Resource-effective exploration of tumor treatments with multi-scale simulations

Vasileios Stavropoulos
v.stavropoulos@iit.demokritos.gr
Institute of Informatics and Telecommunications,
NCSR "Demokritos"
Athens, Greece

Evangelos Michelioudakis
vagmcs@iit.demokritos.gr
Institute of Informatics and Telecommunications,
NCSR "Demokritos"
Athens, Greece
Dpt. of Informatics and Telecommunications,
National and Kapodistrian University of Athens
Athens, Greece

Charilaos Akasiadis
cakasiadis@iit.demokritos.gr
Institute of Informatics and Telecommunications,
NCSR "Demokritos"
Athens, Greece

Alexander Artikis
a.artikis@iit.demokritos.gr
Dpt. of Maritime Studies, University of Piraeus
Piraeus, Greece
Institute of Informatics and Telecommunications,
NCSR "Demokritos"
Athens, Greece

ABSTRACT

Machine learning is regularly used to interpret and analyze information from large and complex datasets originating from numerous fields. In Bioinformatics, the exploration of potentially beneficial drug configurations for tumor treatments via simulations requires multiple processing units to be used in parallel and a considerable amount of time to be completed. In this paper, we apply a state-of-the-art model exploration workflow for the characterization of a new drug configuration parameter space, using a redesigned simulator. Moreover, we incorporate different clustering and optimization approaches and compare their performance in in-silico simulation trials on high-performance computing infrastructure, with respect to time and resource efficiency. The overall goal is to discover regions in this parameter space that can lead to more viable treatments in reasonable time, and thus guide the related research towards more focused and effective real-world trials.

SETN 2022, September 7–9, 2022, Corfu, Greece

ACM ISBN 978-1-4503-9597-7/22/09…$15.00
https://doi.org/10.1145/3549737.3549745

CCS CONCEPTS

• **Computing methodologies** → **Cluster analysis**; **machine learning algorithms**; • **Applied computing** → **Bioinformatics**.

KEYWORDS

Active Learning, Genetic Algorithms, Simulated Annealing, Tumor Simulations, Computational Biology

ACM Reference Format:
Vasileios Stavropoulos, Evangelos Michelioudakis, Charilaos Akasiadis, and Alexander Artikis. 2022. Resource-effective exploration of tumor treatments with multi-scale simulations. In *12th Hellenic Conference on Artificial Intelligence (SETN 2022), September 7–9, 2022, Corfu, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3549737.3549745

1 INTRODUCTION

Cancer disease is one of the leading causes of death globally, being responsible for approximately 10 million in 2020 according to the World Health Organization.[1] The alarming statistics have made the discovery of promising treatments a top priority for the medical and biological research community [15]. However, the dynamic nature and the mortality of the disease make clinical experiments difficult, since the recovery time window is often narrow and any possible error or miscalculation may have devastating results to the patient's health. Computational biology has assisted the researchers in drug discovery [3, 11] by providing models that

[1] https://www.who.int/news-room/fact-sheets/detail/cancer

attempt to describe the behavior of tumor cells in the human organism. These models allow practitioners to study via simulations the effectiveness of various tumor treatments *in-silico*, without putting the human lives on the line.

Although such simulations are detailed and insightful, in most cases they are computationally expensive, hindering the researchers from exhaustively examining the effect of every possible treatment configuration. To that end, Computational Intelligence and Machine Learning methods can be used to reduce the required number of simulations in an informed manner, and to provide accurate predictions about the effectiveness of treatments. In particular, Ozik et al. [19] presented a workflow that aims to discover the characteristics of promising drug treatments, by identifying the value ranges of the drug-related parameters that can induce desirable effects, i.e. that reduce tumor growth and its resistance to the drug. This workflow consists of two parts. The first one comprises an Active Learning (AL) algorithm [23] that aims to identify the interesting regions of the treatments parameter space, by iteratively evaluating sample points and training a classifier. The second part consists of a Genetic Algorithm (GA) for discovering optimized treatment configurations. This workflow provides a valuable tool in the researchers' arsenal and enables them to perform targeted and promising real-world trials.

In this paper, we apply a similar model exploration workflow combined with a different simulator. Specifically, we examine the effect of a signal molecule that can induce death in tumor cells in a similar manner to tumor treatments, called *Tumor Necrosis Factor* (TNF) [7]. TNF is a critical cytokine that binds to cell receptors and activates signalling pathways to cell necrosis, restraining in this way the growth and spread of tumor cells. We conduct experiments utilizing PhysiBoSSv2.0,[2] an add-on that expands upon the PhysiCell cell simulator [10]. PhysiBoSSv2.0 introduces an agent-based multi-scale model for tumor cell growth used to examine the impacts of given drug configurations [2]. Our overall goal is to sample subsets of the TNF related parameter values that resemble different treatment configurations and simulate the effect of each treatment on high-performance computing infrastructures so that to discover the most effective ones.

In particular, the evaluated samples are used to train a classifier, which then indicates the most uncertain regions that in turn determine the samples to be evaluated in the following iterations. These regions, however, presumably contain an infinite number of points. In order to guide the exploration, the uncertain points are clustered according to their similarity, e.g. using the K-Means algorithm, and only a few representatives are finally evaluated via simulations. Since the requirements for computational resources to sufficiently explore alternative treatments are quite high, it is important to devise faster and less resource-demanding alternatives for such approaches, nevertheless without compromising overall performance. Such modifications can be proven really valuable in even more complex settings, such as when exploring the synergistic effects of multiple drugs being administered simultaneously to the patient [17].

