Adaptive Reference Update (ARU) Algorithm: A Stochastic Search Algorithm for Efficient Optimization of Multi-Drug Cocktails

Mansuck Kim[1] and Byung-Jun Yoon*[1]

[1] Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128, USA

Email: Mansuck Kim - mk22893@neo.tamu.edu; Byung-Jun Yoon*- bjyoon@ece.tamu.edu;

*Corresponding author

Abstract

Background: Multi-target therapeutics has been shown to be effective for treating complex diseases, and currently, it is a common practice to combine multiple drugs to treat such diseases to optimize the therapeutic outcomes. However, considering the huge number of possible ways to mix multiple drugs at different concentrations, it is practically difficult to identify the optimal drug combination through exhaustive testing.

Results: In this paper, we propose a novel stochastic search algorithm, called the adaptive reference update (ARU) algorithm, that can provide an efficient and systematic way for optimizing multi-drug cocktails. The ARU algorithm iteratively updates the drug combination to improve its response, where the update is made by comparing the response of the current combination with that of a reference combination, based on which the beneficial update direction is predicted. The reference combination is continuously updated based on the drug response values observed in the past, thereby adapting to the underlying drug response function. To demonstrate the effectiveness of the proposed algorithm, we evaluated its performance based on various multi-dimensional drug functions and compared it with existing algorithms.

Conclusions: Simulation results show that the ARU algorithm significantly outperforms existing stochastic search algorithms, including the Gur Game algorithm. In fact, the ARU algorithm can more effectively identify potent drug combinations and it typically spends fewer iterations for finding effective combinations. Furthermore, the ARU algorithm is robust to random fluctuations and noise in the measured drug response, which makes the algorithm well-suited for practical drug optimization applications.

Background

Biological networks are known to be redundant at various levels, which makes them robust to various types of perturbations. As a consequence, it is generally difficult to change their long-term dynamics by blocking a specific gene or intervening in a specific pathway. This is one of the reasons why monotherapy is often not very effective in treating complex diseases, such as cancer and diabetes. In fact, multi-target therapeutics based on combinatory drugs are known to be much more effective, and they are commonly used these days for treating various diseases [1–6]. However, considering the huge number of possible ways to mix multiple drugs, it is practically impossible to find the optimal "drug cocktail" simply by trial and error or by exhaustive testing. Clearly, we need a systematic way of identifying the most effective drug cocktail, and recently, several algorithms have been proposed to address the problem of combinatorial drug optimization [7–12].

For example, Calzolari et al. [7] developed a drug optimization algorithm based on sequential decoding algorithms that have been traditionally used in digital communications [13, 14]. In [7], it was shown that we can algorithmically identify the optimal drug combination by testing only a relatively small number of drug combinations, compared to exhaustive search. Unlike the approach proposed by Calzolari et al. [7], which was deterministic, Wong et al. [9] proposed a different approach based on a stochastic search algorithm, called the Gur Game algorithm [15, 16]. In this work [9], they formed a closed-loop optimization framework, in which the Gur Game algorithm was used to predict an updated drug combination that is likely to improve the current drug response, and the drug combination is iteratively updated until the response is maximized. It was shown that this closed-loop optimization method can quickly identify potent drug combinations. More recently, another stochastic search algorithm was proposed in [11] that addresses the limitations of the Gur Game algorithm, thereby further improving the performance of the closed-loop optimization approach originally proposed in [9].

In this paper, we propose a novel stochastic search algorithm, called the adaptive reference update (ARU) algorithm, that can significantly improve the performance of the existing stochastic search algorithms [9, 11]. The key idea of this algorithm is to adaptively update the reference drug combination to guide the search algorithm and help it to make better informed guesses without requiring extensive prior knowledge of the underlying biological network. We demonstrate that the proposed ARU algorithm outperforms existing stochastic drug optimization algorithms, in terms of both efficiency, success rate, and robustness.

Methods

Combinatorial drug optimization problem

Suppose we have N different types of drugs, where each drug can take one of pre-specified concentrations. Our goal is to predict the optimal drug cocktail, by mixing the available drugs, that maximizes the overall therapeutic effect. Let x_n be the concentration of the n-th drug, where x_n can take one of M_n distinct concentrations in the set $\mathcal{C}_n = \{c_n^1, c_n^2, c_n^3, \cdots, c_n^{M_n}\}$, where $c_n^k < c_n^{k+1}$ for all k. The drug cocktail is represented by an N-dimensional vector $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, which consists of the N drug concentrations. Let $f(\mathbf{x})$ be the normalized drug response that quantitatively measures the effectiveness of a given drug combination $\mathbf{x}$. We assume the response has been normalized such that $0 \leq f(\mathbf{x}) \leq 1$ for $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_N$ is the set of all possible drug combinations. We denote the number of all possible drug combinations as $M = |\mathcal{X}| = \prod_{n=1}^{N} M_n$. $f(\mathbf{x}) = 0$ implies that the drug cocktail $\mathbf{x}$ is completely ineffective, and larger $f(\mathbf{x})$ implies higher efficacy. In practical applications, $f(\mathbf{x})$ may be obtained by monitoring the cell response to a drug intervention using fluorescent imaging, microarrays, or sequencing techniques. Under this setting, we aim to find the optimal drug combination $\mathbf{x}^*$ that maximizes the normalized drug response:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}).$$

As we can see, this is a combinatorial optimization problem, in which we have to find the optimal drug combination out of $M_1 M_2 \cdots M_N$ possible combinations. The total number of distinct drug combinations quickly grows as the number of drugs increases. Considering the practical cost of experimentally measuring the normalized drug response function $f(\mathbf{x})$, it is apparent that we cannot test all drug combinations to find the most effective one.

