Into the Void: Mapping the Unseen Gaps in High Dimensional Data

Xinyu Zhang, Tyler Estro, Geoff Kuenning, *Life Member, IEEE*,
Erez Zadok, *Member, IEEE*, Klaus Mueller, *Fellow, IEEE*

***Abstract*—We present a comprehensive pipeline, integrated with a visual analytics system called GapMiner, capable of exploring and exploiting untapped opportunities within the empty regions of high-dimensional datasets. Our approach utilizes a novel Empty-Space Search Algorithm (ESA) to identify the center points of these uncharted voids, which represent reservoirs for potentially valuable new configurations. Initially, this process is guided by user interactions through GapMiner, which visualizes Empty-Space Configurations (ESCs) within the context of the dataset and allows domain experts to explore and refine ESCs for subsequent validation in domain experiments or simulations. These activities iteratively enhance the dataset and contribute to training a connected deep neural network (DNN). As training progresses, the DNN gradually assumes the role of identifying and validating high-potential ESCs, reducing the need for direct user involvement. Once the DNN achieves sufficient accuracy, it autonomously guides the exploration of optimal configurations by predicting performance and refining configurations through a combination of gradient ascent and improved empty-space searches. Domain experts were actively involved throughout the system's development. Our findings demonstrate that this methodology consistently generates superior novel configurations compared to conventional randomization-based approaches. We illustrate its effectiveness in multiple case studies with diverse objectives.**

***Index Terms*—High-dimensional data, multivariate data, empty space, data augmentation, configuration space, parameter optimization**

I. Introduction

THIS paper focuses on a methodology for effectively discovering "empty spaces"—regions where data points are absent—in multivariate and high-dimensional (high-D) datasets. Identifying and exploring these empty spaces is both a challenge and an opportunity. The challenge is rooted in the *curse of dimensionality*, which is the exponential increase in volume and data sparsity associated with adding dimensions [4]. It makes the search for meaningful empty spaces increasingly complex, even for just a moderate number of attributes parameterizing the data.

Overcoming this complexity is not merely academic. It directly impacts the practicality of discovering and verifying new, unknown, and yet unimagined configurations that might reside within these empty spaces. Advocating for a configuration-discovery technique that can explore unconventional or even radical changes brought by unknown configurations and parameter settings offers a powerful alternative to conventional parameter tuning and optimization. These tasks are increasingly recognized as high priorities across diverse fields, including aerospace engineering [7], manufacturing [14], computer systems [1], personalized healthcare [55], and others, where techniques of this sort often address optimization problems with multiple objectives.

The emergence of crossover cars and hybrid vehicles illustrates well the high potential of exploring large parameter spaces to discover unique and unexpected configurations. Crossovers blend features from different vehicle types, offering drivers the versatility of an SUV with the agility of a sedan. Similarly, hybrids integrate gasoline engines with electric motors, improving fuel efficiency and reducing emissions without sacrificing performance. These examples demonstrate how investigating gaps within extensive parameter spaces can challenge conventional design constraints and embrace unconventional configurations. Clearly, innovation of this sort can also occur in lesser contexts.

While ingenious configurations become obvious once discovered, there are a multitude of them that defy practicality. Also, more often than not, high cost and substantial effort are required to obtain or simulate a hypothesized configuration to verify its merit. A human expert is often the best judge to decide whether to take up the risk of engaging in such a testing effort at all. But even with the human in the loop, the challenge lies in ideating meritorious configurations in the presence of this massive realm of possibilities.

To illustrate this challenge, consider a 4-dimensional parameter space with 50 levels for each dimension. This results in $50^4 = 6.25$ million possible configurations. Suppose we have data on the merit of 10,000 configurations from previous experiments. This means we have information on only 10,000/6,250,000 = 0.16% of the parameter space. While it is impractical to expect valid data at every location of the parameter space, the challenge lies in identifying *which* configurations are most useful. This uncertainty underscores the importance of intelligent sampling and discovery techniques to uncover valuable and innovative solutions.

While AI holds promise for replacing human experts in this search, it requires ample high-quality training data to be effective. Inspired by human-in-the-loop (HITL) [38] machine learning, our pipeline integrates the training of a deep neural network (DNN) that evolves alongside the exploration process. This DNN aids in evaluating identified configurations and improves with accumulated data, enhancing search efficiency for verifying new configurations.

Conceptually, our method looks for gaps in high-D data spaces. While in theory the pairwise distances among data points in high-D space tend to be normally distributed with small variance, in practice, however, data configurations aggregate into *hubs*—points that occur more often in k-neighborhoods of nearby points than others [43]. Likewise, there are also *anti-hubs*—points that are unusually far away from most other points. When these points

Xinyu Zhang, Tyler Estro, Erez Zadok and Klaus Mueller are with the Department of Computer Science, Stony Brook University, New York. E-mail:{zhang146 | testro | ezk | mueller}@cs.stonybrook.edu .

Geoff Kuenning is with the Department of Computer Science, Harvey Mudd College, Claremont, California. E-mail: geoff@cs.hmc.edu .

exist, they are referred to as outliers, hiding in gaps and sparsly occupied pockets of the data space. Essentially, our method looks for *hypothetical* outliers—thus-far unsampled configurations that might bear promise or are adversarial.

The major contributions of our paper are hence as follows:

- A scalable, parallelizable *Empty-space Search Algorithm (ESA)* that can identify empty spaces in numerical continuous high-D datasets.
- A visual analytics system, *GapMiner*, by which users can further modify the identified Empty-Space Configurations (ESC).
- A Human-in-the-Loop (HITL) to AI pipeline that trains an AI agent (a DNN) for eventual ESC search autonomy.
- A dimension-reduction method that allows the visualization of the neighbor distributions around empty spaces.
- Several case studies that demonstrate the effectiveness of our methodology in diverse application domains and objectives.
- A user study that evaluates our methodology and implementation.

In the following, Sec. II presents related work. Sec. III gives an overview of our three-phase workflow. Sec. IV describes our empty-space search algorithm. Sec. V introduces our visual analytics system, GapMiner. Sec. VI explains our Human-in-the-Loop to AI pipeline. Sec. VII showcases our method applied to the computer systems domain and Sec. VIII presents an associated user study. Sec. IX describes one further case study. Sec. X concludes and discusses future work.

II. Related Work

The research literature on the specific topic of empty-space visualization is sparse; we only know of two research groups who tackled this.

Strnad *et al.* [48] investigated the identification of empty spaces in protein structures. However, their study was limited to 3D space and did not address the concept of empty spaces in other fields. Additionally, their algorithm, which relies on Delaunay Triangulation, faces scalability issues in higher dimensions, as discussed in Sec. II-B.

Giesen *et al.* [21] introduced the *Sclow plot* for identifying and visualizing empty spaces. They use flow lines to depict these empty spaces and employ a scatter-plot matrix to provide a comprehensive view of the high-D dataset along with the flow lines. A downside of this approach, however, is that the flow lines become cluttered and difficult to track and explain at large scales and the scatter-plot matrix view suffers from quadratic growth with increasing data dimensionality, which further complicates the visualization. Also, Sclow plots focus on detecting data distribution features, while our work concentrates on applying empty-space points for optimal configuration search and multi-objective optimization.

We separate the remaining related research into three areas: (1) high-D visualization, (2) identification of empty space via computational geometry, and (3) optimal configuration search.

A. High Dimensional Space Visualization

Understanding empty spaces within high-D environments relies on effective visualization. While various dimension-reduction methods have been devised to project high-D datasets onto a 2D screen, it is important to note that dimension reduction capitalizes on the existence of empty spaces, deflating them to pack the existing points as efficiently as possible in the 2D display. Therefore it is best to conduct empty-space analysis in the native high-D space, with only redundant dimensions removed.

Prominent linear techniques include Principal Component Analysis (PCA) [24], Linear Discriminant Analysis (LDA) [15], and classical Multidimensional Scaling (MDS) [37], all of which project data to a lower-dimensional space while preserving global structure and minimizing distortion. However, these methods often struggle to capture the complexities of data manifolds. On the other hand, manifold-learning techniques such as Locally Linear Embedding (LLE) [44], Uniform Manifold Approximation and Projection (UMAP) [36], and t-distributed Stochastic Neighbor Embedding (tSNE) [50] excel at revealing nonlinear relationships and intrinsic data geometries but the embedding process loses the context of the attributes. The Data Context Map (DCM) [13] offers a unified view for visualizing both variables and data items.

Parallel-Coordinate Plots (PCPs) [26] directly explore the original data space, thus avoiding potential information loss and distortion. PCPs arrange dimensions linearly, aiding in uncovering data relationships and patterns. But PCPs are not without challenges, which has inspired efforts to reduce visual clutter [17], enhance subset tracing and correlation through bundle representation [41], employ graphical abstraction [35], and introduce a many-to-many axis format to reveal deeper variable relationships [32]. We combine PCPs with PCA to visualize the essential attributes of high-D datasets. This integrated approach synergizes the strengths of both techniques, providing a comprehensive, user-centric exploration of high-D spaces in a cohesive and interactive manner.

Subspace analysis is yet another technique for dimension reduction [29]. It decomposes the high-D data space into a set of lower-D subspaces. This can help reveal data patterns obscured by irrelevant (*e.g.*, noisy) data dimensions [20]. A challenge here is the combinatorial explosion in the set of possible subspaces, and this has been the subject of extensive research [34], [49], [53], [54]. Our method could readily incorporate these techniques, and we plan to study this in future work.

B. Empty-Space Identification in Computational Geometry

One way to identify empty spaces is through a geometrical perspective. Computational-geometry techniques offer a complete partition of the data space based on the dataset, and this can be used to locate empty regions accurately among points. These techniques use Delaunay Triangulation, Voronoi Diagrams, and Convex Hulls, all of which are interconnected: the d-dimensional Delaunay Triangulation corresponds to the $(d+1)$-dimensional convex hull; Delaunay Triangulation and Voronoi Diagrams are in duality, *i.e.*, the circumscribed circle centers of Delaunay triangles serve as vertices of Voronoi Diagrams [16].

We initially studied these techniques with a specific focus on Delaunay Triangulation. Each circumscribed circle of a Delaunay triangle describes an empty space, facilitating a comprehensive exploration. However, the downside is its exponential time and space complexity, both $O(n^{\lceil d/2 \rceil})$ [47]. Efforts to mitigate time complexity constraints have explored parallel-computing acceleration, which has shown promise in 2D and 3D spaces [5], [40]. However, extending these methods to higher dimensions is non-trivial and this limits their utility in analyzing empty spaces

within multivariate datasets. Appendix D presents empirical studies we conducted that reveals these shortcomings.

Other solutions include approximation methods. Peled *et al.* [23] proposed a Voronoi Diagram replacement with linear space complexity. Balestriero *et al.* [2] introduced Deep Hull, which employs a deep neural network to determine points within or outside the convex hull. Graham *et al.* [22] proposed a method for higher-dimensional convex hull identification, but its unordered points impede the transferability of Delaunay triangulation. Although these methods reduce time or space complexity, they either approximate specific attributes or lack a meaningful order, limiting their applicability.

C. Optimal Configuration Search

A defining goal of ours is optimal configuration search, which involves systematically navigating through various configurations, arrangements, or settings within a system, application, or model to identify those that deliver the best performance based on specific criteria or objectives.

Optimizing an algorithm or machine-learning model's hyperparameters is often complex. Grid search, which equally divides the range of each variable and examines every point within the high-D data space, is intuitive but inefficient. Alternative strategies like Iterated Local Search (ILS) [33] and its enhancements, such as ParamILS and Focused ILS [25], use hill climbing to iteratively adjust and improve the current solution. Most recent methods rely on population based optimization to explore the hyperparameters space efficiently [27], [42]. These methods require many evaluations to effectively explore the solution space. Our approach, in contrast, focuses on directly identifying and probing candidate solutions that are most distant from existing ones, aiming to find new data instances outside the current range.

Deep neural networks (DNNs) have also been employed for optimal configuration search, serving as function approximators to expedite configuration optimization [12], [51]. While powerful, DNN-based approaches require large datasets for training to ensure reliable results, which can be challenging when working with limited data.

III. Overview

Fig. 1 presents an overview of our methodology. It comprises three phases centered around training an assistive deep neural network (DNN) that becomes increasingly adept at recommending useful empty-space configurations (ESCs) for the target (domain) application's possibly multiple objectives. The process begins with a fully untrained DNN, where the user (typically a domain expert) and our empty-space search algorithm (ESA) collaborate to identify initial "raw" ESC candidates (depicted in Fig. 1 (a)). These candidates are then interactively refined by the user using our GapMiner visual tool and applied within the target application (bottom arrows in Fig. 1 (a)). The application is run with these configurations and their performance is measured. Once verified, these ESCs augment the dataset and are used to train the DNN (middle arrow in Fig. 1 (a)) . As the DNN improves, the workflow reaches the developed phase1. Although ESA and GapMiner are still necessary in this phase, the DNN can provide initial performance estimates for the user to take into account when refining ESCs (arrows on the right-hand side of Fig. 1 (b)).