The clustering component of the sampling process plays an important role in the number of the required simulations and the results produced. Based on their design principles, different methods may lead to results with quite noticeable differences. Moreover, most require user-defined hyper-parameters in order to operate, which may be difficult for non-experts to determine in advance. To this end, we examine the effect of additional clustering methods, i.e., DBSCAN and BIRCH, to the effectiveness of the sampling process in the workflow, and compare their performance. Furthermore, we also examine an additional optimization method, that is Simulated Annealing, for the discovery of the most efficient treatments, which is expected to be more resource efficient. Our experiments demonstrate that there can be differences in the performance of each clustering algorithm and optimization method. Also, results indicate the existence of a trade-off between the amount of simulations performed and the stability of the produced solutions, both during the characterization of the treatment configuration space and the discovery of the most effective treatments.

The remainder of this paper is structured as follows. In Section 2 we present the related work and the details of the parameter space exploration workflow. Then, in Section 3 we discuss the additional clustering and optimization alternatives that we evaluate. In Section 4 we present the experimental results of the treatment configurations space exploration, while in Section 5 we summarize the work and propose further research directions.

2 BACKGROUND & RELATED WORK

ML methods are widely used in Computational Biology and Bioinformatics to discover behavioral patterns of biological systems. The predictive models generated by ML give insights to the functional relationships of the systems and provide accurate statistical predictions in a range of biological applications [4, 8]. For instance, Błażewicz et al. [5] proposed a time-effective method to discover low energy protein structures using a heuristic optimization method. The authors combine simplified protein structure prediction models and the Tabu Search algorithm in order to discover the native structure of the protein. An essential feature of the Tabu Search algorithm is the exclusion of possible candidate solutions from being evaluated as they are not expected to be of interest. In a similar manner, the AL algorithm in

[2] https://github.com/bsc-life/PhysiBoSSv2

our workflow identifies non-interesting regions of the space and focuses the exploration to more interesting ones. The non-interesting regions are then eliminated from the best treatment search by the GA during the optimization part.

GAs have also been combined with clustering techniques in Active Learning workflows for the classification of biomedical images. [13] adopts such an approach combined with self-organizing maps, to reduce the amount of manual labor required for annotating and analyzing cancer patient screening images. Interactivity allows human supervision and intervention, but this is only required in a smaller scale. Unsupervised learning helps to detect uncertain regions and ask for more targeted input by experts, while automatically expanding learned classification rules to known cases.

Mohamed et al. [16] incorporate AL for guiding the selection of protein pairs for future experimentation in order to accelerate accurate predictions of the human protein interactome. Random selection for uncertainty sampling with random forests is compared against density-based sampling using K-Means. The results suggest that AL manages to accelerate the discovery of interacting protein pairs, even in datasets where the ratio of interacting pairs is very low.

The work of [20] presents an overview of AL methods that are used to detect potentially promising regions of drug configurations that maximize the desired treatment effects, both in prospective and retrospective. The behavior of particular unsupervised learning methods in such domains however, has not received enough attention so far.

Ozik et al. [19] presented an approach for assisting in the tumor treatment discovery. The method consists of two parts: in the first one, an AL algorithm is used in order to divide the treatment parameter space into promising and non-promising regions, while in the second part, a GA is used in order to discover the most promising treatment configuration and to validate the results of the parameter space characterization of the first part. The AL algorithm utilizes a Random Forest classifier to divide the parameter space into viable and non-viable areas, and a K-Means algorithm to cluster the points in the most uncertain regions, i.e. that is not definite if they are viable or not, to focus the search in the subsequent exploration steps. An area is considered to be viable if it includes treatments that manage to reduce the count of the alive tumor cells below a set threshold by the end of the simulation. Although this workflow yields promising results, it suffers from various limitations regarding its performance. In the AL part, and in particular during the sampling process, K-Means fails to identify clusters of arbitrary shape and to eliminate noise in the candidate points set. It is also unable to dynamically adjust the number of clusters to the spatial distribution of each different candidate set. These restrictions lead to an unstable sampling process, as outlying points are taken into account in the formation of the clusters and, in many cases, larger clusters are broken down to smaller ones (or vice versa) in order to reach a user-defined number, which is a required configuration parameter. Such restrictions might degrade the sampling process and often result in the execution of redundant or less informative simulations. To overcome these shortcomings, we incorporate BIRCH and DBSCAN as alternative clustering techniques for the first part of the workflow, and also Simulated Annealing (SA) for the optimization in the second part. BIRCH and DBSCAN are more noise-tolerant than K-Means, thus are expected to designate more suitable points for evaluation. Moreover, the SA operates on a single candidate point, instead of a larger population, as the GA does, thus requiring less simulations to be performed.

3 WORKFLOW FOR TUMOR TREATMENT EXPLORATION

In this section, we present the tumor treatment exploration workflow that we incorporate, and we introduce the alternative clustering and optimization methods for the exploration and the optimized treatment discovery, respectively.

The various steps of our approach are illustrated in Figure 1. To begin, each three dimensional point of the parameter space corresponds to a specific treatment configuration, indicating the values for TNF duration and frequency of administration, and the TNF concentration. Our goal is to find the values for these three parameters that lead to the most effective treatments. Starting from a number of randomly selected points of the unexplored space, we evaluate them by configuring the simulator with the respective parameter values and then performing the in-silico trials on the HPC. According to the obtained results, i.e. the number of tumor cells that are still alive at the end of the simulation, we train a Random Forest (RF) classifier, which is then queried to obtain the labels of a grid of unexplored points—either interesting or non-interesting, according to the treatment effectiveness. Next, the ones for which the RF's decisions are the most uncertain, are clustered to determine only a subset in order to proceed with their evaluation in the following iteration of the algorithm. This point selection process significantly reduces the number of required simulations—as we only evaluate the centroids of each cluster instead of the whole set—and ensures the spatial diversity of the points that are to be evaluated next. Then, the simulations take place and their results are used to augment the training set of the RF. Next, the RF is retrained using the updated set of evaluated points, leading to a more refined characterization of the parameter space as the execution progresses, up to a user-defined number of iterations, which is the termination criterion for the first phase of the workflow.

