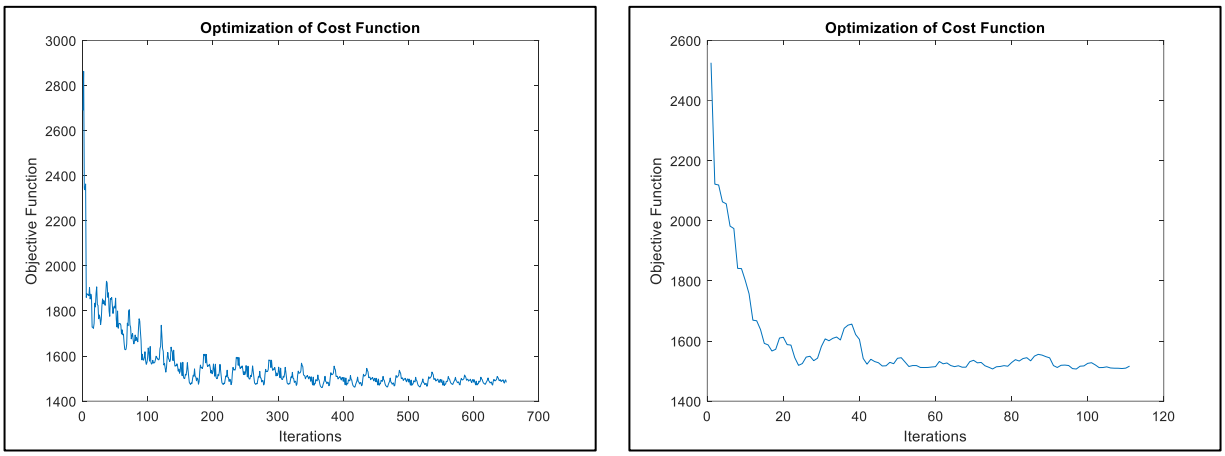
3. Results

3.1. Effect of Learning Rate

The learning rate (Eqn. 4) should properly be selected to attenuate the fluctuation in the solution and allow it to stabilize and converge. It was noted that the SOM was sensitive to the correction factor β. The faster the convergence of the SOM, the more likely it will converge at a non-global minimum solution. If properly selected, this factor allows the search method to navigate the surrounding space around the current solution to fine-tune it. The tradeoffs will be the longer search process, the fluctuation around the final destinations, and the additional iterations that it will go through before converging.

To better illustrate this observation, the correction factor β was initially set to 0.999 and the simulation was carried out. The search was set to terminate if the percentage of cost function reduction was less than 1% for 20 consecutive iterations. Figure 4 shows the progression of the SOM towards the optimal solution (clusters locations) using two different learning rates. With the correction factor set at 0.999, the cost function was optimized at 1482 after 650

iterations. After adjusting the correction factor to 0.99, the cost function was reduced to 1516 after 111 iterations. The two values were very comparable (~2% difference), however the number of iterations were significantly reduced (83% reduction). Thus, the correction factor was fixed at 0.99 throughout the remaining simulations.