Stochastic search algorithms

Stochastic search algorithms [9, 11] aim to efficiently identify the potent drug combinations without exploring the entire combinatorial solution space. The basic idea is to randomly search through the solution space by iteratively updating the drug combination until an effective combination emerges. At each step, the current drug combination is incrementally updated towards the direction that is likely to improve the overall drug response. The updating decision is made in a stochastic manner, which allows the search to proceed towards directions that are deemed to be less likely to improve the response. This is an important characteristic of stochastic search algorithms, which is critical for keeping the search from being trapped in local maxima. Since a stochastic search algorithm tries to arrive at the optimal solution (i.e.,

the most effective drug combination) by performing iterative local searches, its overall performance depends on how it chooses the next solution state (i.e., an updated drug combination in $\mathcal{X}$, the set of all possible combinations) from a given state (i.e., the current drug combination). The two performance metrics of interest are: (i) the effectiveness of the predicted drug combination, in terms of how close its response is to the optimal response, and (ii) the number of search steps that the algorithm needs to take until an effective combination is found. Basically, we want to predict a potent drug cocktail by testing minimal number of drug combinations to minimize the actual experimental cost for measuring the cell response to combinatorial drugs. When choosing the next state, the search algorithm has to be as parsimonious as possible, in terms of the number of function evaluations, so that the overall experimental cost for identifying the optimal drug combination can be minimized. This has been one of the main design considerations of existing stochastic search algorithms that have been developed for combinatorial drug optimization [9, 11].

For example, the Gur Game algorithm adopted in [9] determines how to update the drug combination solely based on the current drug response. Suppose $\mathbf{x}^c = (x_1^c, x_2^c, \cdots, x_N^c)$ is the current drug combination with a normalized drug response of $f(\mathbf{x}^c)$. The algorithm generates N random numbers $r_n \in [0, 1]$, one for each drug. Each r_n is compared against the current drug response $\mathbf{x}^c = (x_1^c, x_2^c, \cdots, x_N^c)$, which is used to either "reward" or "penalize" the n-th drug. For example, the drug is rewarded if $f(\mathbf{x}^c) > r_n$. Otherwise, it is penalized. This update process is repeated for each of the N drugs. According to this scheme, drug combinations with higher response are more likely to be rewarded, while combinations with lower response are more likely to be penalized. In the long run, the algorithm is expected to drive the concentration of each drug towards the one that maximizes the response. Now, an important relevant question is what is the right way of rewarding (or penalizing) the current concentration of a given drug. The Gur Game algorithm used in [9] uses a predetermined finite state automaton (FSA) for this purpose. For example, let $x_n^c = c_n^k \in \mathcal{C}_n$ be the current concentration of the n-th drug. Suppose we have $f(\mathbf{x}^c) > r_n$, hence the current concentration x_n^c of the n-th drug should be rewarded. The FSA in [9] is designed such that the drug concentration is increased further if the current concentration x_n^c is larger than a reference concentration c_n^{ref}, and decreased further if x_n^c is smaller than c_n^{ref}. More specifically, if $x_n^c = c_n^k$ and

$f(\mathbf{x}^c) > r_n$, then the drug concentration is updated as follows:

$$x_n^c = \begin{cases} c_n^{k+1}, & \text{if } x_n^c > c_n^{\text{ref}} \text{ and } k < M_n \\ c_n^k, & \text{if } x_n^c > c_n^{\text{ref}} \text{ and } k = M_n \\ c_n^{k-1}, & \text{if } x_n^c < c_n^{\text{ref}} \text{ and } k > 1 \\ c_n^k, & \text{if } x_n^c < c_n^{\text{ref}} \text{ and } k = 1 \end{cases} \tag{1}$$

The reference concentration c_n^{ref} is typically chosen as the median of the set $\mathcal{C}_n$. As shown in (1), the drug concentration remains unchanged if it cannot be increased (or decreased) further. Now, suppose that $f(\mathbf{x}^c) < r_n$, and therefore the current concentration $x_n^c = c_n^k$ should be penalized. In this case, the concentration is updated in the opposite direction:

$$x_n^c = \begin{cases} c_n^{k-1}, & \text{if } x_n^c > c_n^{\text{ref}} \\ c_n^{k+1}, & \text{if } x_n^c < c_n^{\text{ref}} \end{cases} \tag{2}$$

Note that penalization moves the current drug concentration closer to the reference concentration c_n^{ref}.

As previously discussed in [11], one of the weaknesses of the Gur Game algorithm is that it uses a predetermined FSA for updating (i.e., rewarding/penalizing) the drug concentrations and does not adapt to the drug response function at hand, which is not known in advance. As a result, the algorithm may perform poorly unless the drug response function $f(\mathbf{x})$ is properly normalized and the reference concentration c_n^{ref} is chosen adequately for each drug. For example, consider the one-dimensional drug response $f(x)$ shown in Figure 1A. As we can see, the drug response has been over-normalized, hence $f(x) < 0.5$ for any allowed concentration $x \in [c_{\min}, c_{\max}]$. Since $f(x) < 0.5$, a uniformly distributed random number $r \in [0, 1]$ is more likely to be larger than $f(x)$. This implies that the Gur Game algorithm always tends to penalize the current drug concentration (no matter what its value is), which will probabilistically drive the concentration towards c_{ref} although it is clearly not optimal. Figure 1B shows another drug response, for which the Gur Game algorithm will not work properly. In this example, we have $f(x) > 0.5$, hence the Gur Game algorithm is always more likely to reward the current drug concentration, which tends to drive the concentration away from the reference concentration c_{ref}. This will push the concentration either towards $c_{\min}$ or $c_{\max}$, both of which are suboptimal.