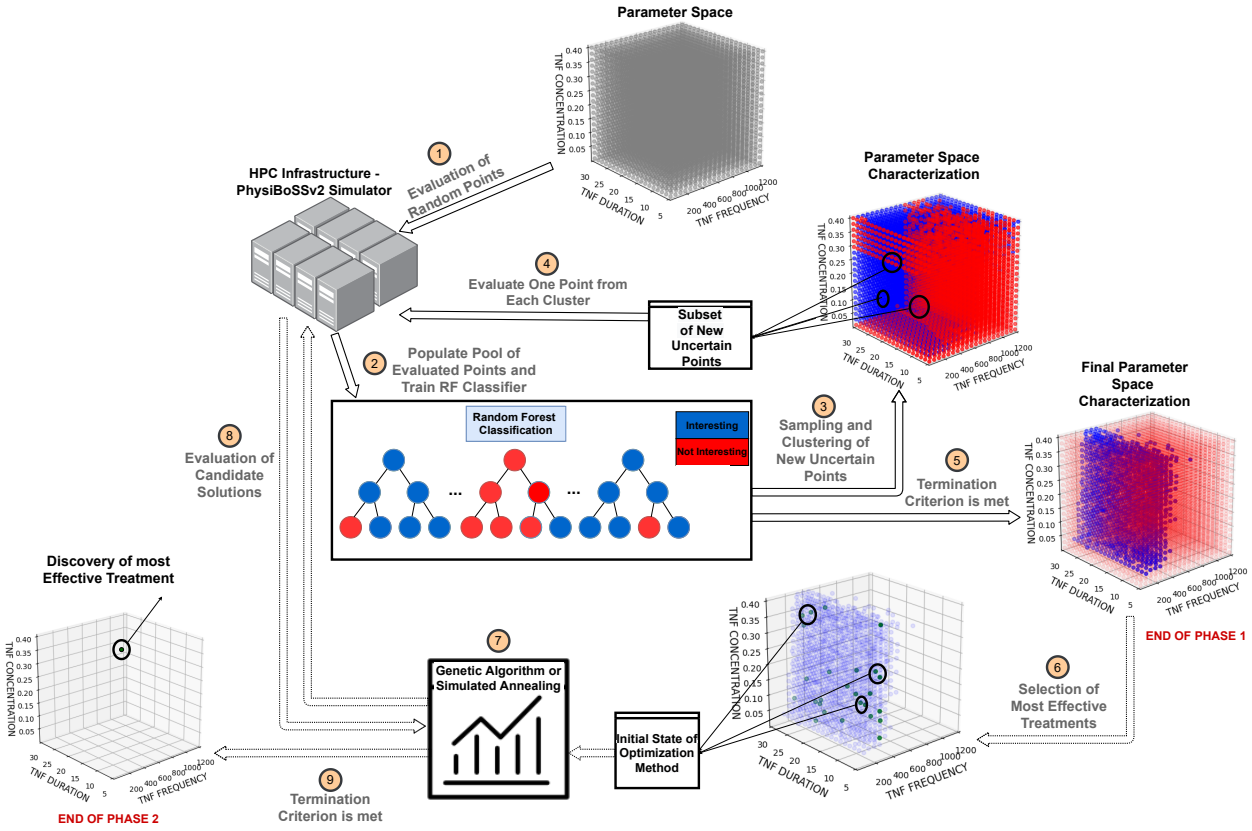


Figure 1: Overview of the examined workflow.

Continuing with the optimization phase (bottom of Figure 1, marked with dashed arrows), an evolutionary algorithm is used, either GA or SA, in order to discover the most promising treatments, ones that lead to the lowest tumor cell counts. For the initialization of the evolutionary algorithms, we use the best points discovered so far by the AL part. The selected points are combined to generate new ones following the respective procedure according to the selected evolutionary algorithm, and the new points are evaluated on the HPC. This repeats up to a user-defined number of iterations.

3.1 Clustering Algorithms

The two key components of the AL part of the workflow are the RF classifier and the clustering algorithm that is used to reduce the number of candidate points for evaluation in the HPC. Here, we discuss different clustering algorithms which can help our workflow to explore the treatment configurations space faster. Such algorithms are used in a plethora of fields for the identification of similarities between different data instances. The three main categories of clustering algorithms are partitioning, hierarchical and density-based [1]. The partitioning type aims to divide the data instances space into k clusters. Finding the optimal k parameter is crucial and requires prior knowledge regarding the true distribution of the instances. This knowledge is usually not available in most applications and especially in our problem, where the candidate points may vary between iterations. Hierarchical algorithms on the other hand focus on the space decomposition. This is represented as a tree that splits the space into smaller clusters until a termination criterion is reached. Finally, density-based algorithms cluster the data based on their density in the space. Clusters are considered to be sets of points of high density separated by lower density regions. For our purposes, we focus on a representative algorithm from each of these categories, in particular the K-Means, BIRCH, and DBSCAN algorithms [24].

K-Means. One of the most widely used clustering methods is the K-Means algorithm [14]. In brief, K-Means consists of three steps. First, K points are selected from the dataset to form the initial centroids. Then, the remaining points are assigned to the nearest centroid, and each centroid is redefined as the center of mass of all the points assigned to that cluster. These two steps are repeated until convergence is reached.

The K-Means algorithm is simple and fast, and was utilized by [19] for the AL part. However, it entails limitations that may lead to poor accuracy. First, the number of clusters is difficult to determine beforehand, as in most cases there is no prior knowledge regarding the spatial distribution of the dataset. Second, clustering may be affected by noise, as outliers are considered for the calculation of the centroids. Also, it mainly identifies spherical clusters of similar sizes, while it fails to find ones of arbitrary sizes and densities.