a. Correction factor β = 0.999 b. Correction factor β = 0.99

Figure 4 Simulation results for the data set in the Planar System

3.2. Simulation Results of Cartesian Space:

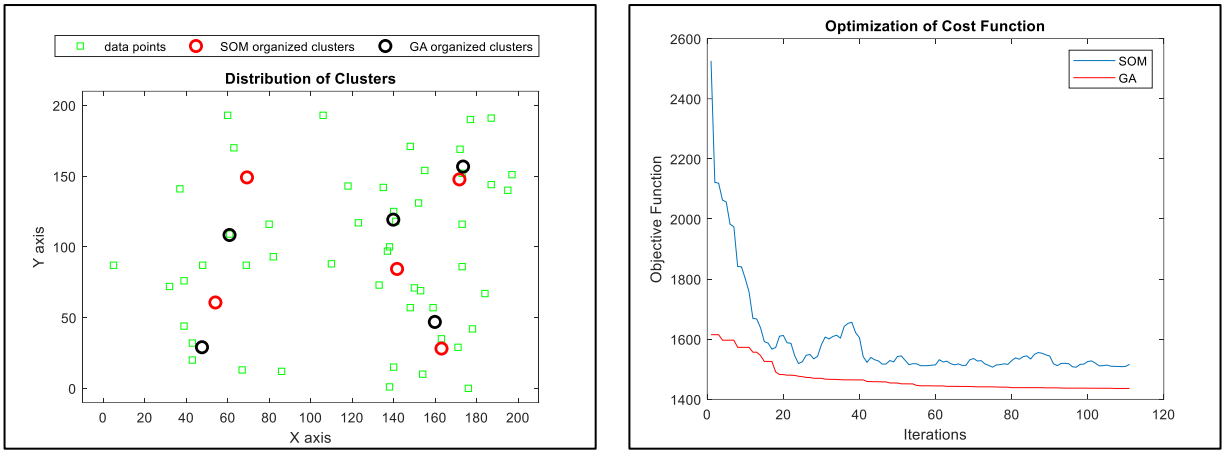
All simulations were carried out using MATLAB (MathWorks, Inc.). The simulation (optimization) processes for both SOM and GA iterate until either there is no further significant reduction in the cost function is achieved over a preset range of successive generations or that the maximum number of generations is exhausted.

The initial cluster origins were randomly selected at different locations then they emerge into their final destinations as the process iterates (while minimizing the cost function in Eqn.7). The same data set (data points) were kept during the GA process and the generated cluster origins were added on the same figures to show the different set of cluster origins that were achieved. Later, the data points were allowed to move freely within the operating range and both SOM and GA were deployed again to reposition the clusters to their optimal spatial locations.

It was noted that the SOM yielded identical results each time the search was performed, regardless of the initial cluster spatial coordinates, which demonstrates its stability and independence of the initial position of the clusters. One limitation that the SOM has is that the movements of the cluster positions are incremental, systematic, and sequential (i.e. small movement in all directions). Although the incremental motions (clusters' positioning) may not necessarily minimize the objective function momentarily, it guarantees that an optimal solution will be reached with successive incremental motions (updates) on the clusters' positions. Also, SOM relies on the proper selection of the learning rate (and correction factor). A reduced learning rate will make the SOM rigid and will converge at non-optimal cluster locations, whereas a higher learning rate will cause it to fluctuate around the optimal solution and take longer iterations to settle.

Due to the nature of the search in the SOM and GA, the solution set (clusters) will likely be different for each search process. GA, on the other hand, doesn't have an incremental build-up that depends on the previous location of the cluster origins. The heuristic nature of the search and the ability to navigate the entire space through crossover and mutation can lead to breakthroughs and multiple movement in one or more coordinates to optimize the cost function. The main drawback is the need for the GA to evaluate the entire population before it updates the markers' locations. Thus, requiring additional computational power.

The following figure shows the comparison between SOM and GA in terms of the cost function as well as the location of clusters in each case. GA was able to reduce the cost function slightly to 1420 (from 1515) with different cluster distribution and spatial coordinates.



(a) Distribution of Clusters **(b)** Optimization of the Cost Function

Figure 5 Simulation results for the data set in the Planar System

Figure 5.a shows the distribution of clusters in the planar system. The spatial coordinates of the cluster origins are within the vicinity of each other. Figure 5.b shows the minimization of the cost function over the iterations. The results of the GA surpassed those of SOM for all simulation runs, as shown in Figure 5. The SOM stops the search process when there is no significant reduction in the cost function over a certain number of iterations (i.e. less than 1% reduction for 20 consecutive iterations). To compare results with the GA, the GA was allowed to iterate for the same number of iterations of the SOM.