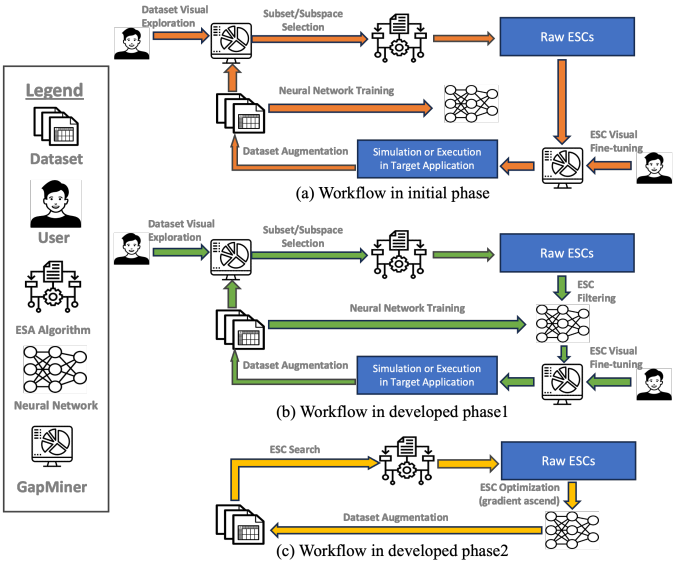

Fig. 1. Our three-phase workflow. Each phase is highlighted with a unique color. The workflows in the initial phase (a) and the developed phase1 (b) are largely similar. However, in the developed phase1, the neural network (DNN) is more advanced and assists in filtering the raw ESCs (c) shows the fully autonomous phase enabled by a fully trained DNN.

Ultimately, the pipeline operates autonomously: the ESA identifies ESC candidates, the DNN optimizes and approximates the target application's outcomes, and the dataset is continuously updated with new ESCs (depicted in Fig. 1 (c)). In the following, we describe the various components of this workflow.

IV. Empty-Space Search Algorithm

The search for empty regions in a high-D data space is a main premise in our work. The challenge here is to describe this space effectively, without enumerating all of the points that reside within each continuous (empty) region. A key requirement is that interior points within an empty space should be far from known points. To locate the emptiest region within a group of data points, one could use Delaunay Triangulation, where the circumscribed center represents the emptiest point. However, as mentioned, computational-geometry methods face the curse of dimensionality.

We instead propose an agent-based approach. Here, the agent is repelled from known data points if it comes too close and is attracted back if it strays too far. This premise is the aim of a physics-inspired function called the *Lennard-Jones Potential*, which is the principle guiding our heuristic search algorithm. A particular advantage of this scheme is that it is easily parallelizable. Agents can be deployed throughout the high-D data space and operate autonomously to identify empty-space points. Next, we describe this physics-inspired search method in detail.

A. Physics Background: The Lennard-Jones Potential

There are multiple causes of intermolecular interactions, including Van der Waals forces, hydrogen bonds, and electrostatic interactions. Some of the forces are repulsive and others are attractive, so that a dynamic equilibrium of molecules is maintained. The Lennard-Jones (L-J) Potential [30] (see Fig. 2 for an example) is an efficient model of intermolecular interactions. Although many

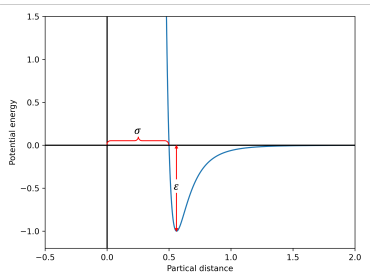

Fig. 2. An example of Lennard-Jones Potential. The x axis is the distance between particles (r) and the y axis is the outcome $V(r)$. The potential is positive when the particle distance is less than σ, indicating a repulsive force, and negative for larger distances, where there is an attractive force. The minimum potential (strongest attraction) is $-\epsilon$.

modern models effectively capture complex physical phenomena, we simplify the empty-space search by simulating agents that interact only with neighboring data, ignoring distant points or other empty-space agents. This led us to choose the L-J potential for its simplicity and efficiency. It is described as:

$$V(r)=4\epsilon\left[(\frac{\sigma}{r})^{12}-(\frac{\sigma}{r})^{6}\right] \tag{1}$$

where r and $V(r)$ are the distance and potential of a pair of particles, and ϵ represents the depth of the potential well, correlating to the strength of the interaction between two particles. σ represents the effective diameter of the particles, that is, the distance at which the total potential energy between two particles becomes zero. At distances less than σ, the repulsive force dominates, causing the potential energy to increase sharply. At distances greater than σ, the attractive force is stronger, pulling the particles together, but this force diminishes with r. The nature of intermolecular interactions keeps a point dynamically steady in a local region. Our search algorithm uses the L-J Potential function.

B. Our Lennard-Jones Potential Based Search Algorithm

We assume that there is a possibly small initial dataset with verified configurations, and we populate the high-D search space with these known configurations. Then we place an agent in this space to search for Empty-Space Configurations (ESC)—the raw ESCs in Fig. 1. The agent starts from a randomly sampled initial position and uses the L-J Potential function to move to a location where the potential becomes zero. This search trajectory (we call "trajectory" for short) is sampled along the way to form a set of raw ESCs.

The vector intermolecular force $F(r)$ driving the agent is:

$$F(r)=\frac{dV(r)}{dr}=24\frac{\epsilon}{\sigma}\left[2\left(\frac{\sigma}{r}\right)^{13}-\left(\frac{\sigma}{r}\right)^{7}\right] \tag{2}$$

To adhere to physical reality, we use the resultant force $\Sigma\vec{F}$ to determine the direction of motion $\vec{d}$, rather than simply using $V(r)$. However, we will stop the agent if $||\Sigma\vec{F}||$ falls below a threshold as this indicates it has moved beyond the data manifold. $\Sigma\vec{F}$ is the sum of forces due to the agent's k nearest neighbors in the dataset:

$$\Sigma\vec{F}=\sum_{i}^{k}\vec{u_i}F(r_i) \qquad \vec{d}=\frac{\Sigma\vec{F}}{||\Sigma\vec{F}||} \tag{3}$$

where k is the number of neighbors and $\vec{u_i}$ is the unit vector from the agent to the ith neighbor.

Algorithm 1: Empty-Space Search for a Single Agent

Set the number of neighbors k, the particle effective diameter σ, the number of search steps n, the step size α, the discount factor γ, the vanishing threshold δ and the rollout interval j;
Initialize a search trajectory $\tau=[]$ and an agent $\pi=\mathbf{c}$ where $\mathbf{c}$ is a random coordinate;
set cumulative magnitude $L=0$, momentum $\vec{m}=\vec{0}$;
Specify constraints on agent:
$f_1(\pi)<=0$; $f_2(\pi)<=0$; …, $f_p(\pi)<=0$;
for $i \ldots n$ **do**
  if i % j == *0* **then**
    Add current coordinate of π to τ;
  end
  Find the k nearest neighbors of the agent from the dataset;
  Use Eq. 3 to calculate $||\Sigma\vec{F}||$ and $\vec{d}$;
  if $||\Sigma\vec{F}||<\delta$ **then**
    return τ;
  end
  $\vec{d}=(\vec{d}*||\Sigma\vec{F}||+\vec{m}*L)/(L+||\Sigma\vec{F}||)$;
  $\pi=\pi+\vec{d}*\alpha$;
  $L=\gamma*L+||\Sigma\vec{F}||$;
  $\vec{m}=\gamma*\vec{m}+\vec{d}$;
  $\vec{m}=\vec{m}/||\vec{m}||$;
  if π *violates any constraint* f **then**
    return τ;
  end
end
return τ;

Alg. 1 illustrates the details of our empty-space search (see Appendix C for a detailed explanation of the parameters and their empirical values). The time complexity of ESA is $O(dknp+np\log N)$ where d is the dimensionality of the dataset, k is the number of nearest neighbors, n is the number of search steps, p is the number of agents and N is the size of the dataset. $O(dknp)$ represents the time complexity to determine the next step, while $O(np\log N)$ represents the time complexity to query k nearest neighbors; the space complexity of ESA is $O(dp)$. In contrast, the time and space complexity of Delaunay Triangulation is $O(N^{\lceil d/2\rceil})$. ESA is considerably more efficient in both time and space. Moreover, it scales well to higher dimensions and larger datasets since it works in local space with simple calculations. As mentioned, since each agent operates independently, it is feasible to expedite the empty-space search using a GPU and deploy a large batch of agents concurrently. Delaunay Triangulation does not offer this kind of parallelism.

C. Incorporating a DNN into the Empty-Space Search

ESA returns the trajectory τ of an agent, where each element is a raw ESC. Since the performance of raw ESCs is unknown and verification is typically costly, we introduce a DNN to predict performance. However, the DNN requires sufficient data to function effectively, so we update the dataset while training it. Users can set two accuracy levels for the DNN, defined by the DNN's acceptable prediction error, which is measured by the loss function. When the DNN performs worse than the lower accuracy level, which occurs

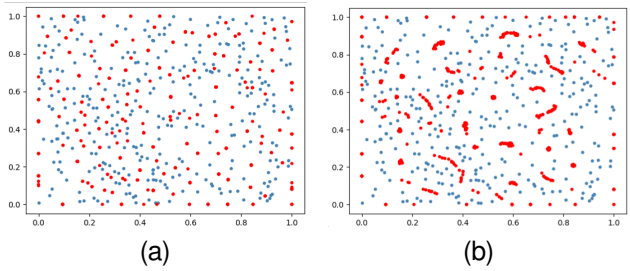

Fig. 3. Given 300 random samples (blue) and 600 random agents (red) in 2D space, we show the results of ESA (a) without and (b) with momentum.

in the initial learning phase, it cannot reliably estimate the value of an ESC. In this case we set $\gamma = 0$, causing the agent π to simply move in the direction of $\vec{d}$; when converged it returns the end of τ as the ESC. This strategy encourages the algorithm to identify as many gaps as possible, fostering the development of a robust neural network and accelerating training. However, it may also lead to convergence on local minima.

Once the DNN has developed further (phase 2 in the workflow of Fig. 1) and meets the lower accuracy level, we begin incorporating a momentum $\vec{m}$ with factor $\gamma > 0$ to move the agent along. This mechanism considers historical directions alongside those calculated by Eq. 3, resulting in a smoother update for the direction of π. We constrain $\gamma < 1$ to prevent a long-term effect; thus, the effect of historical forces gets lower and lower with each search step. The momentum mechanism encourages the agent π to explore the space in a broader range before converging to the empty region, leading it to discover global minima. Then, upon terminating it returns τ as a list of ESCs. Eventually the DNN is accurate enough to achieve the higher accuracy level. We then improve τ by DNN-enabled gradient ascent and pick the best result.

Fig. 3 shows our algorithm's outcomes for a 2D dataset. Agents tend to converge to a small region without momentum (Fig. 3(a)), but to a larger region with momentum (Fig. 3(b)). Larger regions are explored more thoroughly in the latter case while smaller gaps are left alone.

Beyond the in-distribution search shown in Fig. 3, we demonstrate in Appendix F that ESA can also explore out-of-distribution regions, making it valuable for discovering novel configurations beyond the dataset's search boundaries.

V. GapMiner

Our visual analytics system, GapMiner, combines data space exploration and HITL to aid users in identifying promising ESCs during the first two phases of the workflow when the DNN is not yet fully trained.

A. Design Goals

Following the design model devised by Munzner [39], we studied popular datasets on Kaggle, reviewed recent research in our example domain (computer systems [52]), and interviewed several domain experts. This study yielded the essential features an effective visual analytics system for empty-space search should have, formulated as five designs goals (DG):

- **DG1 : Subset/Subspace selection.** Users often focus on specific data segments; *e.g.*, a system designer with a limited budget might want to explore moderate options first. Sometimes these subsets are so small that their key features are hidden within the entire dataset. Hence, explorations within data subspaces must be supported.
- **DG2 : Data distribution visualization and cluster highlighting.** For data with categorical outcomes, each label typically forms a distinct cluster. For continuous outcomes, high and low values usually don't overlap. Highlighting clusters and visualizing distributions will help users identify valuable empty spaces for exploration.
- **DG3 : HITL ESC fine-tuning.** While the ESA identifies numerous ESCs, they are not fully refined configurations, but only starting points for exploration. Users should be able to define empty-space search regions and refine raw ESCs based on their expertise.
- **DG4 : ESC neighbor visualization.** In order to select promising ESCs, users need to understand the distribution of their neighbors, *i.e.*, the shape of the empty space (hypersphere, paraboloid, hyperbolas, etc.), to aid informed decision making.
- **DG5 : Outcome evaluation.** Users must be able to evaluate the performance of an ESC which sometimes is gauged by more than a single outcome variable. Additionally, the cost of obtaining certain configurations can also be important.

B. Terms, Color Semantics, and Representations

Fig. 4 shows the GapMiner interface, visualizing a dataset related to computer system optimization . The dataset consists of multi-tier cache configurations, each with three devices: $L1$, $L2$, and $L3$. Each cache device has three variables: $size$ and *average read/write latencies* (the size of the backend storage device $L3$ is fixed for all configurations and so it is not included). The target variables are *average throughput* and *total purchase cost*, used to evaluate the performance and cost of each configuration respectively. This dataset serves as an example case in Sec. VII and VIII.

The term *existing configurations* used in Fig. 4 (A) refers to configurations in the dataset that have been verified, whereas *proposed configurations* are configurations proposed by the various available ESC search algorithms but have not been verified yet. GapMiner distinguishes these two types with different colors. All gray points and lines in the interface represent proposed configurations, while everything related to existing configurations is colored blue, including density contours in Fig. 4 (B), histogram bars in Fig. 4 (A) and Fig. 4 (C), and scatter points in Fig. 4 (B). Additionally, we apply an orange-to-red colormap in Fig. 4 (D) to encode target variables in the Pareto front of existing configurations, and the same colormap in Fig. 4 (C) to encode target variables of existing neighbors around the proposed one. To clarify the color coding, we added annotations in Fig. 4 to indicate the marks for proposed and existing configurations.