DBSCAN. Ester et al. [9] introduced the DBSCAN algorithm, a density-based clustering approach designed to identify clusters of arbitrary shapes. DBSCAN relies on the *Eps* and *MinPts* parameters. *Eps* defines the maximum distance between two points to be considered as neighbors. *MinsPts* is the minimum number of points required to be in the neighborhood of a point p in order for the point p to be considered a core point of a cluster. According to DBSCAN, points can be divided into three categories, the core, border, and noisy points. Core points refer to representative points of the cluster, while border points are the ones on the edges of the cluster. Every other one is considered as an outlying point. The algorithm arbitrarily picks a point p and calculates the number of points in its *Eps* neighborhood. If the number of neighboring points is greater than *MinPts*, then p is the core point and a cluster is formed. If p is a border point, then DBSCAN visits the next point in the dataset. The process is repeated until all points are examined. DBSCAN is a commonly used algorithm due to its ability to identify clusters of arbitrary shapes and sizes and its robustness to noise.

BIRCH. BIRCH is a hierarchical clustering algorithm introduced by Zhang et al. [25] to handle large datasets efficiently. Its efficiency is achieved by creating a summary of the dataset and then processing this summary, instead of clustering the original dataset as a whole. BIRCH is based on the concepts of the *Clustering Feature* (CF) vector and the *CF tree*.

Given N d-dimensional data points in a cluster x_i, where $i = 1, 2, \ldots, N$, the clustering feature vector (CF) of the cluster is defined as a triple $CF = \{N, LS, SS\}$, where N is the number of data points in the cluster, LS is the linear sum of the N data points and SS is the square sum of them. A CF Tree is a height-balanced tree, which acts as a compact representation of the original dataset. A leaf node in the tree contains at most L entries of CFs and two pointers linking the node to the previous and the next leaf node. Internal nodes contain entries of the form $[p, CFp]$, where p is a pointer to a child node and CFp is the sum of all the CFs in the child node. Each CF Tree requires two parameters, the branching factor B and the threshold T. Each internal node of the tree can contain at most B entries and the diameter of each leaf entry has to be less than T. The algorithm scans the data and creates a CF Tree by iteratively selecting data samples. At each step, a new data sample is selected and the nearest leaf node sub-cluster in the existing CF tree is obtained. If the distance between the centroid of the closest sub-cluster and the new sample is less than the threshold T, then the sample is added to the sub-cluster and the properties of the leaf node and its parent nodes are updated. Otherwise, a new sub-cluster is created and added to the CF Tree. In case the addition of the new sub-cluster breaks the branch factor condition, then the parent node is split. In this way, outlying points do not distort the existing clusters and can be isolated into smaller clusters consisting only from close noisy points. In our approach, we consider the leaf nodes of the CF Tree as the final clusters. These clusters can be categorized to ones containing informative points and those containing outlying points. BIRCH is a fast algorithm, which efficiently clusters large datasets, it does not require specifying the number of clusters and it can also detect outlying points. Table 1 summarizes the characteristics and limitations of each clustering method under examination.

3.2 Searching for Optimized Treatments

Heuristic search methods, such as the GA, have been developed in order to solve optimization problems that are difficult or even impossible to be reduced to an analytical form and thus solved by exact numerical algorithms. Such methods require little or no prior knowledge of the problem's domain and aim at the discovery of the global minimum (or maximum) of an objective function. Although they cannot provide guarantees for finding the true globally optimal solution, they can discover many "good" solutions that are locally optimal. Application examples come from a wide variety of fields, e.g. bioinformatics, power systems, etc. [18].

In addition to the GA incorporated by [19], we also examine the application of the Simulated Annealing (SA) optimization technique [6, 12] for discovering the best treatment configurations. SA is a probabilistic optimization method that mimics the process of metal annealing, in which a metal is heated and cooled slowly in order to solidify its crystals and reach an optimal state of minimum energy. The basic elements of the SA method are the set of possible points S, an energy function $E : S \mapsto \mathbb{R}$ (objective function), an initial temperature (T_o), a minimum temperature (T_{min}), the temperature at k_{th} level T_k, the number of iterations in each temperature level N and the cooling schedule. The SA algorithm consists of two nested loops. The algorithm starts from an initial point (current solution s) and evaluates its "energy". In the inner loop, the set of neighbors of the current point is generated, a random neighbor n is selected and, in turn, its "energy" is evaluated. The selected neighbor is accepted as

Table 1: Examined Clustering Methods Overview

Method	Characteristics	Limitations
K-Means	Clusters of similar shape and size	Difficult definition of optimal K Clustering data of varying sizes and density
DBSCAN	Clusters of varying size and shape, Noise detection	Parameter sensitive Varying density clusters
BIRCH	Time and memory efficient, Noise detection	Not scalable for high dimensional data

the new solution s with probability:

$$P_{accept}(n) = \begin{cases} 1, & E(n) \leq E(s) \\ exp\left(-\frac{\Delta E}{kT_k}\right) & E(n) > E(s) \end{cases}$$

i.e. the new candidate solution is always accepted if it performs better than the current solution. Otherwise, the candidate solution is accepted with an acceptance probability. The acceptance probability decreases exponentially with the inferiority of the candidate solution. The inner loop is repeated until N iterations are completed (equilibrium condition). The acceptance of inferior candidate points gives the ability to escape from local minima and to continue the search for the globally optimal solution. In the outer loop, the temperature level is decreased according to the cooling schedule, until reaching T_{min} (cooling condition).