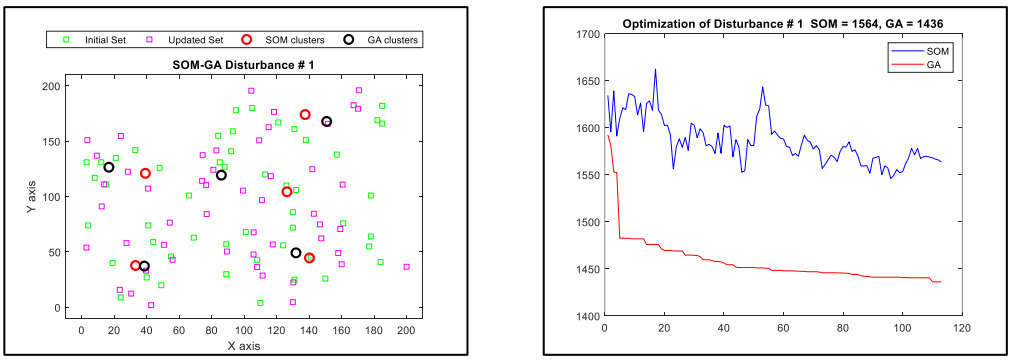
3.3. Dynamic Simulation

The task considered in this work (rovers navigating the seabed) is dynamic in nature. Thus, these rovers (explorers) are constantly roaming the seabed to perform different operations. The followers should also adapt their spatial locations to continue to serve the explorers. While the explorers are constantly moving, the followers wouldn't necessarily follow each individual motion in real-time. Rather, the motion of the explorers is first discretized (for the benefit of coding and assessing the effectiveness of each search method) and the followers will then update their spatial locations.

It is assumed that the explorers' locations will be captured few times, during which they will be able to cross up to 10% of the entire space in any arbitrary direction. The search will then update the follower's locations (clusters) according to the new distribution of the explorers. Both SOM and GA are simulated during these increments. A total of five increments (updates) were captured and simulated consecutively.

3.4. Simulation Results of Cartesian Planar Coordinates

During the disturbance testing (dynamic testing), GA outperformed the SOM both, in cost optimization as well as speed of convergence, as shown in Figure 6 below.