The enhanced stochastic algorithm proposed in [11] addresses this problem by making the search algorithm adapt to a given drug response. The basic idea of this algorithm is to make an "informed-guess" about how to beneficially update a given drug concentration, instead of following a predetermined update rule. Unlike the Gur Game algorithm in [9], where all N drugs are simultaneously updated based on the (same) current drug response $f(\mathbf{x}^c)$, the enhanced algorithm updates the concentration of one drug at a time. As an

example, suppose during the last update of the n-th drug, the drug combination has been updated as

$$\mathbf{x} = (x_1, \cdots, \cdots, x_N) \implies \mathbf{x}' = (x'_1, \cdots, \cdots, x'_N),$$

where $\mathbf{x}$ and $\mathbf{x}'$ are identical except for the concentration of the n-th drug, hence $x_i = x'_i$ $(i \neq n)$ and $x_i \neq x'_i$ $(i = n)$. We assume that x_n and x'_n differ only by a single concentration level, so that $x_n = c_n^k$ and $x'_n = c_n^{k+1}$, or $x_n = c_n^{k+1}$ and $x'_n = c_n^k$, for some k. The algorithm compares the two drug responses $f(\mathbf{x})$ and $f(\mathbf{x}')$, thereby determine wether it would be more beneficial to further increase or decrease the concentration of the n-th drug. For example, we may have the following four cases:

$$\begin{aligned}
&\text{(Case-1)} \quad x_n < x'_n \text{ and } f(\mathbf{x}) < f(\mathbf{x}') \quad : \textbf{increasing} \text{ the concentration is more beneficial} \\
&\text{(Case-2)} \quad x_n > x'_n \text{ and } f(\mathbf{x}) > f(\mathbf{x}') \quad : \textbf{increasing} \text{ the concentration is more beneficial} \\
&\text{(Case-3)} \quad x_n < x'_n \text{ and } f(\mathbf{x}) > f(\mathbf{x}') \quad : \textbf{decreasing} \text{ the concentration is more beneficial} \\
&\text{(Case-4)} \quad x_n > x'_n \text{ and } f(\mathbf{x}) < f(\mathbf{x}') \quad : \textbf{decreasing} \text{ the concentration is more beneficial.}
\end{aligned} \tag{3}$$

The above rules allow the algorithm to adaptively determine *how* to reward (or penalize) a given drug concentration based on the observed drug response values. However, the decision *whether* to reward or penalize the current drug is made in a probabilistic manner. For this purpose, we evaluate the following function

$$g(\mathbf{x}, \mathbf{x}') = \frac{1}{2}\left\{1 + \alpha \cdot \max\left[f(\mathbf{x}), f(\mathbf{x}')\right]\right\}, \tag{4}$$

where $\alpha \in [0, 1]$ is a control parameter that adjusts the randomness of the algorithm [11]. This $g(\mathbf{x}, \mathbf{x}')$ is compared with a uniformly distributed random number $r_n \in [0, 1]$. If $g(\mathbf{x}, \mathbf{x}') > r_n$, the n-th drug is rewarded, i.e., updated in such a way that appears to be more beneficial for enhancing the drug response according to the rules shown in (3). Otherwise, the n-th drug is penalized, i.e., updated in a way that appears to be less beneficial based on the past observations. It is not difficult to see that this algorithm is always more likely to reward, or beneficially update, a given drug. Since the algorithm proposed in [11] adaptively determines how to update the drug concentration based on previous observations, it can also effectively deal with drug response functions shown in Figures 1A and 1B, for which the Gur Game algorithm does not perform well. Despite its merits, this algorithm also has its own shortcomings. For example, as the update rule for a given drug is determined only based on the two observations that correspond to its latest update, not on a longer-range trend, the algorithm may be sensitive to small variations in the drug response. As a result, it may not show satisfactory search performance for drug response functions that are similar to the one in Figure 1C. Furthermore, considering that $f(\mathbf{x})$ has to be experimentally estimated, where a certain level of measurement noise and small variations due to a number

of practical factors may not be avoidable, such sensitivity may adversely affect the overall performance of the algorithm. Another weakness of the algorithm is that it only utilizes a very small part of the past observations without fully utilizing them. In the following section, we introduce a novel stochastic search algorithm that can effectively address the aforementioned issues.