At high temperature levels, the SA is more tolerant towards moving to inferior solutions and aims at the discovery of a "good" neighborhood. As T decreases, smaller deterioration in energy is allowed, and focus is put into the discovery of the globally optimal solution. If the search space is small, SA and GA yield similar results, as both methods perform well in the improvement of the set of parameters around an initial solution [21]. However, the SA focuses on a single candidate solution, while the GA maintains a population of possible solutions. This difference makes the SA less demanding in terms of computational resources when facing such problems. We take advantage of this characteristic and apply SA in the optimized drug treatments discovery by incorporating the knowledge we have deprived from the AL part of the workflow, focusing only on regions already classified as viable. In particular, we set the initial point of the SA to be one of the most promising treatments evaluated by the AL.

4 EXPERIMENTAL RESULTS

Our implementation can be found in an online repository, along with instructions on how to reproduce the experiments presented in this section.[3] The scikit-learn Python library[4] was used for the clustering methods and the DEAP Python library for the GA.[5] The SA is based on a custom implementation. All experiments were performed on the Mare Nostrum 4 (MN4) HPC infrastructure provided by the Barcelona Supercomputing Centre.[6] The hyperparameter configuration of each method was obtained after performing smaller scale experimental runs. First, we present the results of the clustering methods for the sampling process of the AL part, and then we compare the performance of the GA against the SA for discovering the most promising treatments.

4.1 Performance of different clustering methods in the active learning

In the original approach of [19], K-Means clustering was applied for the AL part. The number of clusters (k) was set to 20. Thus, to obtain a baseline, we examine the performance of K-Means variants with k equal to 20 and 50. We refer to the aforementioned configurations as KMEANS_20 and KMEANS_50, respectively. Moreover, we apply a K-Means clustering with $k = 500$, referred to as KMEANS_500, which acts as a benchmark and allows us to compare the performance of the clustering methods throughout the experimental process with that of an exhaustive method. For the evaluation purposes, the parameters of DBSCAN were configured to $Eps = 0.025$ and $MinPts = 20$ and the parameters of BIRCH with branching factor $B = 100$ and distance threshold $T = 0.1$, as such values were shown to be a good selection from smaller scale trials. We define the effectiveness of each treatment configuration as follows:

$$\text{Tumor Cell Survival Rate} = \frac{\text{Final Tumor Cell Count}}{\text{Initial Tumor Cell Count}}$$

i.e., as the ratio of the count of alive tumor cells after treatment to the count of the initial alive tumor cells, for a simulation duration of 24 hours. This metric reflects the number of the final alive cells as a percentage of the initial alive cells before the application of the treatment. Viable treatments are considered those that achieve an effectiveness score of less than 0.3. For each experiment, the AL part was run for 20 iterations. An exhaustive sweep search of the parameter

[3] https://github.com/xarakas/spheroid-tnf-v2-emews
[4] https://github.com/scikit-learn/scikit-learn
[5] https://github.com/DEAP/deap
[6] https://www.bsc.es/marenostrum/marenostrum

space was also performed as a second benchmark, in order to compare the performance of the newly incorporated methods. The sweep search evaluates a sparser grid of points, uniformly distributed in the search space. However, points lying between consecutive ones on the uniform grid are not evaluated, and thus some interesting ones might be skipped.

To measure the performance of the clustering methods we use two metrics. These are (a) the number of uncertain points and (b) the total simulations performed until the execution terminates. Experiments were repeated five times using various random seeds for each method and configuration, allowing us to simulate the randomness of the process, while still being able to compare the methods in similar experimental conditions by calculating average result values. For the initialization of the RF classifier, 100 points were randomly selected for the first evaluation. The results for these points are used to compose the initial training set of the classifier but these simulations are not counted in the reported values. For each random seed, the initial set of points is identical for every clustering method under examination.

The set of the most uncertain points in each iteration of the AL part of the workflow defines the classification boundary of the classifier. Thus, their number can be used as a metric for monitoring performance, since it provides an estimate of the certainty of the classifier regarding the characterization of the parameter space. Consider the set of all treatment configurations $\mathcal{X}$, the classes $\{0, 1\}$ representing the non-viable and viable treatments respectively, and $Pr(i, x)$ the probability assigned by the classifier for a treatment configuration x to belong to class i. Then, the number of uncertain points N_U is computed based on the following rule:

$$N_U = \left|\left\{x \in \mathcal{X} \mid \min(Pr(0, x), Pr(1, x)) \geq \text{threshold}\right\}\right|$$

i.e. the points whose uncertainty is above a predefined threshold. We chose the uncertainty threshold to be equal to 0.4. The number of uncertain points is measured at the beginning of each iteration. The clustering method used in the second part of the sampling process aims at the selection of the most representative points in the set N_U. A large number of uncertain points at the end of an experiment reveals a weaker performance of the algorithm, as it signifies the existence of large ambiguous regions that cannot be classified with relative certainty as viable or non-viable.

Figure 2 depicts the number of uncertain points per iteration for each clustering method. The results suggest that, BIRCH and KMEANS_500 achieve the lowest number of uncertain points among the examined clustering methods. The performance of BIRCH is comparable to that of the benchmark KMEANS_500 while both are superior to the rest of the examined clustering methods. Moreover, DBSCAN and KMEANS_50 converge to similar numbers of uncertain points within the experimental time frame, with

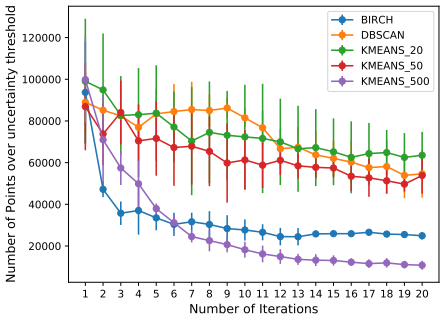


Figure 2: Number of uncertain points per iteration.

KMEANS_20 performing closely to the two other methods. Moreover, we note that BIRCH and KMEANS_500 performance is stable across the different experimental runs without any large variations as iterations progress.