Zhang et al. [56] illustrated the benefits of contour maps in exemplar identification while Li et al. [31] combined density contours and scattered points for outlier examination. In the PCA map in Fig. 4 (B), we follow their design pattern, employing density contours to abstract existing configurations and scattered points to denote proposed configurations. In the PCP, we visualize proposed configurations as dashed lines and existing configurations as solid lines shown in Fig. 4 (C).

Next we describe all of GapMiner's components in detail, referring to the Interface in Fig. 4 and the Design Goals they satisfy.

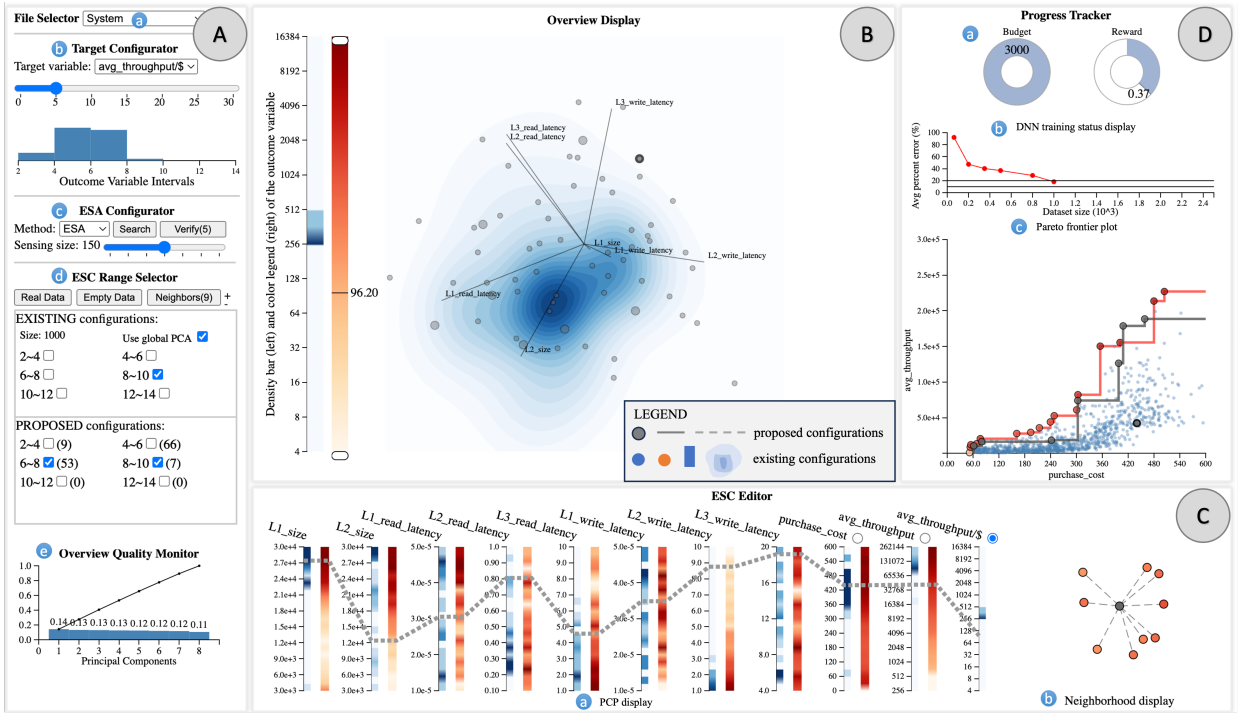

Fig. 4. GapMiner visual interface where a selected ESC is reflected in all displays. (A) Control Panel. From top to bottom: (a) File Selector to load a dataset of initial verified configurations with values for all parameter variables. (b) Target Variable Configurator with an interface for breaking its value range into discrete intervals. (c) Empty-space Search Algorithm (ESA) Configurator to select the ESA and a slider to set the ESC batch size. (d) Empty-Space Configuration (ESC) Range Selector to control which target variable intervals are used for display and ESC proposals. (e) Overview Quality Monitor screeplot that shows the amount of data variance captured by the Overview (PCA) Display. (B) Overview (PCA) Display with data distribution contours, raw or modified ESCs rendered as points, and color legend. (C) Empty-space Configuration (ESC) Editor. From left to right: (a) Parallel Coordinate Plot Display where users can configure ESCs starting from a raw ESC or an existing configuration. (b) Neighbor Display of the selected ESC providing a local view of the distribution of its nearest existing configurations. (D) Progress Tracker. From top to bottom: (a) Budget/Reward Display that captures the aggregated evaluation cost and merit of the ESC exploration so far. (b) Training Status Display of the assistive DNN. (c) Pareto Frontier plot that shows the Pareto frontiers of existing configurations (red) and ESCs (gray) with respect to two user-chosen merit (target) variables.

C. Control Panel

After loading the dataset containing the initial verified configurations with values for all variables, the user will head to the Target Configurator (Fig. 4 (A)) to select a target variable subject to optimization. The target variable can either be a native outcome variable such as *performance* or *cost*, or it can be a ratio like *performance/cost*. The latter provides quick insights into efficiency, while the former two can be optimized within our multi-objective optimization to account for the user's preferences. The user can now utilize the slider below to evenly split the range (**DG1**) of the selected target variable. This action divides the dataset according to the selected value interval and the histogram below visualizes the distribution (**DG2**). The user can now choose among one of three empty-space search methods from the dropdown menu: the physics-based ESA described above, random sampling, Pareto improvement, and a baseline (see Sec. VI).

The ESC Range Selector enables users to select one or more subsets of the data, as specified in the Target Configurator (see above), using the checkboxes in the "existing"/"proposed" group (**DG1**). The "Size" in the verified "existing" group indicates the number of these configurations.

D. Overview Display

The PCA-based Overview Display (Fig. 4 (B), called "PCA map" for convenience) linearly transforms the high-D data space into a variance-maximizing 2D projection. It is an intuitive way to visualize data distributions and identify clusters (**DG2**), We chose the linear dimension reduction provided by PCA since evaluating results from nonlinear methods like tSNE is challenging and connecting their embedding space to the original space is difficult.

Once a checkbox is selected in the ESC Range Selector, the associated subset of the data will be displayed in the Overview Display either as density contours ("existing") or as scattered points ("proposed" or selected via other means). The scented color legend bars on the left map values to color: the left one represents the target variable density of the chosen subset, while the right one shows the value range of the entire dataset. The "Use global PCA" checkbox in the ESC Range Selector (Fig. 4 (A)) determines whether the PCA layout is applied to the chosen data subset (**DG2**) or the entire dataset; the PCA map will update accordingly. Applying the PCA layout to the currently selected data subset will reduce the layout loss and make the display more accurate and focused. This improvement is quantified in the Overview Monitor scree plot.

The loading vectors in the PCA display represent the projection of Cartesian axes from the original space, forming a biplot. Due to the linear nature of PCA, any value change along one axis in the original high-D space results in a proportional update along the corresponding loading vector in the PCA map. Additionally, the loading vector projection is translation-invariant; any point can be chosen as an anchor to project the Cartesian basis. We derive this mechanism in closed form in Appendix A.

E. Empty-Space Configuration (ESC) Editor

This interface panel has two displays: a PCP (left) where an ESC can be configured and refined and a neighborhood display (right) that shows the topology of the ESC's immediate point neighborhood.

PCP display. The PCP allows users to fine-tune a proposed ESC or propose one on their own. It is the most effective display for this task as it provides simultaneous access to all variables and allows for easy adjustments through simple mouse interactions. For additional insight we provide two scented bars along each PCP axis (**DG1, DG2**).

The (blue) density bar next to each axis shows that variable's value distribution in the currently chosen data subset, with darker blue indicating higher density in that range. This visual information complements the density contours in the Overview Display (Fig. 4 (B)). The correlation bar next to the density bar shows the relationship between the variable and the selected target variable, calculated from all existing configurations. Users can select the target variable for this calculation by clicking the radio button along its corresponding axis (*e.g.*, in (Fig. 4 (C)) there are three such variables—two native variables and one ratio variable). For a linear relationship, the orange-red bar along each variable axis indicates whether the correlation with the selected target is positive or negative. For a nonlinear relationship, it reveals which value interval corresponds to a higher target value (darker color) and which to a lower target value (lighter orange).

Lastly, when users modify an ESC in the PCP, its 2D position in the Overview Display is updated proportionally along the direction of the corresponding loading vector. This is particularly useful when there is an empty space in the Overview Display, providing users with important feedback that they are on the right track.

Neighborhood Display. This display, located to the right of the PCP, helps users understand the shape of an empty space as well as the neighbor distribution of the associated ESC (**DG4**). Castermans *et al.* [9] proposed SolarView, which embeds neighboring entities from a high-D space into a 2D radial layout around a central entity. However, their method preserves only the pairwise Euclidean distances, neglecting the topological structure of neighboring entities in the original space and so leading to significant distortion in the shape and integrity of local empty spaces. To preserve the topology structure and visualize the empty space and point distribution around an agent, we devised a dedicated dimension-reduction method we call *cos-MDS*, using a layout optimization scheme similar but not identical to MDS.

Our method embeds the agent at the center and places its immediate high-D neighbors around it. The distance from each neighbor to the agent reflects their true distance in the high-dimensional space, while the pairwise cosine distance between neighbors, as measured from the agent, approximates their true cosine distance in the high-dimensional space. The mapping flattens an N-dimensional hub-and-spoke arrangement into a 2D space, where the spokes vary in both the pairwise angles and in length, rather than being uniformly distributed. It effectively reveals the distances and distribution of neighbors around the agent, as well as the shape and topology of the empty space (fully enclosed, semi-enclosed, cluster boundaries, or anti-hub outliers).

To construct the *cos-MDS* display, we normalize each agent-neighbor vector to the unit hypersphere, compute a pairwise cosine-distance matrix, perform eigenvalue decomposition, and use the top two eigenvalues and their eigenvectors for the low-dimensional embedding. Finally, we scale the low-D agent-neighbor vector to the true length in the original space.

The example shown in the ESC Editor (Fig. 4 (C)) has neighbors uniformly surrounding the configuration, and the distances to it are almost equal. In essence, this is the special case where the empty space is fully enclosed—a pocket in a high-D point cloud. Users can also change the number of neighbors by clicking the +/- button next to the *Neighbors* button in the control panel. This provides a more comprehensive perspective to understand the empty space *e.g.*, in a hypersphere, within a paraboloid, or between hyperbolas. Detailed examples can be found in Appendix B.

F. Progress Tracker

The Progress Tracker (Fig. 4 (D)) keeps the user abreast of the achievements made so far (**DG5**). It has three main components (top to bottom): (a) a Budget/Reward Display that captures the aggregated evaluation cost and merit of the ESC exploration so far, (b) a Training Status Display of the assistive DNN, and (c) a Pareto Frontier Plot that shows the Pareto frontiers of existing configurations (red) and ESCs (gray) with respect to two user-chosen merit (target) variables. We now present these three displays in reverse order according to their use in practice.

Pareto Frontier Plot. The Pareto frontier is an effective mechanism for evaluating a set of proposed configurations in the presence of multiple target variables. To reduce visual complexity we currently restrict the number of target variables to two. Our goal is to aid users in recognizing trade-offs among the two target variables and identify ESCs that can expand the frontier (push the envelope) at desirable trade-off points on the curve. The choice of target variables depends on the application scenario and can include raw variables (*i.e.*, configuration attributes) or aggregated evaluations (*e.g.*, configuration performance). In the computer system case shown in Fig. 4, the goal is to identify configurations that achieve higher average throughput at a lower cost, so the target variables are average throughput and purchase cost.

We use the following color scheme in this plot: (1) existing configurations are colored blue, with their Pareto front connected by edges. The configurations defining these edges (and the edges themselves) are colored by their respective target variable values according to the red-toned color map to the left of the Overview Display and are represented as larger nodes there; (2) proposed configurations are colored gray, with their Pareto front connected by gray lines. Comparing the two curves facilitates an understanding of a candidate ESC's merits in the context what is known already and what the user's trade-off preferences are.

When the user modifies an ESC (say, in the PCP) its position in the grey Pareto plot and frontier (and Overview Display) will update accordingly. Then, upon verification, the ESC is added to the "existing" set, its Pareto point is colored blue, and the red Pareto front updates.

DNN Training Status Display. As mentioned, once the user loads the initial dataset, GapMiner trains an assistive DNN on the backend for ESC performance prediction. The DNN evaluates the performance of each proposed configuration before verification takes place. The testing error of the DNN is then shown in the DNN Training Status Display. We observe in (Fig. 4 (D)) that there is a

fairly large error in the initial stage (first part of the curve), hence at that stage user involvement is typically necessary to identify optimal configurations. The DNN is retrained whenever new configurations are added to the dataset and are verified.

The two horizontal lines in the display signify the user-specified DNN error tolerance. The DNN can be used to filter out poor configurations once it reaches below the first line, and it can be used for configuration optimization once it reaches below the second line. Details of the progressive search pipeline are introduced in Sec. VI.

Budget/Reward Display. The Pareto plot provides an intuitive way to visualize the progress of multi-objective optimization, but it does not offer a quantitative measure of improvement. To address this limitation, we incorporate the concept of *Pareto dominance area* [8] into the Progress Tracker (Fig. 4 (D)) to monitor the dataset's Pareto front. The Pareto dominance area is a well-established quality metric in multi-objective optimization that quantifies the volume of the objective space dominated by the Pareto front. A larger Pareto dominance area indicates a better Pareto front. To encourage users to maximize this area during the empty-space configuration search, we use it to quantify the *reward*.