The adaptive reference update (ARU) stochastic search algorithm

In order to make the search algorithm robust to small variations in the observed drug response, the update rules have to be decided based on the general trend of the drug response change over a wide range of drug concentration, not just based on the response change resulting from a *single-level* concentration change. Based on this motivation, we propose a novel algorithm called the **adaptive reference update (ARU) algorithm**. In this algorithm, we compare the current drug response $f(\mathbf{x}^c)$ with the response $f(\mathbf{x}^{\text{ref}})$ of a *reference drug combination* $\mathbf{x}^{\text{ref}}$, which is adaptively updated based on past observations. In the beginning, $\mathbf{x}^{\text{ref}}$ is set to the initial drug concentration, where we start the search process. During the search, when the algorithm encounters a local maximum, the current reference combination is replaced by the corresponding drug combination. As an example, let us consider the one-dimensional drug response function $f(x)$ in Figure 2. For illustration, we consider the following hypothetical search process, where the drug concentration is constantly updated from left to right, starting from the lowest concentration $c_{\min}$ to the highest concentration $c_{\max}$. Suppose the search begins at the concentration $x = c_{\min}$. Initially the reference concentration is also set to this initial drug concentration $x^{\text{ref}} \leftarrow c_{\min}$. As the search reaches the first local maximum, the reference concentration is updated to this local maximum solution $x^{\text{ref}} \leftarrow x^{\text{ref1}}$. As the search continues to the right, this reference concentration is used until we reach the next local maximum. After passing the second local maximum solution x^{ref2}, the reference point is updated to $x^{\text{ref}} \leftarrow x^{\text{ref2}}$. In a similar manner, as the search continues further to the right, the reference point is updated to $x^{\text{ref}} \leftarrow x^{\text{ref3}}$ after passing the third local maximum point.

At each iteration, the current drug response $f(\mathbf{x}^c)$ is compared to the response $f(\mathbf{x}^{\text{ref}})$ of the reference drug combination, based on which the drug update rule is determined. For example, let $\mathbf{x}^c = (\,\cdots, x_n^c, \cdots)$ and $\mathbf{x}^{\text{ref}} = (\cdots, x_n^{\text{ref}}, \cdots)$, and assume that we want to update the concentration of the n-th drug by comparing

the two drug response values $f(\mathbf{x}^c)$ and $f(\mathbf{x}^{\text{ref}})$. As before, we have the following four possible cases:

$$\begin{aligned}
&\text{(Case-1)} \quad x_n^c < x_n^{\text{ref}} \text{ and } f(\mathbf{x}^c) < f(\mathbf{x}^{\text{ref}}) \quad : \textbf{increasing} \text{ the concentration is more beneficial} \\
&\text{(Case-2)} \quad x_n^c > x_n^{\text{ref}} \text{ and } f(\mathbf{x}^c) > f(\mathbf{x}^{\text{ref}}) \quad : \textbf{increasing} \text{ the concentration is more beneficial} \\
&\text{(Case-3)} \quad x_n^c < x_n^{\text{ref}} \text{ and } f(\mathbf{x}^c) > f(\mathbf{x}^{\text{ref}}) \quad : \textbf{decreasing} \text{ the concentration is more beneficial} \\
&\text{(Case-4)} \quad x_n^c > x_n^{\text{ref}} \text{ and } f(\mathbf{x}^c) < f(\mathbf{x}^{\text{ref}}) \quad : \textbf{decreasing} \text{ the concentration is more beneficial.}
\end{aligned} \tag{5}$$

Conceptually, we can view the above as estimating the "virtual" slope between two points $(x_n^c, f(\mathbf{x}^c))$ and $(x_n^{\text{ref}}, f(\mathbf{x}^{\text{ref}}))$ as follows

$$\frac{f(\mathbf{x}^{\text{ref}}) - f(\mathbf{x}^c)}{x_n^{\text{ref}} - x_n^c}, \tag{6}$$

based on which we determine how to update the concentration x_n of the n-th drug to increase the drug response $f(\mathbf{x})$. Given the update rules in (5), the actual update decision is made by evaluating the following function

$$g(\mathbf{x}^c, \mathbf{x}^{\text{ref}}) = \frac{1}{2}\left\{1 + \alpha \cdot \max\left[f(\mathbf{x}^c), f(\mathbf{x}^{\text{ref}})\right]\right\} \tag{7}$$

and comparing it with a random number $r_n \in [0, 1]$. If $g(\mathbf{x}^c, \mathbf{x}^{\text{ref}}) > r_n$, the drug concentration x_n is updated by a single level, following the beneficial update direction predicted by (5). Otherwise, the concentration is updated in the opposite direction. As briefly mentioned before, the parameter $\alpha \in [0, 1]$ controls the randomness of the algorithm. For example, $\alpha = 0$ will make the search process completely random, regardless of the observed drug responses. Using a larger α means that we are giving a larger weight to the past observations when deciding how to update the drug concentrations. The value of this control parameter is limited to $\alpha \leq 1$ such that $g(\mathbf{x}^c, \mathbf{x}^{\text{ref}}) \leq 1$. Also note that we always have $g(\mathbf{x}^c, \mathbf{x}^{\text{ref}}) \geq 0.5$, which implies that at any drug update step, the update is always more likely to take place in accordance with the rules in (5), which have been derived based on past observations of the drug response. In other words, the ARU algorithm tries to effectively utilize the past response data to beneficially update the drug concentrations, and ultimately, to identify a potent drug combination, while keeping the search still stochastic. For illustration, let us again consider the drug response function in Figure 2, where the hypothetical search process proceeds from the lowest drug concentration to the highest concentration. The black solid arrows below the graph shows the drug update direction that gets higher probability according to the new algorithm, described above. For example, in region-A ($c_{\min} < x < x^{\text{ref1}}$), the algorithm tends to increase the drug concentration x further, as the response $f(x)$ is larger than $f(c_{\min})$ of the initial reference concentration (i.e., $c_{\min}$). As x continues to increase and passes the first local maximum point x^{ref1}, the reference is updated to $x^{\text{ref}} \leftarrow x^{\text{ref1}}$. In region-B ($x^{\text{ref1}} < x < x^{\text{ref2}}$), the