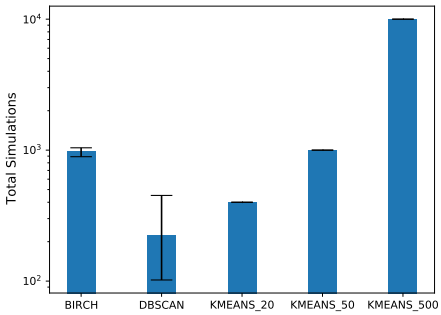


Figure 3: Total simulations performed per clustering method (y-axis in log-scale).

As we stressed earlier, the PhysiBoSSv2.0 simulations used to estimate the effectiveness of tumor treatments are time-consuming and require multiple CPUs for their execution. Thus, the resource-effectiveness of the examined methods is evaluated by counting the total number of simulations performed. This number can also be considered a good indicator of the time performance of each approach. Results are depicted in Figure 3. We observe that although BIRCH yields comparable results to the benchmark KMEANS_500, it requires significantly less simulations. In particular, the benchmark required 10,000 simulations and the BIRCH algorithm required only 982 on average, yielding approximately a 90% decrease. It is worth noting that DBSCAN required the fewest simulations from the examined methods, however it is not shown to be equally stable across the experiment repetitions as the increased variance in the results indicates. In particular, DBSCAN required only 223 simulations on average, while KMEANS_50 required 1000, and KMEANS_20 even fewer (400).

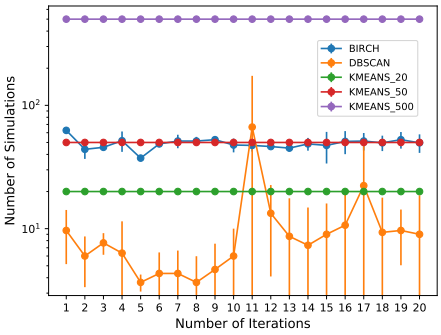


Figure 4: Simulations performed per iteration (y-axis in log-scale).

Figure 4 shows the number of simulations performed across the iterations of the AL part. In each iteration, a point from each resulting cluster is selected, and simulations are conducted for all selected points. Hence, the number of simulations required in each iteration is equal to the number of clusters identified by each clustering method. Although KMEANS_50 and BIRCH identify similar numbers of clusters, their performance differs substantially as illustrated in Figure 2. This is because the incorporation of outlying points into the clusters degrades the results of the clustering process of KMEANS_50, leading to the identification of distorted clusters for the candidate points. On the contrary, BIRCH recognizes outlying points and isolates them into smaller sub-clusters, thus keeping the ones that consist either of informative points that play an important role in the definition of the classification boundary, or of outlying points. Here, we can also observe increased volatility in the resulting clusters identified by DBSCAN, which reveals its high sensitivity to hyperparameters' values selection and to the density of the distribution of the candidate points.

Figure 5 provides a visual representation of the parameter space characterization for each clustering method used in the experiments, initiated with the same random seed for each method. The figure also includes results from the uninformed sweep search, that predetermines a relatively sparse number of points, and evaluates them exhaustively. We observe that all methods succeed in characterising the general region of the viable treatments. In particular, BIRCH, KMEANS_20 and KMEANS_50 achieve a space characterization that is very similar to the one obtained by the benchmarks KMEANS_500 and Sweep Search. Moreover, the low number of uncertain points achieved by BIRCH is evident in the visual representation of the space characterization, as BIRCH identifies an interesting region with smoother borders than KMEANS_20 and KMEANS_50. The smoothness of the borders indicates the certainty of the classifier regarding the boundaries of the region. On the contrary, DBSCAN identifies a smaller space as the interesting region, failing to capture the details of the edges of the actual area that includes the viable treatments.

Table 2: Results of the optimized drug treatment configuration discovery and consumed resources.

Method	Tumor Cell Survival Rate	Final Tumor Cell Count	Simulations	CPUs
GA	0.166	189	811	30
SA	0.195	223	91	5

4.2 Performance of treatment optimization

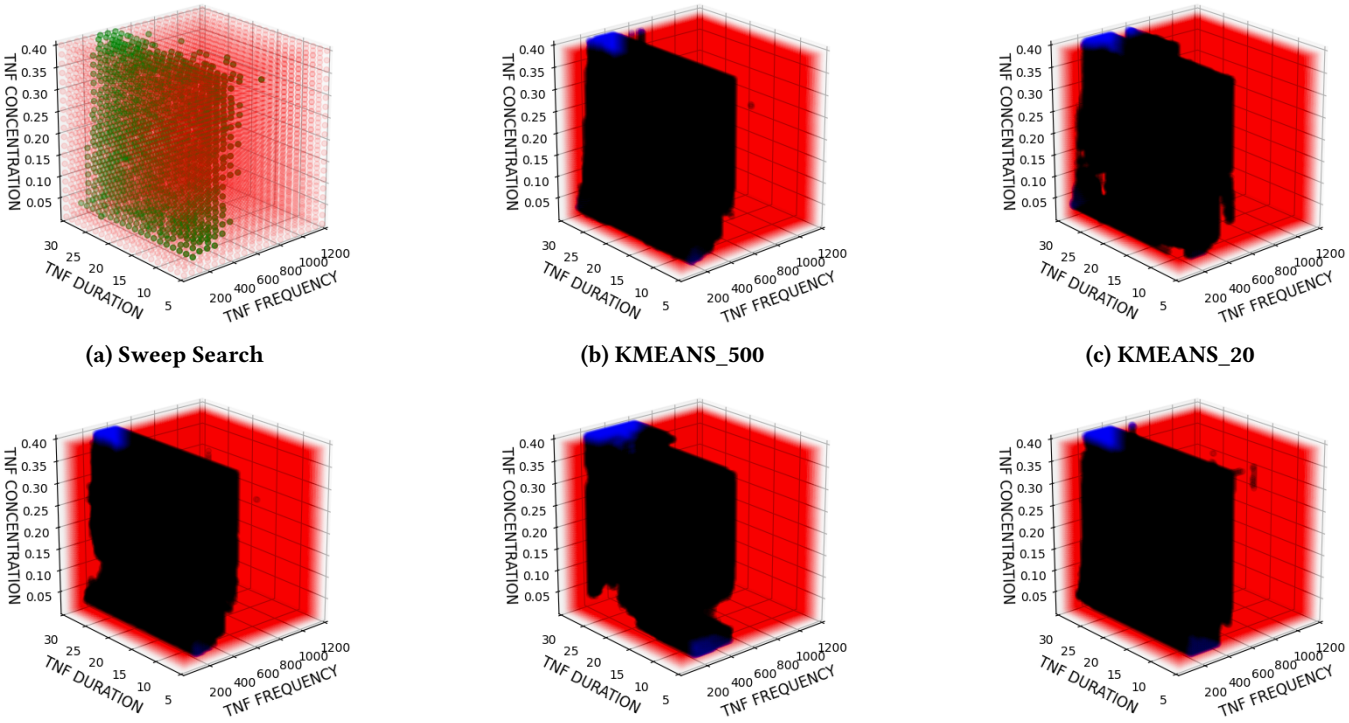
As described earlier, GAs have been used in the past in order to discover the most promising treatment configurations. In our evaluation, the GA was configured to 30 generations, with a population of 50 individuals, a tournament selection with tournament size equal to 3, a uniform crossover with crossover probability 0.75 and a mutation probability 0.5.