On the other hand, each ESC verification incurs a cost, which can be significant. The aggregated verification cost represents the expense of running a selected configuration in the real world. It is a broad concept that can correspond to price, energy, time, effort, calories, etc. For example, in the computer system case shown in Fig. 4, users had to purchase two tiers of cache and a backend storage specified by the configuration , and run them for a week to obtain the real average throughput. In our application, thanks to CloudPhysics [52], we could quickly simulate a configuration to get an approximate result. However, verification costs are unavoidable in many scenarios. To address this, we provide a budget counter in the Progress Tracker (Fig. 4 (D)) to remind users of the aggregate cost of continued ESC verification. The cost donut chart will remain inactive if no cost metric is defined.

We chose donut charts for both the Pareto Dominance-based Reward Display and the Budget Display due to their compactness and their ability to effectively convey proportional data at a glance.

VI. DNN-Assisted Configuration Search Pipeline

While GapMiner effectively assists users in the search for optimal ESCs, manually identifying a large number of optimal ESCs can be tedious. To address this challenge, we propose a pipeline that trains an assistive DNN to eventually automate this process, positioning the analyst as a mentor to the DNN. This section provides a breakdown of the pipeline, as illustrated in Fig. 1. We begin by describing the various search methods we have implemented and then describe their role in the DNN-training process.

A. Configuration Search Methods

In addition to the physics-inspired ESA, GapMiner integrates three more methods for investigating the empty space: random sampling, Pareto improvement, and a blank baseline. The blank baseline is a configuration with every variable set to 0.5, locating it at $[0,0]$ in the PCA map. This strategy provides no hint from the initial position, requiring users to rely solely on GapMiner interactions to optimize a configuration.

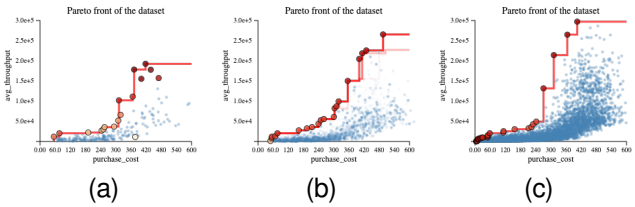

Fig. 5. An example Pareto front at the three key stages. (a) The initial stage when the DNN has just started training. There are only few configurations in the "existing" set, colored blue. (b) The front when the DNN has achieved t_1. The "existing" set has grown. (c) The front when the DNN has achieved t_2. The "existing" set now covers much of the front's interior.

Random sampling, as the name suggests, samples configurations in a random manner. Pareto improvement, on the other hand, starts from the Pareto front of the existing set, allowing users to improve a configuration from a Pareto-optimal point. While Pareto improvement can be a good starting point for breakthroughs, it does not provide insights into the empty space. Empty-space search and random sampling are likely to find configurations better than those based on the Pareto front. Initially, we used another strategy, random walk, but experiments showed it was statistically similar to random sampling, so we did not include it in GapMiner.

All of these methods, including ESA, work as initial configuration hints for users in the startup stage. Users can then fine-tune the configurations proposed by these strategies to get better configurations. Additionally, users can also let the algorithm propose and verify a batch of configurations. The batch size is determined by the slider in the ESA Configurator in Fig. 4(A). Once new configurations have been verified, they will be added to the "existing" configuration set and used to retrain the DNN.

B. The Assistive DNN Model

Our default DNN is a MLP (Multi-Layer Perceptron) model with 3 hidden layers of 256 neurons each, suitable for general scenarios, and predefined loss functions, including mean squared error (MSE), mean absolute percentage error (MAPE), and cross entropy (CE). Users can also load custom neural networks and loss functions into the pipeline, as long as they are implemented in TensorFlow/PyTorch. The training loss and selected loss function are displayed in the DNN Training Status Display. Any DNN compatible with regression or classification tasks on tabular datasets can be used in our system.

C. Pipeline Throughout the DNN Training

Fig. 5 shows the Pareto front development at the three pipeline stages. We now describe these in more detail.

Initial Stage. In the early stage, the dataset is still small and its empty-space regions remain largely unexplored. With the data set still being insufficient to train a usable neural network, users can utilize GapMiner and its various exploration tools to identify empty spaces, propose new configurations, optimize them, and verify their effectiveness. In our studies with analysts, we found that combining various configuration search strategies yields more diverse and higher-quality results to grow the dataset than sticking with only one search algorithm. Additionally, the visual feedback provided by the Progress Tracker incorporates elements of gamification. It not

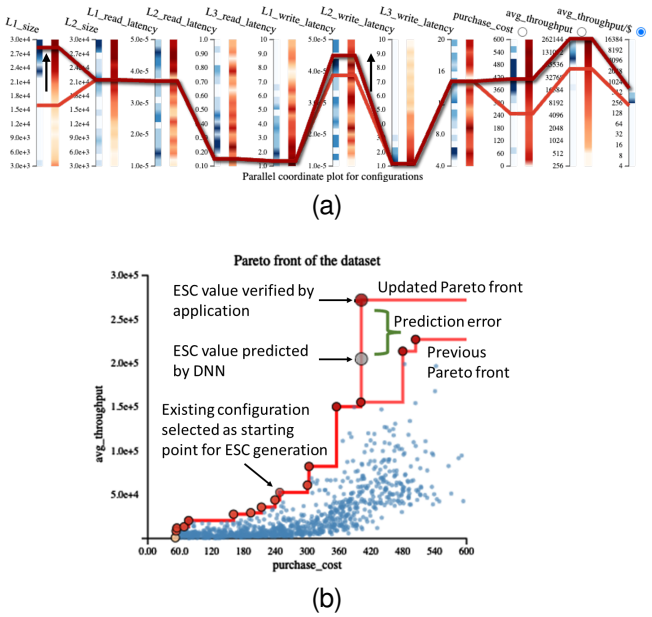

Fig. 6. Improvement of the Pareto front by interaction in the PCP followed by DNN estimation of the target values and subsequent verification in the domain application which revealed a lack of DNN training. (a) PCP showing the result of some user interactions; (b) annotated Pareto front.

only indicates progress but also challenges users to find high-quality ESCs in a cost-effective manner.

Developed Stages. As the set of existing configurations grows, paying attention to the error plot in Fig. 4 (D) becomes crucial. As mentioned, it shows the average percentage error of the DNN along with the two user-set accuracy levels t_1 and t_2. Once the error is below t_1, the system will use the DNN to evaluate the proposed configurations, improving the quality of suggestions. Users can rely more on the system at this stage but still apply their domain knowledge to enhance results. Then, when the error drops below t_2, the system will go one step beyond, using gradient ascent of the DNN to refine configurations further. The performance estimation now becomes accurate enough to find optimal configurations without user supervision. At this stage, the DNN and ESA can completely replace the user.

VII. Application Example

To illustrate our pipeline, we present a use case in the application area of computer system optimization. Fig. 4 shows a snapshot taken of the interface during this session.

We used a trace of a real-world workload run on physical machines, provided by CloudPhysics [52]. We chose trace *w11*, which captures a week of virtual disk activity from a production VMware environment. This trace file contains I/O requests recorded over the week, which can then be replayed using either a physical machine or a simulator of that machine. To save time and energy, we used the simulator for all our experiments. We generated our dataset by replaying workload *w11* on simulated system with two tiers of cache *L1* and *L2* and a backend storage device *L3* (see Sec. V-B for an introduction to the dataset).

Given the size of the workload, evaluating a configuration is an expensive process. Additionally, there are a vast number of potential multi-tier configurations. Thus, conducting an exhaustive search of this space is infeasible, making the efficient identification of optimal configurations highly valuable to system administrators [18], [19].

To begin, we collected a small number of configurations with the help of system experts. We then invited a field expert to try GapMiner, ESA, and the overall pipeline. Empirically, a configuration with good *avg_throughput* is usually more expensive; thus the expert's goal was to find optimal configurations with good performance at a lower price.

The expert began by loading the dataset and setting the two DNN thresholds: $t_1 = 20\%$, $t_2 = 10\%$. Next, he selected $avg\_throughput/\$$ as the target variable since it captures a reasonable first compromise between the two main objectives $avg\_throughput$ and $purchase\_cost$. After bracketing the target variable he learned from the range histogram (Fig. 4 (A)) that there were just a few configurations in the best and worst value intervals. Next the expert ran the ESA algorithm and then checked the (higher) 8˜10 interval of the "existing" group and the 6˜8 and 8˜10 groups from the "proposed" configurations found by the ESA (see (Fig. 4 (A)). This action revealed the former as a density plot and the latter as scattered points in the Overview Display (Fig. 4 (B)). A Pareto front, computed from the "proposed" configurations, appeared in the Pareto front display (Fig. 4 (D)) in grey, with the points defining the front being drawn as larger nodes in the Overview Display.

The red curve in this Pareto front display shows the front of the "existing", verified configurations, and the configurations that define it are colored by the selected target variable, $avg\_throughput/\$$, according to the red-toned colormap to the left of the Overview Display. The expert noticed that the grey front only surpassed the red front in the high $purchase\_cost$ area but had poor $avg\_throughput$ for low $purchase\_cost$ configurations. He decided not to evaluate or verify any of the proposed ESCs but rather focused on the red Pareto front in the low $purchase\_cost$ area. He found a lower-performing configuration there, indicated by its orange, less vibrant red color (see Fig. 6(b)). He first tried to improve it by moving it to the dense region in the Overview Display where the high-performing "existing" configurations reside, but this strategy failed. To investigate why, he examined the scree plot in Fig. 4 (A) and noticed the slow growth of the aggregate curve. This essentially means that the PCA map did not explain much of the data variance and so moving configurations along loading vectors that pointed directly to the dense region was not sufficient. Some short or nearly orthogonal loading vectors like $L1\_size$ or $L1\_write\_latency$ might also matter.

Looking for an alternative approach he turned to the density bars in the PCP to learn about the per-variable distributions. Fig. 6(a) shows the PCP with the polyline of the Pareto configuration under consideration colored in orange, indicating its lower performance. And indeed, despite residing in a dense PCA region, this configuration suffered from sub-optimal $L1\_size$ and $L1\_write\_latency$ settings (it falls in less dense and lightly colored regions in the density and correlation bars, respectively, for both variables). These variables minimally contributed to the PCA space but apparently significantly impacted the target variable outcomes.

The expert adjusted the configuration to dense intervals in both variables, as indicated by arrows in Fig. 6(a). He then asked the DNN to estimate the value for $avg\_throughput$ and calculated other target variables, which showed significant improvement and expanded the Pareto front in the upper region (grey node in Fig. 6(b)). Pleased, the expert clicked the *Verify* button to obtain the true performance. After some time, the verification results showed that this configuration was even better than the DNN prediction (red node at the top of

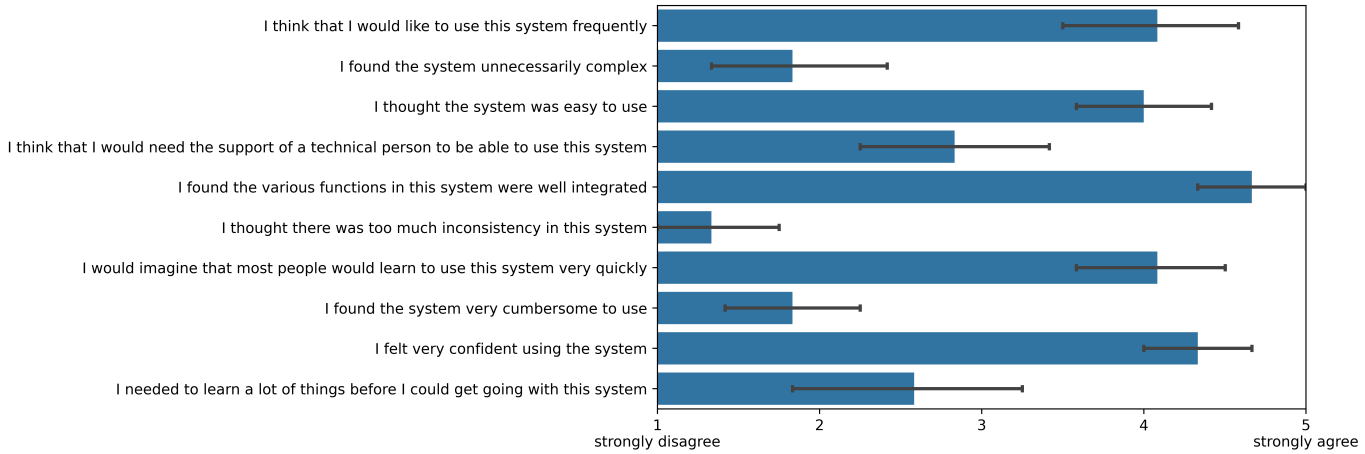

Fig. 7. The SUS scores from 12 users regarding each question. The y-axis lists SUS questions and the x-axis is the scores to each question ranked from 1 (strongly disagree) to 5 (strongly agree).

Fig. 6(b)), and the dominance area increased significantly.

While this was a positive development, the expert also realized that the DNN did not perform well in the upper region of the Pareto front (see the large prediction error in Fig. 6(b)). More configurations in that range were needed to train the DNN effectively. To address this, the PCP interface enables users to select specific value intervals and run ESA exclusively within these intervals to generate ESC batches.

As the expert created more ESC batches and verified some of the generated ESCs, the DNN eventually reached the t_1 threshold. At this point, the expert was able to rely more on the DNN to assess the quality of ESCs and he started to only focus on the ESCs located on the proposed Pareto front for verification. Finally, once the DNN surpassed t_2, the AI model completely took over the expert's role. As shown in Fig. 5, this three-stage pipeline increased the dominance area from 0.27 to an impressive 0.56, more than doubling it.