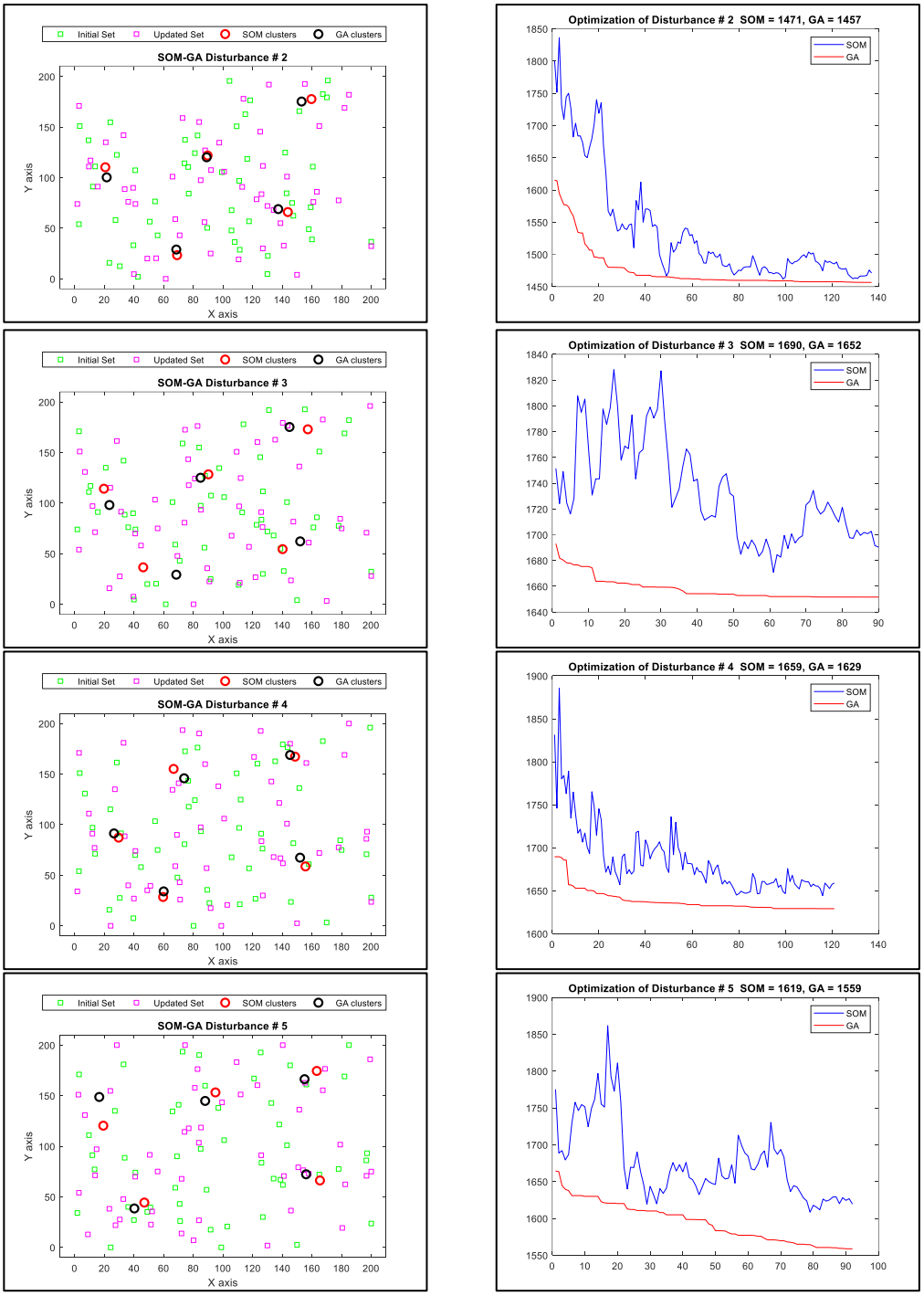


Figure 6 Simulation results for the dynamic representation of data set in the Planar System

SOM have a direct and immediate impact on the placement of the clusters. This explains the fluctuation in the cost function (it varies over iterations). While the GA retains the locations and allows it to change only if the new locations achieve more optimal outcome; thus, it gradually and steadily reduces the cost function. In both SOM and GA, the search method uses the previous spatial locations of the followers (from the previous iteration) as initial positions and attempts to place them in their new positions.

3.5. Simulation Results of Cartesian Space (3D):

The SOM and GA searches were extended to cluster data sets in the Cartesian space. Representing this work in 3D provides a new area of practical and interesting implementation in many areas (i.e. material science, cells in tissue, crystallization of atoms, and guided wrinkling).

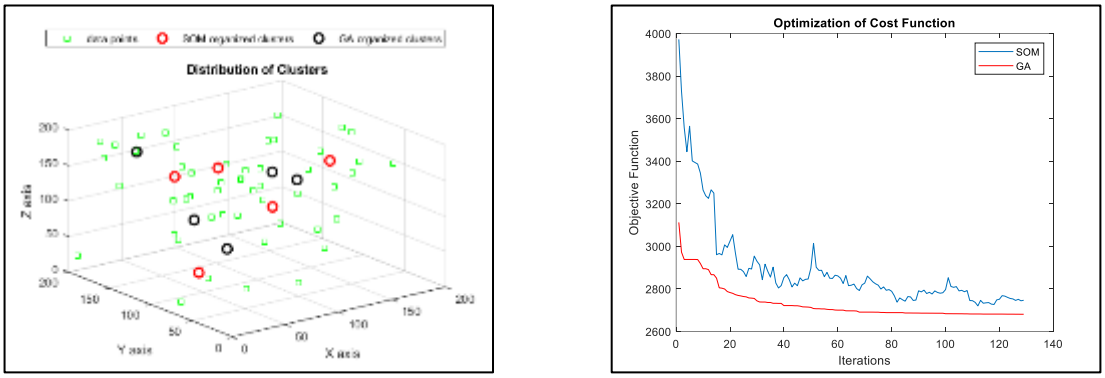
The additional coordinate did not have any significant change on the SOM; however, the number of genes in each chromosome has increased (as shown in Eqn. 9 below).