The alternative method examined, SA, was configured with T_o=100, T_{min}=15 and N=10. A geometrical cooling schedule with a cooling factor equal to $a = 0.8$ was applied, in which at each temperature level i the new temperature T_i is calculated as: $T_i = a^i \cdot T_o$. We examine the scenario in which both methods are initialized using information from the AL part using the BIRCH clustering algorithm in the sampling process, as this can be considered to provide a better balance between resources spent and discovered treatment efficiency. The initial population of the GA consists of 12 individuals (25% of the total population that the GA evaluates) found to be the most promising by the active learning, as well as of 38 random treatments (75% of the GA population). The initial point of the SA is set to be the most promising one discovered by the AL.

Table 2 presents the results of the two examined search methods. The GA discovers a treatment that leads to 189 alive tumor cells after administration, while the treatment discovered by the SA method leads to 223 alive cells. Although GA discovers a more efficient treatment, it requires noticeably more resources than SA. In particular, GA and SA require 811 and 91 simulations, respectively. Hence, SA requires approximately 12% of the simulations initiated by the GA, making it less demanding in runtime, nevertheless without inducing large losses in the quality of the final results. Moreover, the utilization of only one candidate solution makes SA less demanding in core processing units, in contrast to the pool of 50 candidate treatments utilized by the GA.

4.3 Results overview

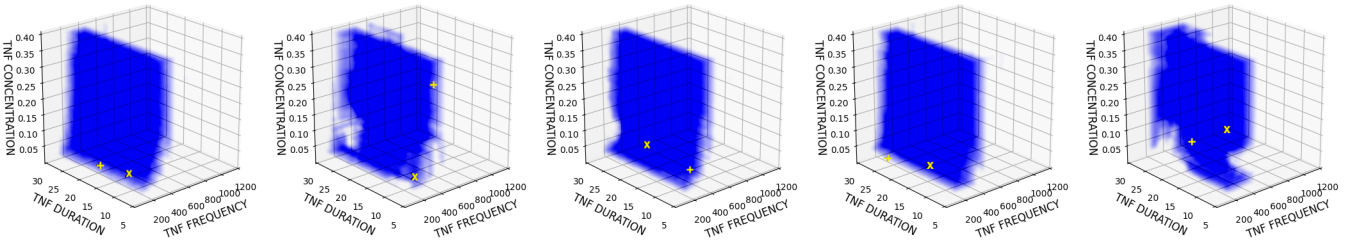
Finally, we present the combined results of both parts of the incorporated workflow. As Figure 6 illustrates, all versions



(a) Sweep Search (b) KMEANS_500 (c) KMEANS_20

(d) KMEANS_50 (e) DBSCAN (f) BIRCH

Figure 5: Visual representation of the viable regions of the parameter space. Subfigure (a) depicts the results of the sweep search of the parameter space and subfigure (b) represents the results of the benchmark KMEANS_500. Subfigures (c), (d), (e), (f) illustrate the results of each additional method examined.



(a) KMEANS_500 (b) KMEANS_20 (c) KMEANS_50 (d) BIRCH (e) DBSCAN

Figure 6: Visual representation of the results when incorporating different clustering and optimization algorithms. Treatment configurations marked with 'x' and '+' represent the optimal ones discovered by GA and SA, respectively. The area shaded with blue represents the regions that are characterized as interesting by the AL part of the workflow.

manage to converge to similar interesting regions. Specifically for KMEANS_20 and DBSCAN, there are some parts of the interesting region that are missed, if we also compare the results of the benchmark (KMEANS_500). Moreover, the optimization methods discover effective treatment configurations (yellow 'x' and '+' marks) that indeed lie inside, or at the border of the designated interesting areas (blue). Regardless of the clustering algorithm used in the AL part, the optimized treatment configurations that are discovered by the GA achieve a tumor cell survival rate of less than 0.175, while when using the SA this rate ranges between 0.209 and 0.177, that is lower than the 0.3 threshold in all cases. These

results may also provide a comprehensive visualization to experts, which can be utilized in order to further guide drug discovery research and real world trials.