VIII. User Study

We conducted a user study to evaluate GapMiner with the same system dataset (see Appendix E for a detailed description of this experiment). Across two rounds of experiments, we recruited 17 graduate students with CS background for this user study and specifically assessed the system-related expertise of 7 of them. Among these, 5 had systems expertise, while 2 did not exhibit expertise in that area. The user study task involved performing an optimal-configuration search starting from an initial set of 200 configurations. The users could make full use of GapMiner with all search methods: ESA, random sampling, Pareto improvement, and the blank baseline. The batch size for searching was fixed to 50 for all users. They could click the *search* button multiple times, but they could click the *verify* button only 5 times to get the true $avg\_throughput$. Once they believed they had a good configuration, they ran one of their alloted verifications. We calculated the dominance area and updated it in the reward donut chart in Fig. 4 (D) [8] as the evaluation metric.

A. Result Analysis

We compared the GapMiner-aided performance of the users with the outcomes achieved with (1) ESA-based search only and (2) verifying the initial random samples directly, in order to see whether

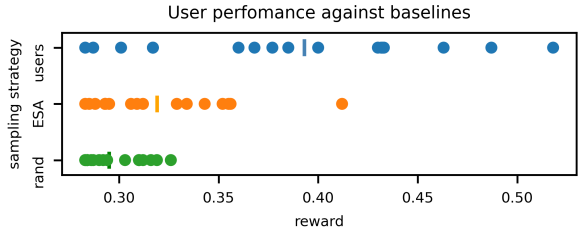

Fig. 8. Rewards achieved by users, ESA, and random sampling. Each dot represents a search round (5 verifications). The reward is defined by dominance area in the Pareto plot. The bar within each strip is the mean value.

GapMiner helped users with the task. Noticing that users clicked *search* three times on average, we set the batch size to 150 for the two baselines (ESA and random) to match the user behavior, calculated the Pareto front estimated by the naïve neural network, and then verified the naïve Pareto front. We ran each baseline 17 times to match the number of participants. We did not choose the top five for verification in the baselines because we found no difference between verifying the top five and the entire Pareto front in the dominance area. Our results are presented in Fig. 8, showing a clear advantage in user performance compared to the two baselines. The average reward for ESA is 0.320, while the average reward for random sampling is 0.298—both significantly lower than the average reward of **0.388** achieved by the users. Notably, **12** users achieved higher rewards than ESA, and **15** outperformed random sampling.

We also did a between-subjects ANOVA with Tukey's Honestly Significant Difference (HSD) to analyze our data's statistical significance; the analysis is shown in Tab. II in Appendix E. Cohen's f from ANOVA was **0.85**, indicating a large effect size. We observe that GapMiner is significantly better than both baselines, while the outcomes between the two baselines do not differ statistically.

B. System Usability

We evaluated the usability of GapMiner with the popular questionnaire System Usability Scale (SUS) [6], which has 10 carefully designed questions that evaluate the system in various aspects. After completing the study, we distributed the SUS questionnaire to all participants and received 12 responses. The responses are summarized in Fig. 7, with detailed information provided in Appendix E. The overall SUS score calculated from

the questionnaire was **76.88**. According the guidelines by Bangor et al. [3] and Sauro et al. [46], our system is ranked B (SUS score $>$ 72.6) and is considered good (SUS score $>$ 71.1). Therefore, we can state that GapMiner has shown good usability in these initial tests.

Additionally, some users provided comments on the system. One user, who completed an M.S. in visualization and is now a Ph.D. student in computer systems, gave very positive feedback on GapMiner. He found that it helped him quickly understand the problem and complete the task efficiently. He also believed that GapMiner could be beneficial for other system analysis projects.

Another user, also a Ph.D. student in computer systems, initially felt that GapMiner provided too much flexibility when exploring a problem. He found the information overwhelming, which made the learning curve steep. However, after becoming familiar with the system, he found it highly effective and noted that it significantly improved his analysis efficiency.

A third user who lacked background in computer systems and visualization, found both the system and the problem challenging to learn, expressing that mastering the system was difficult for users with non-technical backgrounds. She noted that non-experts might need guidance from a technical person to effectively use GapMiner.

This latter observation aligns with our expectations, as GapMiner was developed in collaboration with domain experts rather than for laypersons. Our study confirms that experts can effectively use the tool. Future work will explore ways to enhance onboarding and provide guidance to support users with less technical expertise.

C. Comparison Experiments

In addition to testing user performance against algorithms in the initial stage of the pipeline, we also did two experiments to compare how ESA outperforms random methods in developed stages with the help of a neural network. These experiments did not involve users, but employed the well-trained DNN to evaluate and fine-tune ESCs. In the standard pipeline described in Sec. VI, there is an evolutionary DNN working as a critic to determine promising ESCs, and a black box to collect true values. Based on the advice of field experts, we simplified the pipeline in the block-trace scenario by replacing the critic and the black box with a well-trained DNN. The DNN had an average percentage error of 7.9%, working as a surrogate model to simultaneously determine good ESCs and verify outcome values.

We employed two baselines in this section: random sampling (RS) and random walk (RW). Random sampling simply samples configurations in the data space. Random walk further walks in a random direction for each of 400 steps, which is the same number we used in the ESA.

In the first experiment, we fast-forwarded to a developed stage that had 1,000 configurations in the dataset instead of iterating from an initial stage. In RS, we randomly sampled 1,500 configurations at once. Starting from the same initial positions as used in RS, we ran ESA and RW, and chose the best ones evaluated by the DNN on each trajectory independently. We repeated the process 50 times to reduce result variance. As before, we used the dominance area as the evaluation metric. The average rewards of ESA, RS, and RW were **0.450**, **0.409**, and **0.413**, respectively. ESA was statistically better than RS and RW, and there was no significant difference between RS and RW. The effect size η^2 was **0.3**, indicating that the magnitude of the difference between the averages was **large**. The full statistical analysis is shown in Tab. IV in Appendix E.

In the second experiment, we fast-forwarded to a stage with 3,000 configurations, using the same surrogate DNN as before. We did the same experiments as in the previous one, the difference being that we used gradient ascent to optimize the best configuration for each trajectory in ESA, RS, and RW. The average rewards were **0.453**, **0.418**, and **0.422** respectively. There was a statistically significant difference between ESA and random baselines, but no difference between RS and RW. η^2 was **0.3**, indicating a **large** level of effect size. The full statistical analysis is given in Tab. V in Appendix E.

All of these experiments and statistical analyses clearly show that our empty search algorithm is much better than random methods.

IX. Second Case Study

In addition to the experiments in system research, we applied our work to an entirely different domain, wine investigation [45] to show generalizability to domain-agnostic scenarios.

Different from system verification, the wine dataset was collected from the real world. It has 11 chemical properties and a quality rating associated with each wine instance. The 11 chemical properties include{fixed, volatile, citric} acidity, residual sugar, chlorides,{free, fixed} sulfur dioxide, density, pH, sulphates, and alcohol. The wine quality ranges from 3 to 8.

Imagine there is a winemaker fermenting new wine. With quality being the primary concern in mind, the winemaker also wants to minimize the free sulfur dioxide due to potential health issues. We can help the winemaker with GapMiner. First, the scree plot in GapMiner reveals that the PCA space explains about 60% of the variance, which means that the corresponding region in the original space is less variant given a dense region in the PCA space. Next, the distribution of wine quality—3, 4, 7, and 8 in the PCA space (see Fig. 9(a))—illustrates that alcohol and citric acid point to the high-quality region while volatile acidity and density point to the low. Thus, given a wine instance, increasing alcohol and citric acid while reducing volatile acidity and density is more likely to improve the quality. Moreover, the PCP scented widgets indicate that wine quality is independent from free sulfur dioxide. Thus, wine instances in every quality can be fine-tuned to reduce free sulfur dioxide, fitting a wide range of customers.

Looking at the histogram of outcome variable intervals, we noticed that the dataset is severely imbalanced: almost 85% of the wine instances are ranked quality 5 or 6; only 1% are ranked quality 3 or 8, which results in a low prediction accuracy. Therefore, we had to augment the imbalanced quality groups. To do so, we first referred to the PCP scented widgets to learn the property distributions of each quality. For example, wine in quality 4 has higher volatile acidity but lower citric acidity, while that in quality 7 is the opposite. After finding the distributions, we constrained the search range to a dense interval of each property by brushing on the scented widgets in the PCP. Thus, ESA ran in a specific quality region.

Due to the difficulty of obtaining the quality of new wine instances by ourselves, we made it the same as the nearest existing neighbor. This simple but effective strategy is widely used in imbalanced learning [10]. Having identified over 6,000 ESCs, the DNN accuracy improved from 60% to 80% on quality prediction. On the other hand, free sulfur dioxide was integrated with other properties during the empty-space search. Therefore, its value was determined by ESA, which also reduced the DNN's mean-squared

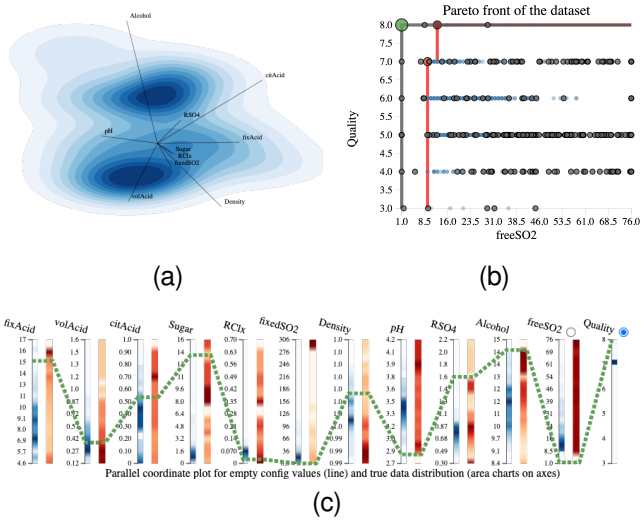

Fig. 9. The key results of GapMiner on the wine dataset: (a) The PCA map showing the distribution of wine instances. The dark region on top represents wine instances with qualities 7 and 8, while the dark region below it corresponds to qualities 3 and 4; (b) ESA results in the Pareto plot. We plot the wine set as blue points and the associated Pareto front as a red line, while ESCs are gray points and the associated Pareto front is a gray line; (c) The full property values of the global optimal wine instance (highlighted in green in (b)).

error from 0.018 to 0.002. During the pipeline execution, we were able to find numerous wines with low free sulfur dioxide in each quality. We show the results in Fig. 9 and Appendix G-B.

To summarize, our ESA, GapMiner, and the pipeline were able to balance the dataset by augmentation, and improve DNN performance. Though we were unable to verify these results in the field, our work could identify numerous innovative wine instances predicted to have good quality or low free sulfur dioxide.

X. Discussion and Future Work

Empty-space exploration for discovering innovative configurations in high-dimensional parameter spaces is a promising yet long-overlooked approach. In this paper, we introduce a novel Empty-Space Search Algorithm (ESA) to identify such configurations. However, because evaluating a proposed empty-space configuration requires domain expertise or a well-trained predictor, we integrate ESA into GapMiner , a newly designed human-in-the-loop (HITL) visual analytics system. This integration is part of a workflow that gradually transitions from HITL to AI-driven search.

Our user study confirms the effectiveness of GapMiner, supported by comparison experiments demonstrating its superiority over random and ESA-only search. Additionally, case studies highlight its potential across various domains. As a follow-up, we plan to explore broader applications, particularly in security fields such as network intrusion detection in cybersecurity [11].

The current initialization strategy of ESA agents is random sampling, which is likely to miss important regions. In future work, we plan to integrate subspace analysis to achieve more efficient agent deployment. Also, our ESA assumes that variables are mutually independent. However, in real-world scenarios, complex causal relationships might exist among the variables. A possible research direction can be to leverage large language models to incorporate causal information into the empty-space search process [28], [57].

Another aspect to improve is the progressive neural network training process. Currently, our empty-space search focuses on optimal configuration search, but it never speeds up neural network training. In future work, we plan to incorporate active learning to identify ESCs that are both significant to network convergence and optimal.

Acknowledgments

This work was partially funded by NSF grant CNS 1900706. We thank Mário Antunes for helpful discussions.

References

[1] O. Alipourfard et al. Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In *USENIX Symposium on Networked Systems Design and Implementation*, pp. 469–482, 2017.
[2] R. Balestriero, Z. Wang, and R. Baraniuk. DeepHull: Fast convex hull approximation in high dimensions. In *ICASSP*, pp. 3888–3892, 2022.
[3] A. Bangor, P. Kortum, and J. Miller. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123, 2009.
[4] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.
[5] G. Blelloch, G. Miller, J. Hardwick, and D. Talmor. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24:243–269, 1999.
[6] J. Brooke. SUS: A 'quick and dirty' usability scale. *Usability Evaluation in Industry*, 189, 11 1995.
[7] S. Brunton et al. Data-driven aerospace engineering: reframing the industry with machine learning. *AIAA Journal*, 59(8):2820–2847, 2021.
[8] Y. Cao, B. Smucker, and T. Robinson. On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design. *Journal of Statistical Planning and Inference*, 160:60–74, 2015.
[9] T. Castermans et al. Solarview: Low distortion radial embedding with a focus. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2969–2982, 2018.
[10] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
[11] J. Chen et al. Real-time network intrusion detection via decision transformers. *arXiv preprint arXiv:2312.07696*, 2023.
[12] J. Chen and Y. Liu. Neural optimization machine: A neural network approach for optimization. *arXiv preprint arXiv:2208.03897*, 2022.
[13] S. Cheng and K. Mueller. The data context map: Fusing data and attributes into a unified display. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):121–130, 2015.
[14] R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, and F. De Felice. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability*, 12(2):492, 2020.
[15] J. Cohen, P. Cohen, S. West, and L. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Routledge, 2013.
[16] M. De Berg. *Computational Geometry: Algorithms and Applications*. Springer Science & Business Media, 2000.
[17] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007.
[18] T. Estro, P. Bhandari, A. Wildani, and E. Zadok. Desperately seeking … optimal multi-tier cache configurations. In *HotStorage*, 2020.
[19] T. Estro et al. Accelerating multi-tier storage cache simulations using knee detection. *Performance Evaluation*, p. 102410, 2024.
[20] T. Fujiwara, Y.-H. Kuo, A. Ynnerman, and K.-L. Ma. Feature Learning for Nonlinear Dimensionality Reduction toward Maximal Extraction of Hidden Patterns . In *IEEE PacificVis*, pp. 122–131, 2023.
[21] J. Giesen, L. Kühne, and P. Lucas. Sclow plots: Visualizing empty space. *Computer Graphics Forum*, 36(3):145–155, 2017.
[22] R. Graham and A. Oberman. Approximate convex hulls: sketching the convex hull using curvature. *arXiv preprint arXiv:1703.01350*, 2017.
[23] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *IEEE Symp. on Foundations of Computer Science*, pp. 94–103, 2001.
[24] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.
[25] F. Hutter, H. H. Hoos, and T. Stützle. Automatic algorithm configuration based on local search. In *AAAI*, vol. 7, pp. 1152–1157, 2007.
[26] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1:69–91, 1985.