search algorithm tends to drive the concentration towards x^{ref1} by decreasing the concentration. Suppose the search continues to increase the drug concentration x beyond x^{ref2}, the second local maximum point, despite the tendency of the algorithm to decrease x back to x^{ref1}. After passing x^{ref2}, the reference is updated to $x^{\text{ref}} \leftarrow x^{\text{ref2}}$. In region-C, the search algorithms assigns higher probability to the update rule that tries to bring the concentration down to x^{ref2}, since $f(x) < f(x^{\text{ref2}})$ in the given region. However, once x enters region-D, where $f(x) > f(x^{\text{ref2}})$, the algorithm begins to drive the drug concentration x further to the right until it passes the third local maximum point x^{ref3}. The reference concentration is updated to $x^{\text{ref}} \leftarrow x^{\text{ref3}}$, once the search continues to the right and the drug concentration x gets larger than x^{ref3}. Since $f(x^{\text{ref3}})$ is larger than $f(x)$ in region-D ($x^{\text{ref3}} < x < c_{\max}$), the search algorithm will tend to bring the concentration down to the current reference concentration, namely, $x^{\text{ref}} = x^{\text{ref3}}$.

Choosing a local maximum solution as a reference combination has a number of practical advantages. First of all, it allows the algorithm to adjust the drug update rules based on a long-range trend of the given drug response function, which makes the algorithm robust to small variations in the observed response. Another advantage of using a long-range trend is that the search process will become also less sensitive to random fluctuations that may exist in the observed drug response. Considering that the drug response function $f(\mathbf{x})$ has to be experimentally estimated through actual biological experiments, where random factors (e.g., measurement noise) that affect the estimation results cannot be completely ruled out, such robustness is critical for the algorithm to be used in practical drug optimization applications. It is also beneficial to use the drug combination that corresponds to the most recent local maximum response, instead of the drug combination that has yielded the highest response among all past combinations, as the reference point. This prevents the search process from dwelling too much on past observations, while keeping it robust to insignificant variations and random fluctuations.

Drug response functions

In order to evaluate the overall performance of the ARU algorithm, we used the algorithm to search for the optimal drug cocktail for several different drug response functions.

Two-dimensional drug response functions For performance assessment, we first used the four two-dimensional drug response functions that are shown in Figure 3. The first drug response function $f_{2a}(x_1, x_2)$ shown in Figure 3A is the normalized HIV inhibitor response obtained from [17], where $x_1 \in \{0, 0.01, 0.03, 0.09, 0.27, 0.82, 2.47, 7.41, 22.22, 66.67\}(nM)$ was considered for Maraviroc and $x_2 \in \{0, 0.09, 0.27, 0.8, 2.41, 7.22, 21.67, 65\}(nM)$ for ROAb14. The second drug response $f_{2b}(x_1, x_2)$ shown

in Figure 3B is the normalized second De Jong function (Rosenbrock's saddle) [18]. We considered $x_1, x_2 \in \{c_0, c_1, \cdots, c_{20}\}$, where $c_k = 4(k/20 - 0.5)$, obtained by evenly dividing the range $[-2, 2]$ into 21 distinct values. The third drug response function $f_{2c}(x_1, x_2)$ in Figure 3C is the normalized lung cancer inhibition response obtained from [1], where $x_1 \in \{0, 1, 2, 4, 6, 8, 12, 16, 20, 22\}(\mu M)$ was considered for Chlorpromazine and $x_2 \in \{0, 0.25, 0.4, 0.6, 0.8, 1, 1.5, 2, 4, 6.8\}(\mu M)$ for Pentamidine. Finally, Figure 3D shows the fourth response function $f_{2d}(x_1, x_2)$, the normalized bacterial (*S. aureus*) inhibition response reported in [6]. $x_1 \in \{0, 0.08, 0.16, 0.32, 0.63, 1.25, 2.5, 5, 10\}$ was considered for Trimethoprim and $x_2 \in \{0, 0.31, 0.62, 1.25, 2.5, 5, 10, 20, 40\}$ was considered for Sulfamethoxazole. All four drug response functions were normalized to span the entire range $[0, 1]$, such that the minimum response is 0 and the maximum response is 1.

Multi-dimensional drug response functions To evaluate the performance for optimizing multi-drug cocktails, we defined several hypothetical drug response functions with up to six drugs. First, we defined two 3-dimensional drug functions $f_{3a}(x_1, x_2, x_3)$ and $f_{3b}(x_1, x_2, x_3)$. The first function is defined as

$$f_{3a}(x_1, x_2, x_3) = x_1^2 \cdot \sin^2(x_2) \cdot \cos^2(x_3), \tag{8}$$

where each of x_1, x_2, x_3 can take one of the 11 discrete concentrations that evenly divide the range $[-2.5, 2.5]$. The second function is defined as follows

$$f_{3b}(x_1, x_2, x_3) = \texttt{peaks}(x_1, x_2) \cdot x_3, \tag{9}$$

using the Matlab $\texttt{peaks}(x_1, x_2)$ function. We assume that each drug can take one of the 11 discrete values that evenly divide $[-3, 3]$. Next, we defined two 4-dimensional drug functions

$$f_{4a}(x_1, x_2, x_3, x_4) = x_1 \cdot e^{-(x_1^2 + x_2^2 + x_3^2 + x_4^2)} \tag{10}$$

and

$$f_{4b}(x_1, x_2, x_3, x_4) = \cos^2(0.3x_1) \cdot \sin(0.3x_2) \cdot \tan(0.1x_3) \cdot x_4. \tag{11}$$