In the Cartesian Space:

$$C = [M_{x1}, M_{y1}, M_{z1}, M_{x2}, M_{y2}, M_{z2}, \ldots., M_{xm}, M_{ym}, M_{zm}]$$

$$\boldsymbol{D_i} = \sqrt{(x_i - M_{xj})^2 + (y_i - M_{yj})^2 + (z_i - M_{zj})^2} \qquad \forall\, j \in [1, m] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (9)$$

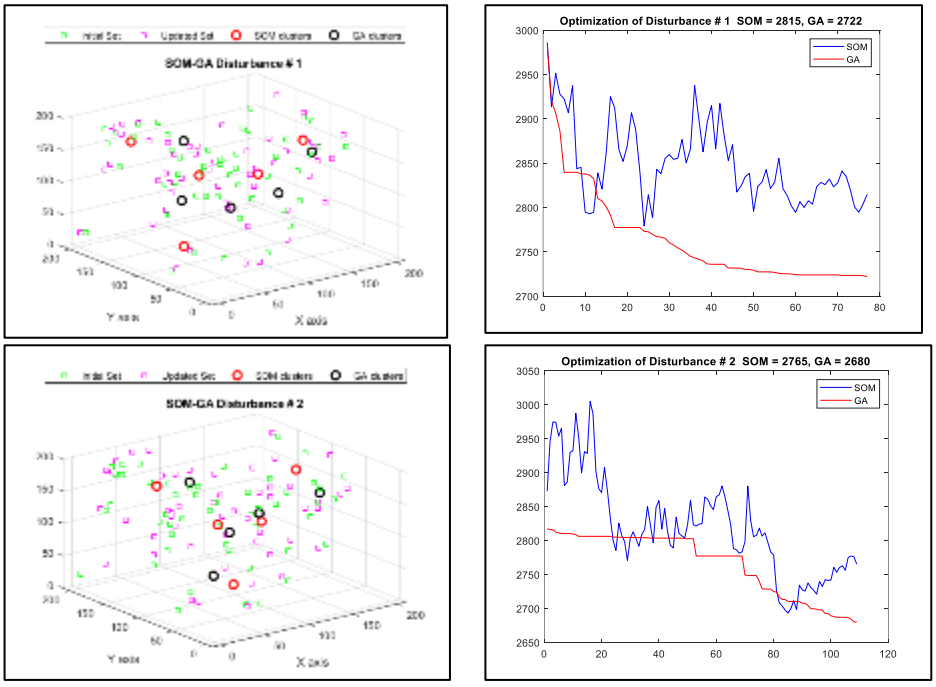
Despite the change in the GA and the inclusion of an additional coordinate, the same principles of search were applied. Figure 7 shows the simulation results of both the SOM and GA using the same data points. SOM optimized the cost function to 2751 while the GA reduced it to 2686.



a. Distribution of the Clusters in the Cartesian Space b. Optimization results

Figure 7 Simulation results for the data set in the Cartesian Space

It was observed that SOM was (again) independent of the initial clustering locations even when the correction factor was changed to 0.999. The same analysis was carried out in the space. The data points were allowed to roam the space freely to create five successive spatial distributions and both SOM and GA were deployed to find the optimal clusters' locations. The simulation results appear in Figure 8 below.