[27] L. Japa et al. A population-based hybrid approach for hyperparameter optimization of neural networks. *IEEE Access*, 11:50752–50768, 2023.
[28] E. Kıcıman, R. Ness, A. Sharma, and C. Tan. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*, 2023.
[29] H.-P. Kriegel, P. Kröger, and A. Zimek. Subspace clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):351–364, 2012.
[30] J. Lennard and I. Jones. On the determination of molecular fields.—I. from the variation of the viscosity of a gas with temperature. *Proceedings of the Royal Society of London. Series A*, 106(738):441–462, 1924.
[31] H. Li, S. Jorgensen, J. Holodnak, and A. Wollaber. ScatterUQ: Interactive uncertainty visualizations for multiclass deep learning problems. In *IEEE VIS*, pp. 246–250, 2023.
[32] M. Lind, J. Johansson, and M. Cooper. Many-to-many relational parallel coordinates displays. In *IEEE InfoVis*, pp. 25–31, 2009.
[33] H. Lourenço, O. Martin, and T. Stützle. Iterated local search. In *Handbook of Metaheuristics*, pp. 320–353. Springer, 2003.
[34] S. Mahmood and K. Mueller. Interactive subspace cluster analysis guided by semantic attribute associations. *IEEE Transactions on Visualization and Computer Graphics*, 2023.
[35] K. T. McDonnell and K. Mueller. Illustrative parallel coordinates. *Computer Graphics Forum*, 27(3):1031–1038, 2008.
[36] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
[37] A. Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D*, 41(1):27–39, 1992.
[38] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054, 2023.
[39] T. Munzner. A nested model for visualization design and validation. *IEEE Trans. Visualization and Computer Graphics*, 15(6):921–928, 2009.
[40] C. Nguyen and P. Rhodes. Delaunay triangulation of large-scale datasets using two-level parallelism. *Parallel Computing*, 98:102672, 2020.
[41] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf. An edge-bundling layout for interactive parallel coordinates. In *2014 IEEE Pacific Visualization Symposium*, pp. 57–64. IEEE, 2014.
[42] J. Parker-Holder, V. Nguyen, and S. J. Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In *NeurIPS*, vol. 33, pp. 17200–17211, 2020.
[43] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531, 2010.
[44] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
[45] S. Sammari. red wine data. https://www.kaggle.com/datasets/midouazerty/redwine. Accessed on 2024-02-01.
[46] J. Sauro. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC, 2011.
[47] R. Seidel. The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5(2):115–116, 1995.
[48] O. Strnad et al. Real-time visualization of protein empty space with varying parameters. *Proc, of BIOTECHNO, IARIA XPS Press*, pp. 65–70, 2013.
[49] A. Tatu et al. ClustNails: Visual analysis of subspace clusters. *Tsinghua Science and Technology*, 17(4):419–428, 2012.
[50] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.
[51] G. Villarrubia, J. De Paz, P. Chamoso, and F. De la Prieta. Artificial neural networks used in optimization problems. *Neurocomputing*, 272:10–16, 2018.
[52] C. Waldspurger, N. Park, A. Garthwaite, and I. Ahmad. Efficient MRC construction with SHARDS. In *FAST*, pp. 95–110, 2015.
[53] B. Wang and K. Mueller. The subspace voyager: Exploring high-dimensional data along a continuum of salient 3D subspaces. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1204–1222, 2017.
[54] J. Wang, X. Liu, and H.-W. Shen. High-dimensional data analysis with subspace comparison using matrix visualization. *Information Visualization*, 18(1):94–109, 2019.
[55] S. Zhang, S. Bamakan, Q. Qu, and S. Li. Learning for personalized medicine: a comprehensive review from a deep learning perspective. *IEEE Reviews in Biomedical Engineering*, 12:194–208, 2018.
[56] X. Zhang, S. Cheng, and K. Mueller. Graphical enhancements for effective exemplar identification in contextual data visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2022.
[57] Y. Zhang et al. An explainable ai approach to large language model assisted causal model auditing and development. *arXiv preprint arXiv:2312.16211*, 2023.

Xinyu Zhang earned his B.E. in Computer Science at Shandong University, Taishan College in 2019. Currently, he is a Ph.D. candidate at Stony Brook University. His research interests include multivariate data analysis, scientific visualization, and reinforcement learning.

Tyler Estro is a Ph.D. Candidate in Computer Science at Stony Brook University being advised by Professor Erez Zadok and a member of the File Systems and Storage Lab. Tyler's primary research interests are multi-tier storage caching, disaggregated memory systems, and Compute Express Link (CXL) technology.

Geoff Kuenning is an Emeritus Professor of Computer Science at Harvey Mudd College. His research interests include storage systems, tracing, and performance measurement. He is an Associate Editor of ACM Transactions on Storage, and the maintainer of the widely used SNIA IOTTA Trace Repository.

Erez Zadok is a Professor of Computer Science at Stony Brook University, where he directs the File Systems and Storage Lab (FSL). His research interests include file systems and storage, operating systems, energy efficiency, performance and benchmarking, security and privacy, networking, and applied ML.

Klaus Mueller is a Professor of Computer Science at Stony Brook University and a senior scientist at Brookhaven National Lab. His research interests include explainable AI, visual analytics, data science, and medical imaging. To date, his 300+ papers have been cited over 14,500 times. He is a IEEE Fellow.

APPENDIX A
CONNECTION BETWEEN PCA SPACE AND ORIGINAL SPACE

Given any multivariate configuration $V=[v_1,v_2,...v_N]^T$ in an $N-$dimensional dataset, let the PCA transformation matrix be

$$M=\begin{bmatrix} m_{11} & m_{12} & ... & m_{1N} \\ m_{21} & m_{22} & ... & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & ... & m_{nN} \end{bmatrix}$$

The corresponding position $v=[v_1^*,v_2^*,...,v_n^*]^T$ in $n-$dimensional PCA space is given by $v=MV$. Changing the value of V at the kth variable by δv_k can be written as $\delta V=[0,...\delta v_k,...,0]$, and $V_{new}=V+\delta V$.

Notice that the ith row of M is the ith principal component in the data space, $P_i=[m_{i1},m_{i2},...,m_{iN}]$, and the jth column of M is the loading vector of variable j: $L_j=[m_{1j},m_{2j},...,m_{nj}]^T$. Therefore, the moving step δv in PCA space can be derived as follows:

$$\begin{aligned} v_{new} &= MV_{new} \\ &= M(V+\delta V) \\ &= [P_1,P_2,...,P_n]^T V+[L_1,L_2,...,L_N]\delta V \\ &= v+L_k\delta v_k \\ \Longrightarrow \delta v &= v_{new}-v \\ &= L_k\delta v_k \end{aligned} \tag{4}$$

From Eq. 4 we can conclude that the moving direction in PCA space is determined only by the corresponding loading vector, and the step size is determined by the value difference and the loading vector's magnitude. Thus we can connect the original space in the Parallel Coordinate Plot with the PCA space in the PCA map: changing the value on a PCP axis will lead to a position update in PCA map as determined by Eq. 4.

APPENDIX B
AGENT-CENTERED DIMENSION REDUCTION

A. Examples for Basic Geometric Structures

We demonstrate the application of cos-MDS on three synthetic datasets: a 3D hyperboloid, a 4D paraboloid, and a 4D hypersphere. For each dataset, we approximately placed the agent at the center of the empty space and set the number of neighbors equal to the size of the dataset.

We generated the 3D hyperboloid dataset by uniformly sampling the 30 coordinates from $[-1,1]$ on the x and y axis, respectively. We then calculated z by

$$z=\pm 2*\sqrt{\frac{x^2}{0.04}+\frac{y^2}{0.04}+1}$$

There are 900 points in this dataset and we placed the agent at $(0,0,0)$. The synthetic dataset is directly plotted in Fig. 10(a). The corresponding result of cos-MDS can be found in Fig. 10(b).

The 4D paraboloid dataset was generated by uniformly sampling 10 points from $[-5,5]$ on x, y, and z axis independently, and the points on the paraboloid were defined by (x,y,z,z^2). This dataset includes 1000 points. We set the agent at $(0,0,0,20)$. The cos-MDS result on this dataset can be found in Fig. 10(c).

For the 4D hypersphere dataset, we first uniformly sampled 1,000 angles (ψ,θ,ϕ), where $\psi\in[0,2*\pi]$, $\theta\in[0,\pi]$, and $\phi\in[0,\pi]$. Then we calculated the points (x,y,z,w) on the hypersphere by

$$\begin{aligned} x &= cos(\psi)sin(\theta)sin(\phi) \\ y &= sin(\psi)sin(\theta)sin(\phi) \\ z &= cos(\theta)sin(\phi) \\ w &= cos(\phi) \end{aligned} \tag{5}$$

This synthetic dataset contains 1,000 points as well. Again, we located the agent at (0, 0, 0, 0). The cos-MDS result can be found in Fig. 10(d).

From the results in Fig. 10, we observe that cos-MDS visualizes a 3D hyperboloid (a) as a 2D hyperbola (b), transforms a 4D hypersphere into a 2D circle (d), and projects a 4D paraboloid onto the plane such that the distribution still approximates the pattern of a parabola (c). Our results indicate that this method successfully extracts the intrinsic features of the high-D data distribution and intuitively visualizes them in the 2D plane, thus highlighting its great potential for empty-space visualization.

Algorithm 2: Dimension Reduction for Cosine Distance

```
Pick n nearest neighbors
  [P_1,P_2,...,P_n] around the agent π in d dimensional space;
for i ... n do
    Let P_i be the vector from π to P_i;
    l_i = ||P_i||;
    P_i = P_i/||P_i||;
end
Calculate a local embedding [p_1,p_2,...,p_n] from cos-MDS;
for i ... n do
    p_i = l_i * p_i/||p_i||;
end
return [p_1,p_2,...,p_n] as neighbor plot results.
```

APPENDIX C
PARAMETERS IN ESA

- **Particle effect diameter** σ: A significant parameter in ESA which have been explained in Equation (1) and the subsequent paragraph in Section IV.A.
- **Number of neighbors** k: Another significant parameter explained in Equation (3) and elaborated upon in the following sentence.
- **Discount factor** γ: Discussed in the fourth and fifth paragraphs of Section IV.B.
- **Number of steps** n: This parameter determines the number of iterations the ESA algorithm runs. If n is too small, the empty-space agent may not converge to the empty regions. Conversely, if n is too large, the agent may oscillate within the converged area, leading to unnecessary computational overhead.
- **Step size** α: It controls the magnitude of each search step, akin to the learning rate in gradient descent. A smaller α requires more steps to converge but offers finer search granularity. A larger α accelerates convergence but may skip over meaningful regions.

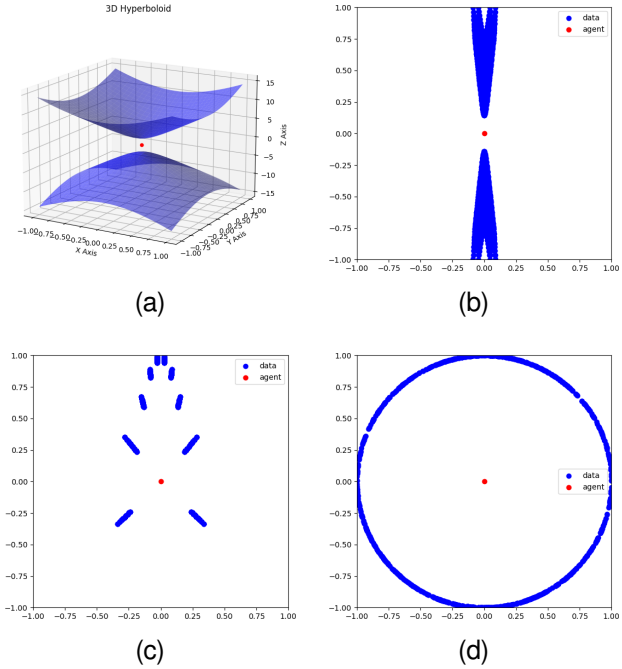

Fig. 10. Examples of cos-MDS on 3 synthetic datasets. The blue points are the synthetic dataset and the red point is the center agent. (a) The 3D hyperboloid dataset; (b) cos-MDS result on the 3D hyperboloid dataset; (c) cos-MDS result on the 4D paraboloid dataset; (d) cos-MDS result on the 4D hypersphere dataset

- **Rollout interval** j: This parameter dictates how frequently we sample ESCs from the search process. For instance, setting $j = 10$ means the algorithm selects the current position as an ESC every 10 steps. If j is too small, the sampled ESCs will be very similar and less informative. Conversely, if j is too large, the algorithm might miss significant configurations. The choice of j is related to the step size α; a smaller step size pairs well with a larger rollout interval, and vice versa.
- **Vanishing threshold** δ: This threshold evaluates the strength of the force acting on the agent. According to the nature of the Lennard-Jones (LJ) potential, the farther an agent is from its neighbors, the weaker the force becomes. If the force is too small, the agent is effectively in a nonsensical space and would require many more steps to reach the data distribution. Therefore, we set a vanishing threshold; if the force falls below δ, we assume the agent has vanished and discard it.
- **Constraints** f_i: The functions $f_1, f_2, ..., f_p$ represent dataset-dependent constraints on the variables. For example, in a dataset where each variable should be within $[0,1]$, or where the sum of the first and second variables must be greater than the third, these constraints are encapsulated by the f_i functions.
- **Cumulative magnitude** L: The cumulative magnitude works in tandem with the momentum m. As discussed in the penultimate paragraph of Section IV.B, while $\vec{m}$ maintains historical directions with a discount factor γ, L keeps track of the magnitude of $\vec{m}$, also discounted by γ.