We assume that x_1 and x_2 in the first function $f_{4a}(x_1, x_2, x_3, x_4)$ can take one of the 11 discrete values that evenly divide the range $[-2, 2]$, while x_3 and x_4 can take one of the 11 discrete values that evenly divide the range $[-3, 3]$. For the second drug response function $f_{4b}(x_1, x_2, x_3, x_4)$, we assume that each drug can take one of the 11 distinct values that evenly divide $[-3, 3]$. In addition, we also defined the following 5-dimensional drug response functions

$$f_{5a}(x_1, x_2, x_3, x_4, x_5) = e^{-x_1} \cdot \cos^2(x_2) \cdot x_3^2 \cdot \left[e^{-(x_4+2)^2 - (x_5+3)^2} + e^{-(x_4-2)^2 - (x_5-3)^2} \right] \tag{12}$$

and

$$f_{5b}(x_1, x_2, x_3, x_4, x_5) = \frac{1}{2}\,\texttt{peaks}(x_1, x_2) \cdot \cos(0.5x_3) \cdot \sin(0.5x_4) \cdot x_5^2. \tag{13}$$

For the first function $f_{5a}(x_1, x_2, x_3, x_4, x_5)$, x_1 and x_2 are allowed to take any value from the set of values obtained by evenly dividing the range $[-2, 2]$ into 11 discrete concentrations. The remaining drug concentrations (x_3, x_4, and x_5) can take any of the 11 concentrations that evenly divide $[-4.5, 4.5]$. For the second drug response function $f_{5b}(x_1, x_2, x_3, x_4, x_5)$, we assume that each drug can take one of the 11 discrete values that evenly divide $[-3, 3]$. Finally, we also defined two 6-dimensional drug response functions

$$f_{6a}(x_1, x_2, x_3, x_4, x_5, x_6) = e^{-0.75(x_1)} \cdot \Big(\sin^2(x_2) + \cos(x_3)\Big) \cdot e^{-0.75(x_4^2 + x_5^2)} \cdot x_6 \tag{14}$$

and

$$f_{6b}(x_1, x_2, x_3, x_4, x_5, x_6) = e^{-0.1(x_1^2 - x_2^2) - 0.1(x_3^2 + x_4^2)} \cdot \cos^2(0.2x_5^3) \cdot \sin(0.2x_6^3), \tag{15}$$

where every drug concentration can take its value from one of the 11 discrete concentrations that evenly divide the range $[-2.5, 2.5]$.

Results

Optimizing the combination of two drugs

We first evaluated the overall performance of the ARU stochastic search algorithm based on four two-dimensional drug response functions (see Methods): (i) HIV inhibitor response $f_{2a}(x_1, x_2)$, (ii) second De Jong function (Rosenbrock's saddle) $f_{2b}(x_1, x_2)$, (iii) normalized lung cancer inhibition response $f_{2c}(x_1, x_2)$, and (iv) bacterial (*S. aureus*) inhibition response $f_{2d}(x_1, x_2)$. These functions are shown in Figure 3. For each drug response function, we searched for the optimal drug response using the proposed algorithm, starting from randomly selected drug concentrations x_1 and x_2. The parameter α that controls the randomness of the search was set to $\alpha = 1$, which implies that the algorithm adaptively determines the search direction (i.e., how to update the current drug concentration) by fully utilizing the past observations. Note that setting the parameter to $\alpha = 0$ in (7) would make the search completely random and independent of the past observations: at each step, the concentration of a given drug will be randomly increased or decreased with equal probability, regardless of the update rules given in (5). In each search experiment, the iterations were repeated up to two times the total number of possible drug combinations. This experiment was repeated 5,000 times to obtain reliable results. For comparison, we performed similar experiments using the Gur Game algorithm [9] and the stochastic search algorithm proposed in [11] with $\alpha = 1$. We computed the following two performance metrics: the success rate and the average number of unique drug combinations that need to be tested. The first metric is defined as the relative proportion of experiments, in which the algorithm was able to identify a potent drug combination whose response is within 5% of the maximum response, i.e., $f(x_1, x_2) \geq 0.95$. The second metric is defined as the average number of unique drug combinations that have to be tested until a potent drug combination is identified, in case the search is successful. These performance assessment results are summarized in Table 1. Note that the Gur Game algorithm has been tested using both the simultaneous update strategy as well as the sequential update strategy. Unlike the ARU algorithm and the search algorithm proposed in [11], which update one drug at a time (i.e., "sequential" drug update), the original Gur Game algorithm adopted in [9] updates all drugs simultaneously. However, it is also possible to use the sequential update strategy with the Gur Game algorithm, and we have evaluated both strategies for comparison. Table 1 shows that the proposed ARU algorithm outperforms the existing algorithms in terms of success rate. Furthermore, when the success rates are comparable, the ARU algorithm can in general identify an effective drug combination more efficiently, as reflected in the smaller number of unique drug combinations that need to be tested. We can get a more complete picture of the efficiency of the ARU algorithm from Figure S1 and Figure S2,

which respectively show the distribution of the number of unique drug combinations that need to be tested and the distribution of the number of iterations that are needed to identify a potent drug combination.