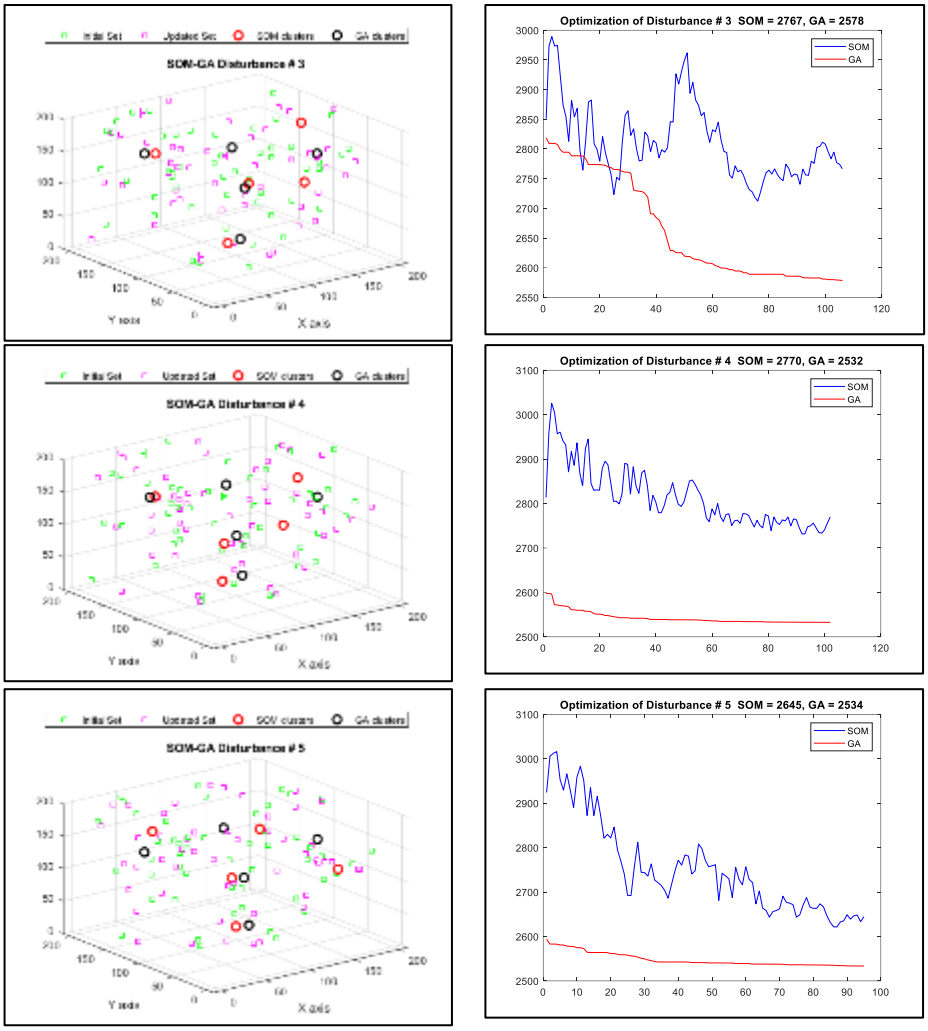


Figure 8 Simulation results for the dynamic representation of data set in the Space (3D)

The GA outperformed SOM in all five cases, with a reduction of cost function between 3-8% compared to SOM.

4. Discussion

Search methods used in this work have been successful at optimizing the cost function to achieve optimal clusters' positioning in planar and space coordinates. Representing a large distribution of data points using smaller number of clusters according to certain criteria has many practical applications in science and computing. The current work shows that GA slightly outperformed SOM in searching for clusters, especially with scaling up the medium from 2D planar search to 3D space search. GA benefits from its heuristic and loosely structured search process that allows it to simultaneously exploit many parameters within the search space. This is illustrated in the significant reduction in the cost function within the first few generations.

SOM, on the other hand, is a structured search method that constantly and incrementally updates the solution set. To allow it to navigate the search space, SOM should freely position the clusters especially at the beginning of the search. However, this freedom should be diminished (using a diminishing correction factor) as the search process approaches global minimum to achieve positioning stability.