Importance of k and σ: The number of neighbors k and the particle effective diameter σ are crucial parameters in our algorithm. Based on our experiments on heuristic optimization benchmarks (*e.g.*, Griewank, Rastrigin, Ackley, Sphere functions), we recommend setting k = dimensionality of the data space + 1 for lower dimensions (dimensions ≤ 32) so that the neighbors around the agent form a hypersphere. We recommend σ to be the mean distance to its k neighbors, approximating the radius of the hypersphere. A too-small k and σ limit the agent's ability to explore empty space, while too-large values prevent agents from converging to local empty spaces. For higher dimensions (dimensions >700 in image experiments), since the dataset is a manifold embedded in high-D space, k does not need to be dimensionality + 1. Empirically, we recommend k to be small (approximately 20) and σ to be half the mean distance to its neighbors. We acknowledge that more experiments are needed in higher dimensions to determine the optimal parameter settings.

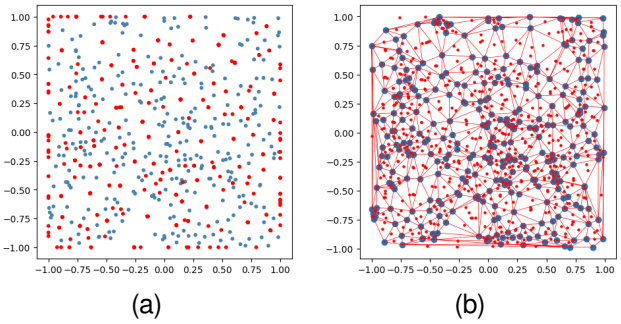

Fig. 11. Given 300 random samples (blue) and 600 random agents (red) in 2D space, we show the outcomes of empty-space search using (a) our empty-space search algorithm (ESA) and (b) Delaunay Triangulation (DT).

Empirical settings: We have empirically set the following parameters:

- **Number of steps** n: 100
- **Step size** α: 0.01
- **Rollout interval** j: 10
- **Vanishing threshold** δ: 10^{-5}
- **Discount factor** γ: ≤ 0.5 (if *momentum* is used). We recommend a discount factor $\gamma \leq 0.5$ when using *momentum*. A smaller γ prevents long-term effects caused by overwhelming repulsive forces when an agent is closer to a neighbor than σ, while still retaining useful historical information.

APPENDIX D
COMPARING ESA AND DELAUNAY TRIANGULATION

Our investigation into high-D Delaunay triangulation acceleration and approximation revealed a scarcity of research addressing the exponential complexity increase with dimensionality. Hence, we conducted some experiments to study this topic further. Our empirical findings underscore the computational and storage impracticality of these types of approaches for our purposes, even for datasets as small as 1,000 points, when the number of dimensions exceeds 7. Fig. 11(a) and Fig.11(b) show the empty-space search results from our search algorithm (ESA) and from Delaunay Triangulation (DT) in the 2D case, respectively. While both methods identify all the empty spaces, DT requires many more triangles compared to the number of agents used in our ESA. It is this abundance (and complexity) of geometric primitives that limits DT's scalability to higher dimensions.

APPENDIX E
USER-STUDY RESULTS

The user-study data including user performance, ESA performance, and random sampling performance are listed in Tab. I. The parameters used in this study and comparison experiments

TABLE I
REWARDS OF EACH ROUND OF OUR USER STUDY (TOP), ESA (MIDDLE) AND RANDOM SAMPLING (BELOW).

User1	User2	User3	User4	User5	User6	User7	User8	User9	User10	User11	User12	User13	User14	User15	User16	User17
0.487	0.4323	0.4317	0.36	0.301	0.368	0.43	0.518	0.287	0.317	0.283	0.385	0.4	0.463	0.377	0.317	0.433
ESA1	ESA2	ESA3	ESA4	ESA5	ESA6	ESA7	ESA8	ESA9	ESA10	ESA11	ESA12	ESA13	ESA14	ESA15	ESA16	ESA17
0.288	0.329	0.295	0.306	0.283	0.334	0.293	0.412	0.293	0.352	0.309	0.343	0.355	0.293	0.356	0.312	0.285
RND1	RND2	RND3	RND4	RND5	RND6	RND7	RND8	RND9	RND10	RND11	RND12	RND13	RND14	RND15	RND16	RND17
0.31	0.326	0.283	0.283	0.284	0.312	0.284	0.292	0.294	0.284	0.29	0.31	0.286	0.316	0.303	0.319	0.287

TABLE II
THE BETWEEN-SUBJECTS ANOVA (ABOVE) AND TUKEY HSD (BELOW) OF USER PERFORMANCE WITH GAPMINER (X1) AGAINST THE TWO BASELINES ESA-ONLY (X2) AND RANDOM SAMPLIGN-ONLY (X3).

Source	DF	SS	Mean Square	F Statistic	P-value
Groups	2	0.075	0.037	17.288	<0.001
Error	48	0.103	0.002		
Total	50	0.178	0.004		

Pair	Diff	SE	Q	CI Bounds	CM	P-value
x1-x2	0.068	0.011	6.019	0.029 ~0.106	0.039	<0.001
x1-x3	0.090	0.011	7.979	0.051 ~0.128	0.039	<0.001
x2-x3	0.022	0.011	1.959	-0.016 ~0.061	0.039	0.356

were $k=9$, $\sigma=$ mean distance of k neighbors, $n=400$, $\alpha=0.001$, $j=10$, $\delta=10^{-7}$ respectively. Momentum is not used.

A. ANOVA of User Study

The ANOVA of the user study can be found in Tab. II.

B. System Usability Scale

The SUS questionnaire included the 10 questions listed below:

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system

Each question was ranked from 1 (strongly disagree) to 5 (strongly agree). The final score was calculated by summing the normalized scores and multiplying them by 2.5 to convert the original score of 0-40 to 0-100. 12 users responded to our questionnaire. The results are given in Tab. III.

C. Comparison Experiment Data

Fig. 12 presents detailed results achieved by ESA, random sampling and random walk.

TABLE III
RESPONSES FROM 12 USERS. Q1–Q10 REFER TO 10 QUESTIONS; R1–R12 REFER TO USER RESPONSES. NS IN THE LAST ROW IS THE NORMALIZED SCORE OVER 12 RESPONSES.

Resp	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
R1	3	2	4	2	4	2	4	2	4	3
R2	5	1	4	4	5	1	2	2	4	3
R3	5	2	3	2	3	1	4	2	5	4
R4	2	3	4	4	5	1	5	3	4	4
R5	4	1	4	2	5	1	5	1	5	3
R6	3	3	3	4	5	3	3	3	3	5
R7	5	1	5	1	5	1	5	1	5	1
R8	4	1	5	2	5	1	4	1	4	1
R9	5	1	4	2	5	1	4	2	5	1
R10	4	2	4	3	4	2	4	2	4	2
R11	4	4	3	4	5	1	4	2	4	3
R12	5	1	5	4	5	1	5	1	5	1
NS	3.1	3.2	3.0	2.2	3.7	3.7	3.1	3.2	3.3	2.4

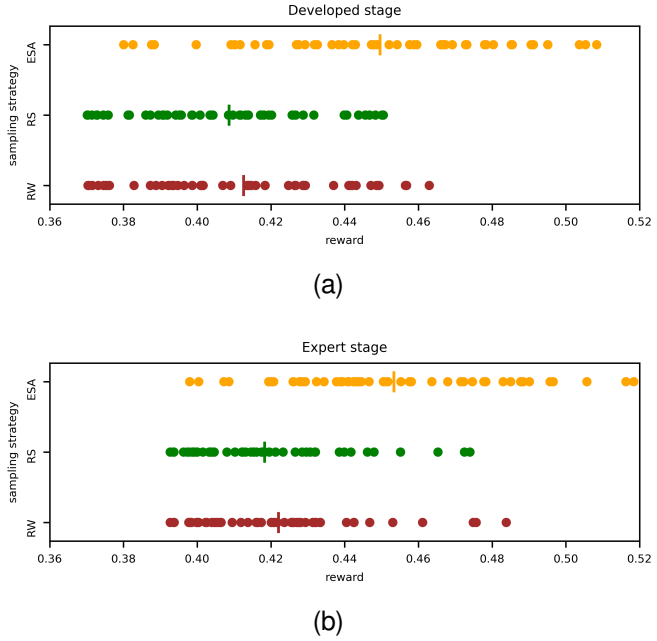

Fig. 12. Comparison experiment: (a) shows the rewards found by ESA, random walk, and random sampling at the development stage; (b) shows the rewards found by the three methods at the expert stage. In both figures, each dot represents a search process with 1,500 random agents involved. Each method in both stages has 50 search processes.

D. ANOVA of Comparison Experiments

The ANVOA of developed stage and expert stage can be found in Tab. IV and Tab. V respectively.

TABLE IV
THE WITHIN-SUBJECTS ANOVA (ABOVE) AND TUKEY HSD (BELOW) OF ESA (X1), RS (X2), AND RW (X3) AT A DEVELOPED STAGE. SUBJECTS ARE THE INITIAL POSITIONS OF AGENTS, TREATMENTS ARE THE EMPTY-SPACE SEARCH METHODS.

Source	DF	SS	MS	F Statistic	P-value
Between Subjects	49	0.042	0.001	1.115 (49,98)	0.320
Between Treatments	2	0.051	0.026	33.224 (2,98)	$<\mathbf{10^{-6}}$
Error	98	0.075	0.001		
Total	149	0.168	0.001		

Pair	Diff	SE	Q	CI Bounds	CM	P-value
x1-x2	0.041	0.004	10.250	0.028~0.054	0.013	$<\mathbf{10^{-6}}$
x1-x3	0.037	0.004	9.272	0.024~0.050	0.013	$<\mathbf{10^{-6}}$
x2-x3	0.004	0.004	0.979	-0.009~0.017	0.013	0.769

TABLE V
THE WITHIN-SUBJECTS ANOVA (ABOVE) AND TUKEY HSD (BELOW) OF ESA (X1), RS (X2), AND RW (X3) AT AN EXPERT STAGE. SUBJECTS ARE THE INITIAL POSITIONS OF AGENTS, TREATMENTS ARE THE EMPTY-SPACE SEARCH METHODS.

Source	DF	SS	MS	F Statistic	P-value
Between Subjects	49	0.033	0.001	1.201 (49,98)	0.220
Between Treatments	2	0.037	0.019	33.233 (2,98)	$<\mathbf{10^{-6}}$
Error	98	0.055	0.001		
Total	149	0.125	0.001		

Pair	Diff	SE	Q	CI Bounds	CM	P-value
x1-x2	0.035	0.003	10.161	0.024~0.047	0.012	$<\mathbf{10^{-6}}$
x1-x3	0.031	0.003	9.082	0.020~0.043	0.012	$<\mathbf{10^{-6}}$
x2-x3	0.004	0.003	1.079	-0.008~0.015	0.012	0.726

APPENDIX F
EXTRAPOLATION BEHAVIOR OF OUR ESA-BASED SEARCH

Our ESA focuses on identifying empty spaces within the data distribution. The repulsive component prevents it from straying too far from the existing distribution. However, empty spaces outside the current distribution can be meaningful and essential. This experiment demonstrates ESA's ability to search for these empty spaces beyond the current distribution.

We used the distribution of the wine dataset for demonstration; we ignored the meaning of each variable and removed the target variable to focus solely on the distribution of the dataset in each dimension. This marked the dataset as the initial distribution. We ran ESA for six iterations, selecting 300 data items from the dataset with the largest average distance to their 8 nearest neighbors as the initial agent position. ESA search was conducted, and the results were added to the dataset. After each iteration, we measured the distance of the ESA results from the initial distribution and created a histogram. The histograms, shown in Fig. 13, indicate that the configurations found by ESA progressively move further from the initial distribution. This demonstrates ESA's ability to explore empty spaces beyond an initial data distribution.

APPENDIX G
DETAILS FOR THE APPLICATION CASES

In this section, we present more detail on the various application examples shown in the paper.