Optimizing multi-drug cocktails

Next, we tested the performance of the ARU algorithm for optimizing multi-drug cocktails that consist of three to six drugs. For this purpose, we used the eight hypothetical drug response functions that were defined before (see Methods). As in our previous experiments, for each drug response function, we used the proposed ARU algorithm (with $\alpha = 1$) to search for a potent drug combination whose response is within 5% of the maximum response (i.e., $f(\mathbf{x}) \geq 0.95$). In each search experiment, we started from an randomly selected initial concentrations, and continued the search up to 1,000 steps for 3 drugs, 2,000 steps for 4 drugs, 3,000 steps for 5 drugs, and 4,000 steps for 6 drugs. This experiment was repeated 5,000 times to obtain reliable performance assessment results. The simulation results are summarized in Table 2. As shown in this table, the proposed algorithm boasts a significantly higher success rate compared to the Gur Game algorithm [9]. It also results in either comparable or slightly improved success rate compared to the previous drug optimization algorithm ($\alpha = 1$) [11]. However, we can see that the ARU algorithm clearly outperforms the previous search algorithm in terms of efficiency, as reflected in the significantly smaller number of unique drug combinations that need to be tested until an effective combination is identified. Figures S3 and S5 show the distribution of the number of unique drug combinations that need to be tested to identify an effective drug cocktail. Similarly, Figures S4 and S6 show the distribution of the number of search iterations that are needed by each algorithm.

Drug optimization in the presence of measurement noise

In order to use a drug optimization algorithm in practical applications, the algorithm has to be robust to random fluctuations in the estimated drug response. To evaluate the robustness of the proposed ARU algorithm, we evaluated its search performance in the presence of measurement noise and compared it with other existing stochastic search algorithms. In these experiments, we considered two different types of search strategies. In the first search strategy (referred as **type-A**), when the search algorithm happens to revisit a drug combination that was previously tested, it does *not* re-evaluate the drug response and simply uses the previously estimated value. On the other hand, according to the second strategy (referred as **type-B**), the search algorithm always re-evaluates the drug response, even if it revisits a previously evaluated drug combination, since the measured response may be different every time due to the random

measurement noise. The first strategy may be useful when the noise level is relatively low, in which case this strategy may be able to reduce the total number of drug response evaluations, thereby reducing the overall experimental cost for identifying a potent drug combination. However, when the noise level is high, the search performance may be degraded as the search algorithm clings to the past (noisy) response, once it has been measured. In contrast, the second search strategy generally requires a relatively larger number of drug response evaluations, but it tends to be more robust to random fluctuations and noise in the measured drug response function.

In order to evaluate the performance of the different search algorithms in the presence of noise, we performed similar search experiments as before at three different levels of additive noise: 2%, 5%, and 8%. More precisely, we assume that

$$f_{\text{obs}}(\mathbf{x}) = f_{\text{true}}(\mathbf{x}) + \eta,$$

where $f_{\text{obs}}(\mathbf{x})$ is the observed drug response, $f_{\text{true}}(\mathbf{x})$ is the true response, and η is an independent random noise that is uniformly distributed over $(-u, u)$, where $u \in \{0.02, 0.05, 0.08\}$. For each drug response function and a given noise level u, we tested the performance of both search strategies. For **type-A** search, we evaluated the success rate and the average number of unique drug combinations that have to be tested until a potent drug combination is identified. For **type-B** search, we evaluated the success rate and the average number of iterations, instead of the number of unique drug combinations, until an effective combination is identified. This is because, in a **type-B** search, the search algorithm re-evaluates the drug response even if it revisits the same drug combination that was previously tested. The simulation results are shown in Table 3 – Table 7, for drug response functions with two to six drugs. The parameter α was set to $\alpha = 1$ for the ARU algorithm as well as the previous search algorithm proposed in [11].

As we can see in these Tables, measurement noise certainly affects the overall performance of the ARU algorithm, where a higher noise tends to reduce the success rate and increase the number of iterations as well as that of the unique drug combinations to be tested. For many drug response functions considered in our simulations, the performance degradation is typically not too significant for the proposed algorithm, showing that the ARU algorithm is relatively robust to measurement noise. However, we can also observe that the extent of performance degradation will critically depend on the landscape of the underlying drug response. In most cases, the ARU algorithm continued to substantially outperform other stochastic search algorithms [9, 11], demonstrating that it is better suited for practical drug optimization applications.

One interesting observation is that the performance of the Gur Game algorithm is typically not very

sensitive to measurement noise. In fact, in some cases, its performance even improves as the noise level goes up. The main reason for this phenomenon is as follows. As discussed earlier, the Gur Game algorithm does not adapt to the observed drug response function, and for this reason, its overall performance crucially depends on whether or not its predetermined FSA matches the drug response function at hand. As a result, if the FSA does not match the original drug response function well, ironically enough, the measurement noise may perturb the search process in such a way that improves the overall performance. In this sense, the fact that the Gur Game algorithm is not very sensitive to measurement noise reflects its inaptitude for handling various types of drug response functions, rather than its robustness to random fluctuations and noise in the measured drug response.