5 CONCLUSIONS & FURTHER WORK

Multi-scale simulations provide a valuable tool to researchers for various fields and applications, including the discovery of promising tumor treatments. We applied a multi-scale simulator in order to characterize the treatment configuration space and identify the most effective treatments, combined with an Active Learning workflow for space exploration. We used a new simulation model imitating the application of the protein called Tumor Necrosis Factor (TNF) in order to estimate the effect of tumor treatments. Moreover, we examined the performance of the effect of various well-known clustering algorithms in the sampling process of the parameter space exploration, as well as the performance of the Genetic Algorithm and Simulated Annealing methods in the discovery of optimized drug treatment configurations. Simulation trials conducted in an HPC environment show that the BIRCH and KMEANS_50 clustering algorithms achieve a high quality characterization of the parameter space similar to the one obtained by the benchmark methods. BIRCH leads to the least number of interim uncertain points and identifies the region of viable treatments with the highest certainty. Moreover, the application of the GA leads to the discovery of a slightly more effective treatment than the SA. However, GA requires more computational resources and simulations than the SA. The experimental results indicate that a trade-off between the required resources and the quality of the results is evident in both parts of the workflow.

In the future, we plan to employ additional classification algorithms in the active learning workflow, such as the Gradient Boosting Trees [22], and to compare their performance with the RF classifier. Moreover, we plan to incorporate interactive tools in order to allow the researcher to better control the exploration process, by explicitly selecting parameter values to evaluate throughout the iterations. The workflow will also be extended to explore the synergistic effects of multiple drugs administration in more complex simulations.

6 ACKNOWLEDGMENTS

This work has received funding from the EU Horizon 2020 RIA program INFORE under grant agreement No 825070.

REFERENCES

[1] C. C. Aggarwal. 2015. *Data Mining - The Textbook.* Springer.
[2] C. Akasiadis, M. Ponce-de Leon, A. Montagud, E. Michelioudakis, A. Atsidakou, E. Alevizos, A. Artikis, A. Valencia, and G. Paliouras. 2022. Parallel Model Exploration for Tumor Treatment Simulations. *Computational Intelligence* (2022), 1–23.
[3] A. Ambesi-Impiombato and D. Bernardo. 2006. Computational Biology and Drug Discovery: From Single-Target to Network Drugs. *Current Bioinformatics* 1, 1 (2006), 3–13.
[4] P. Baldi, S. Brunak, and F. Bach. 2001. *Bioinformatics: the machine learning approach.* MIT press.
[5] J. Błażewicz, P. Łukasiak, and M. Miłostan. 2005. Application of tabu search strategy for finding low energy structure of protein. *Artificial Intelligence in Medicine* 35, 1 (2005), 135–145.
[6] V. Černý. 1985. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications* 45, 1 (1985), 41–51.
[7] W.-M. Chu. 2013. Tumor necrosis factor. *Cancer Letters* 328, 2 (2013), 222–225.
[8] R. Dias and A. Torkamani. 2019. Artificial intelligence in clinical and genomic diagnostics. *Genome Medicine* 11, 1 (2019).
[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining.* AAAI Press, 226–231.
[10] A. Ghaffarizadeh, R. Heiland, S. H. Friedman, S. M. Mumenthaler, and P. Macklin. 2018. PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. *PLOS Comp. Biology* 14, 2 (2018), 1–31.
[11] J. I. Griffiths, A. L. Cohen, V. Jones, R. Salgia, J. T. Chang, and A. H. Bild. 2019. Opportunities for improving cancer treatment using systems biology. *Current Opinion in Systems Biology* 17 (2019), 41–50.
[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680.
[13] N. Kutsuna, T. Higaki, S. Matsunaga, T. Otsuki, M. Yamaguchi, H. Fujii, and S. Hasezawa. 2012. Active learning framework with iterative clustering for bioimage classification. *Nature comm.* 3, 1 (2012), 1–10.
[14] S. P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (1982), 129–136.
[15] A. K. Mitra, V. Agrahari, A. Mandal, K. Cholkar, C. Natarajan, S. Shah, M. Joseph, H. M. Trinh, R. Vaishya, X. Yang, Y. Hao, V. Khurana, and D. Pal. 2015. Novel delivery approaches for cancer therapeutics. *Journal of Controlled Release* 219 (2015), 248–268.
[16] T. P. Mohamed, J. G. Carbonell, and M. K. Ganapathiraju. 2010. Active learning for human protein-protein interaction prediction. *BMC bioinformatics* 11, 1 (2010), 1–9.
[17] R. B. Mokhtari, T. S. Homayouni, N. Baluch, E. Morgatskaya, S. Kumar, B. Das, and H. Yeger. 2017. Combination therapy in combating cancer. *Oncotarget* 8, 23 (2017), 38022–38043.
[18] M. Niu, C. Wan, and Z. Xu. 2014. A review on applications of heuristic optimization algorithms for optimal power flow in modern power systems. *J. of Mod. Power Sys. and Clean Energy* 2, 4 (2014), 289–297.
[19] J. Ozik, N. Collier, R. Heiland, G. An, and P. Macklin. 2019. Learning-accelerated discovery of immune-tumour interactions. *Mol. Syst. Des. Eng.* 4 (2019), 747–760. Issue 4.
[20] D. Reker and G. Schneider. 2015. Active-learning strategies in computer-assisted drug discovery. *Drug discovery today* 20, 4 (2015), 458–465.
[21] S. Russell and P. Norvig. 2010. *Artificial Intelligence: A Modern Approach* (3 ed.). Prentice Hall.
[22] R. E. Schapire. 2003. *The Boosting Approach to Machine Learning: An Overview.* Springer, 149–171.
[23] B. Settles. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.
[24] D. Xu and Y. Tian. 2015. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science* 2, 2 (2015), 165–193.
[25] T. Zhang, R. Ramakrishnan, and M. Livny. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD Rec.* 25, 2 (1996), 103–114.