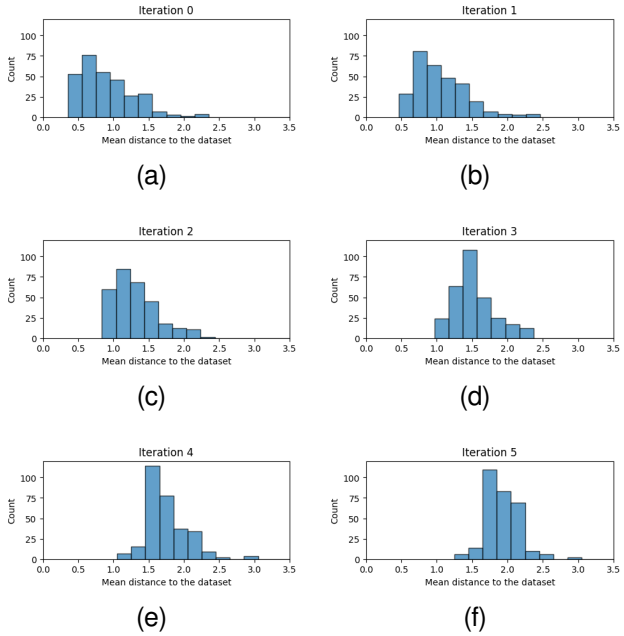

Fig. 13. Histograms of the configurations identified by the ESA with regards to the initial distribution as a function of iteration. We observe that the ESA results progressively move further from the initial distribution, demonstrating ESA's ability to explore empty spaces beyond an initial data distribution.

A. Systems Dataset

Fig. 14 shows the full view of GapMiner for the application example described in Fig. VII, the systems dataset.

B. Wine Investigation

Fig. 15 show the full view of GapMiner for the wine dataset.

C. Cheetah Direction

We demonstrate an additional application of GapMiner in a physical-simulation scenario.

CheetahDir is an environment in MUJOCO that controls a half cheetah robot to move forward. The reward function is composed of two components: forward reward and control cost. Forward reward is the velocity in the given direction, the faster the better. Control cost is the rooted squared sum of the action, indicating the cost to execute the action, the smaller the better. The total reward is the forward reward minus the control cost. The action space has 6 variables: back thigh, back shin, back foot, front thigh, front shin and front foot. All of them are numerical and continuous.

The dataset is collected by an expert agent. Given an observation, it samples actions from its output distribution randomly. In this scenario, we explored the ability to find better empty-space actions than the expert agent given the observation. The visual analytics can be found in Fig. 16.

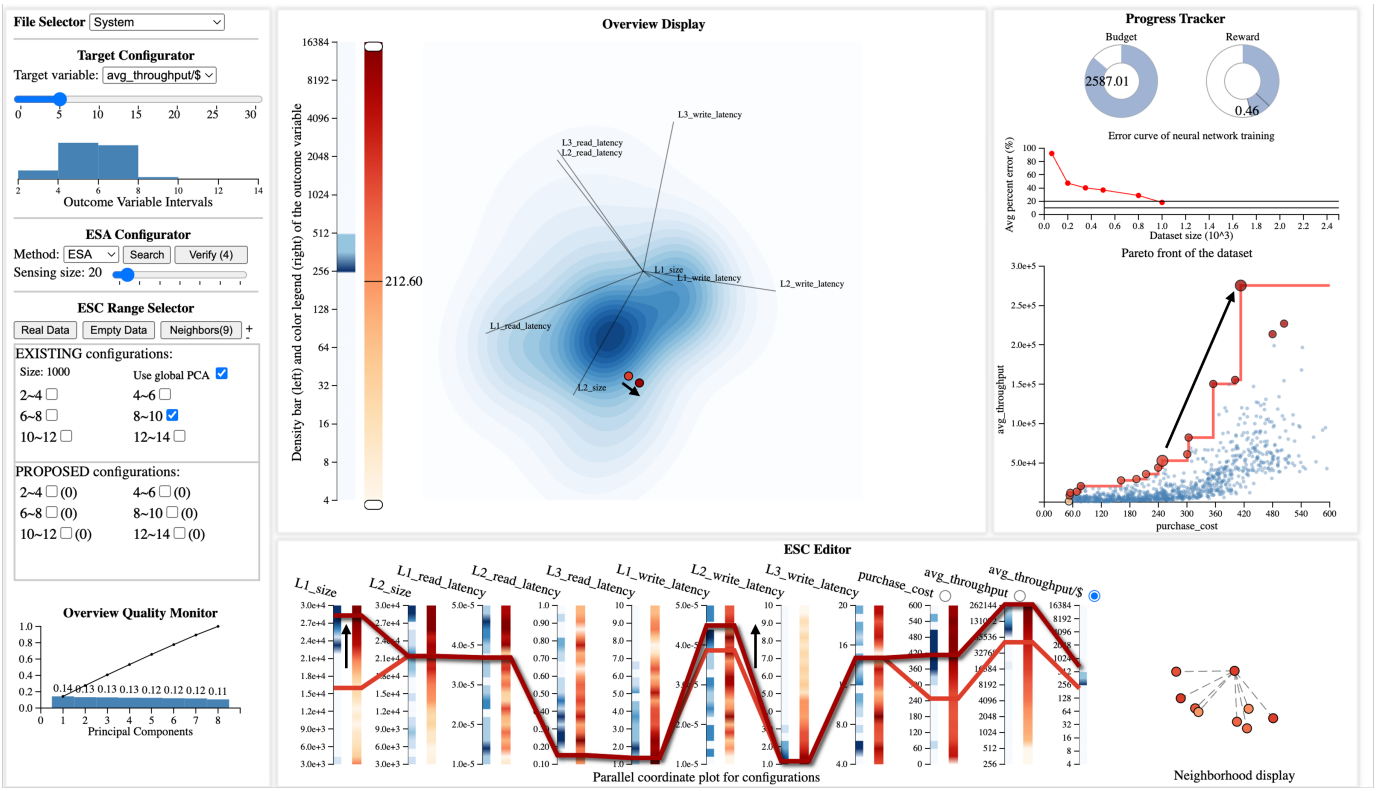

Fig. 14. The full view of GapMiner for the system dataset. The black arrows in the overview display, the PCP, and the Pareto frontier plot show the direction of optimization.

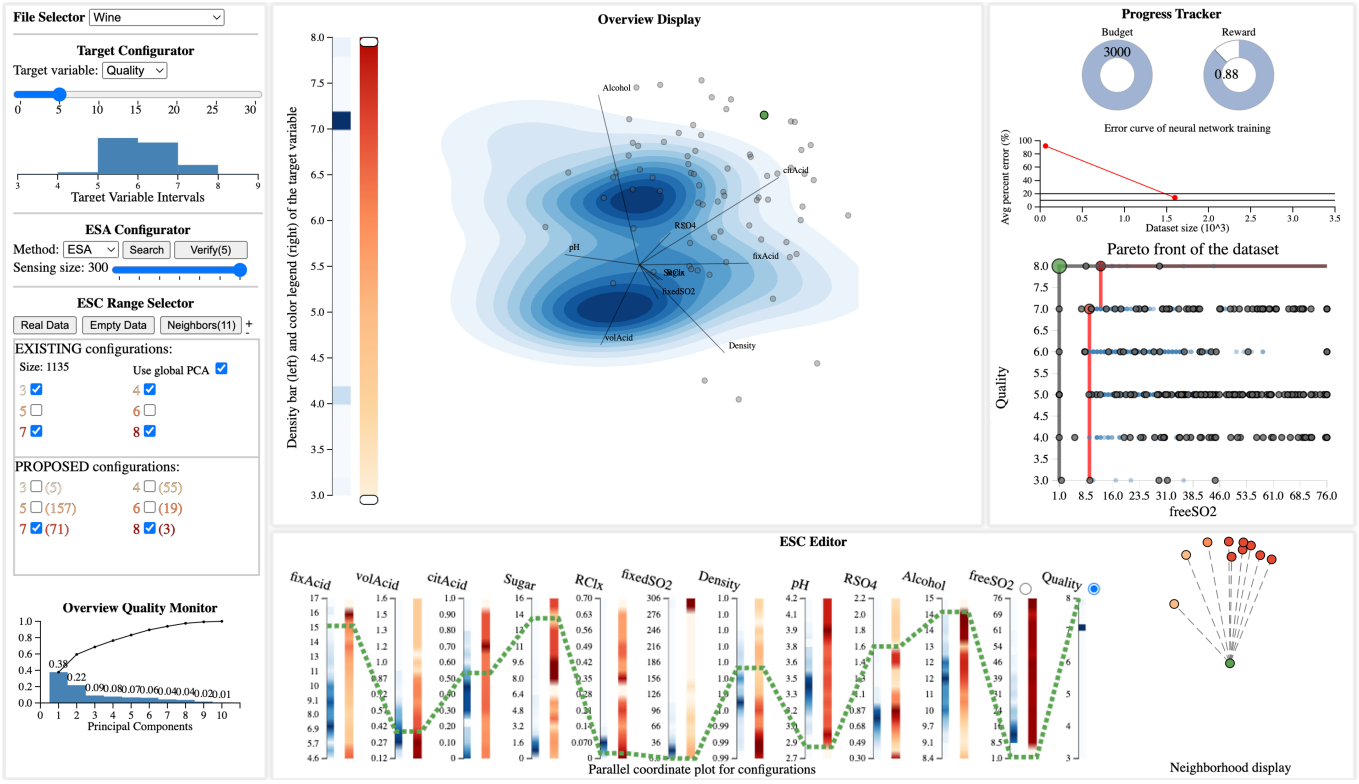

Fig. 15. The full view of GapMiner for the wine dataset study. We highlight the global optimal wine instance in green in the PCA map, the PCP, and the neighbor plot.

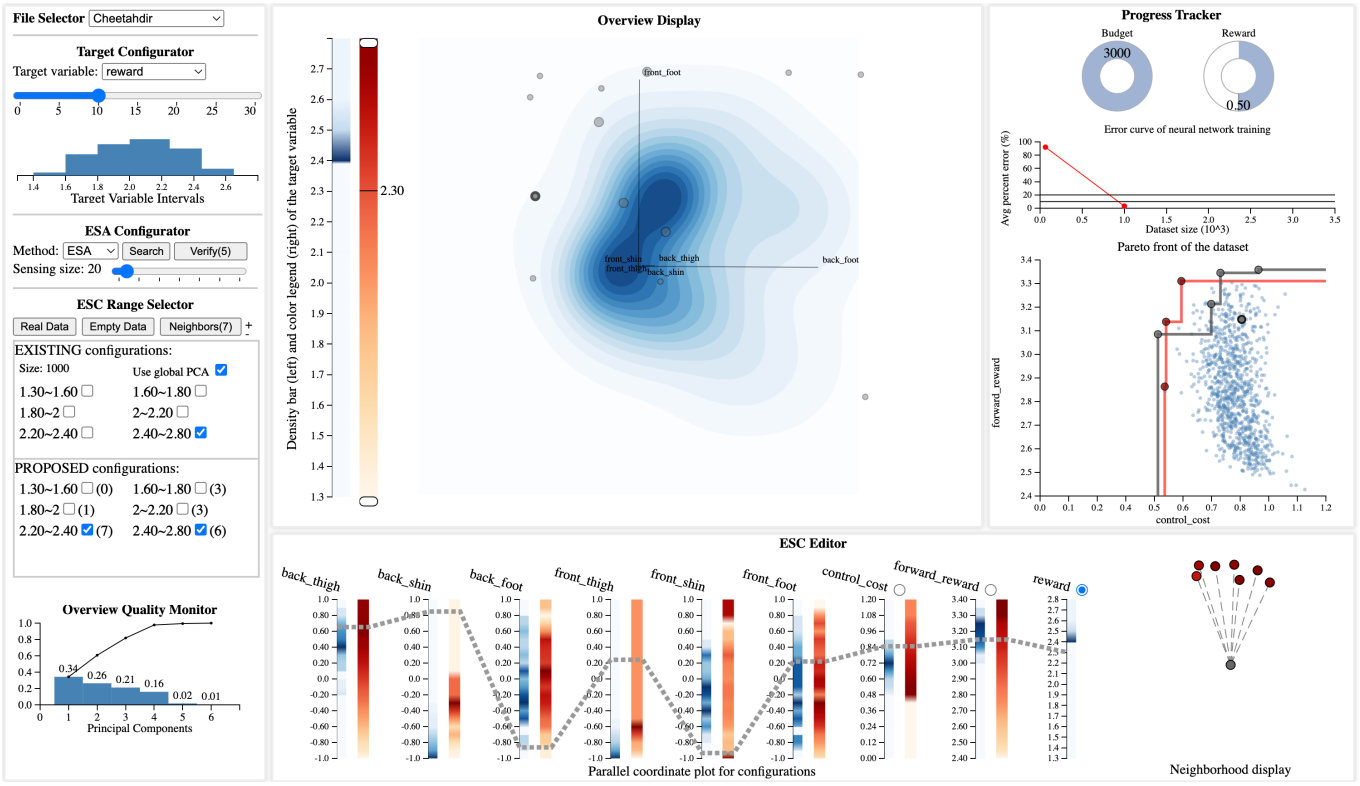

Fig. 16. We trained a neural network to predict forward reward and calculate the control cost; then we calculate the reward of an action as forward reward minus control cost. We applied the dataset to GapMiner, from which we could see a distribution shift from low-reward actions to high-reward actions. It indicated that front foot and back foot are more important to the reward. Then we ran ESA to search empty-space actions and plot the predicted forward reward and control cost in the Pareto plot. It showed that actions from the expert agent always have high cost, but some don't have high reward. ESA can find actions that are either cheaper while maintaining good reward, or expensive but even higher reward. Notice that RL is a sequential decision-making problem. A high immediate reward doesn't necessarily imply a long-term benefit. Searching empty-space actions with higher Q values is another interesting application scenario.