5. Conclusion

This work concerns an optimization problem for which Self Organizing Maps (SOMs) as well as Genetic Algorithms (GAs) were applied to solve. The problem is a recurring one in many areas and so solutions to it are of wider interest. Clustering data points into a smaller number of representatives (centroids) has many practical applications. For instance, large companies and public service entities use it to place warehouses, cell towers, hospitals and other facilities

that optimize such measures as: cost, response time, or service coverage. In computing, this is used in many areas including signal and image compression as well as in the optimal placement of routers and servers to maximize coverage while minimizing latency. It was shown that GA outperformed SOM in optimizing the cost function due to the heuristic nature of its search process.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Guérin, A., Chauvet, P. and Saubion, F. (2024), "A survey on recent advances in self-organizing maps", arXiv:2501.08416, https://doi.org/10.48550/arXiv.2501.08416

[2] Kohonen, T. (1982), "Self-organized formation of topologically correct feature maps", Biological Cybernetics, Vol. 43, No. 1, pp. 59–69.

[3] Holland, J. (1992), Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, USA.

[4] Alhijawi, B. and Awajan, A. (2024), "Genetic algorithms: theory, genetic operators, solutions, and applications", Evolutionary Intelligence, Vol. 17, pp. 1245–1256. https://doi.org/10.1007/s12065-023-00822-6

[5] Su, M. and Chang, H. (1998), "Genetic-algorithms-based approach to self-organizing feature map and its application in cluster analysis", 1998 IEEE International Joint Conference on Neural Networks Proceedings, (Cat. No.98CH36227), Anchorage, AK, USA, pp. 735-740 Vol.1, doi: 10.1109/IJCNN.1998.682372.

[6] Jin, N. H., Leung, N. K., Wong, N. M., and Xu, N. Z. (2003), "An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem", IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics), Vol. 33, No. 6, pp. 877–888. https://doi.org/10.1109/tsmcb.2002.804367

[7] Amor, H. and Rettinger, A (2005), "Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation", In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 1531–1538. https://doi.org/10.1145/1068009.1068250

[8] Kan, S., Fei, Z., and Kita, E. (2009), "Application of self-organizing maps to genetic algorithms", WIT Transactions on the Built Environment, Vol. 106, pp. 3–11. https://doi.org/10.2495/op090011

[9] Kuo, C. J., Kao, C., and Chiu, C. (2009), "Integrating a genetic algorithm and a self-organizing map network for an automatically separating color printed fabric system", Textile Research Journal, Vol. 79, Issue. 13, pp. 1235–1244. https://doi.org/10.1177/0040517509102386

[10] Prabhakar, S. Y., Parganiha, P., Viswanatham, V. M., and Nirmala, M. (2017), "Comparison between genetic algorithm and self organizing map to detect botnet network traffic", IOP Conference Series Materials Science and Engineering, Vol. 263, Issue. 4 (Article 042103). https://doi.org/10.1088/1757-899x/263/4/042103

[11] Ahmed, R. Salama, C. and Mahdi, H. (2020), "Clustering research papers using genetic algorithm optimized self-organizing maps", 15th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 2020, pp. 1-6, doi: 10.1109/ICCES51560.2020.9334573.

[12] Gu, F. and Cheung, Y. (2018), "Self-organizing map-based weight design for decomposition-based many-objective evolutionary algorithm", In IEEE Transactions on Evolutionary Computation, Vol. 22, No. 2, pp. 211-225, doi: 10.1109/TEVC.2017.2695579.

[13] Nourmohammadzadeh, A. and Voß, S. (2021), "Hybridising self-organising maps with genetic algorithms", In Learning and Intelligent Optimization: 15th International Conference, LION 15, Athens, Greece, Vol. 12931, pp. 265–282. https://doi.org/10.1007/978-3-030-92121-7_22

[14] Saadeh, M. (2023), "Hybrid genetic algorithm-system identification approach to model force sensing resistors", Journal of Intelligent Material Systems and Structures, Vol. 34, No. 17. doi:10.1177/1045389X231167178