Conclusions

In this paper, we proposed a novel stochastic search algorithm, called the adaptive reference update (ARU) algorithm, which can be effectively used for optimizing the composition of combinatory drugs. The proposed algorithm intelligently utilizes the drug response values observed in the past to reliably predict how to beneficially update the drug concentrations to improve the drug response. As we demonstrated throughout this paper, the proposed algorithm addresses several shortcomings of previous drug optimization algorithms [9, 11], thereby improving the overall search performance. Numerical experiments based on various types of multi-drug response functions show that the ARU algorithm results in a higher success rate (i.e., higher probability of identifying a potent drug combination) while requiring significantly fewer drug response evaluations. Furthermore, the proposed algorithm is robust to random measurement noise, where its search performance is not substantially affected in the presence of noise and degrades gracefully as the noise level increases. Throughout our experiments, we used $\alpha = 1$ for the ARU algorithm as well as the previous search algorithm [11]. As discussed earlier, the parameter α controls the randomness of the search, by determining how much weight we should give to the drug response values that we observed in the past. In general, unless the observations are very noisy or the underlying drug response function is assumed to be extremely nonlinear, it would be best to set the parameter to the largest allowed value (i.e., $\alpha = 1$), so that we fully utilize the past observations for making our best informed guess about the beneficial drug update strategy. For comparison, we also repeated our simulations using $\alpha = 0.5$ and $\alpha = 0.75$, whose results are summarized in Table S1 – Table S7. We can see from these results that $\alpha = 1$ indeed leads to the best performance for the drug response functions and the noise levels we have considered in this paper.

Competing interests

The authors declare that there are no competing interests.

Authors' contributions

Conceived and developed the algorithm: MK, BJY. Performed the experiments: MK. Analyzed the data and wrote the paper: MK, BJY.

Acknowledgments

This work was supported in part by the National Science Foundation, through NSF Award CCF-1149544.

References

1. Borisy AA, Elliott PJ, Hurst NW, Lee MS, Lehar J, Price ER, Serbedzija G, Zimmermann GR, Foley MA, Stockwell BR, Keith CT: **Systematic discovery of multicomponent therapeutics**. *Proc. Natl. Acad. Sci. U.S.A.* 2003, **100**:7977–7982.
2. Chesney MA, Ickovics J, Hecht FM, Sikipa G, Rabkin J: **Adherence: a necessity for successful HIV combination therapy**. *AIDS* 1999, **13 Suppl A**:S271–278.
3. De Francesco R, Migliaccio G: **Challenges and successes in developing new therapies for hepatitis C**. *Nature* 2005, **436**:953–960.
4. Fitzgerald JB, Schoeberl B, Nielsen UB, Sorger PK: **Systems biology and combination therapy in the quest for clinical efficacy**. *Nat. Chem. Biol.* 2006, **2**:458–466.
5. Sawyers CL: **Cancer: mixing cocktails**. *Nature* 2007, **449**:993–996.
6. Zimmermann GR, Lehar J, Keith CT: **Multi-target therapeutics: when the whole is greater than the sum of the parts**. *Drug Discov. Today* 2007, **12**:34–42.
7. Calzolari D, Bruschi S, Coquin L, Schofield J, Feala JD, Reed JC, McCulloch AD, Paternostro G: **Search algorithms as a framework for the optimization of drug combinations**. *PLoS Comput. Biol.* 2008, **4**:e1000249.
8. Feala JD, Cortes J, Duxbury PM, Piermarocchi C, McCulloch AD, Paternostro G: **Systems approaches and algorithms for discovery of combinatorial therapies**. *Wiley Interdiscip Rev Syst Biol Med* 2010, **2**:181–193.
9. Wong PK, Yu F, Shahangian A, Cheng G, Sun R, Ho CM: **Closed-loop control of cellular functions using combinatory drugs guided by a stochastic search algorithm**. *Proc. Natl. Acad. Sci. U.S.A.* 2008, **105**:5105–5110.
10. Wang Y, Yu L, Zhang L, Qu H, Cheng Y: **A novel methodology for multicomponent drug design and its application in optimizing the combination of active components from Chinese medicinal formula Shenmai**. *Chem Biol Drug Des* 2010, **75**:318–324.
11. Yoon BJ: **Enhanced stochastic optimization algorithm for finding effective multi-target therapeutics**. *BMC Bioinformatics* 2011, **12 Suppl 1**:S18.
12. Zinner RG, Barrett BL, Popova E, Damien P, Volgin AY, Gelovani JG, Lotan R, Tran HT, Pisano C, Mills GB, Mao L, Hong WK, Lippman SM, Miller JH: **Algorithmic guided screening of drug combinations of arbitrary size for activity against cancer cells**. *Mol. Cancer Ther.* 2009, **8**:521–532.
13. Jelinek F: **Fast sequential decoding algorithm using a stack**. *IBM Journal of Research and Development* 1969, **13**(6):675–685.

14. Viterbi A, Omura J: *Principles of digital communication and coding*. McGraw-Hill, Inc. New York, NY, USA 1979.

15. Tsetlin M: **Finite automata and modeling the simplest forms of behavior**. *Uspekhi Matem Nauk* 1963, **8**:1–26.

16. Tung B, Kleinrock L: **Using finite state automata to produce self-optimization and self-control**. *Parallel and Distributed Systems, IEEE Transactions on* 1996, **7**(4):439–448.

17. Ji C, Zhang J, Dioszegi M, Chiu S, Rao E, Derosier A, Cammack N, Brandt M, Sankuratri S: **CCR5 small-molecule antagonists and monoclonal antibodies exert potent synergistic antiviral effects by cobinding to the receptor**. *Mol. Pharmacol.* 2007, **72**:18–28.

18. Storn R, Price K: **Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces**. *Journal of Global Optimization* 1997, **11**:341–359, [http://dx.doi.org/10.1023/A:1008202821328]. [10.1023/A:1008202821328].

Figures

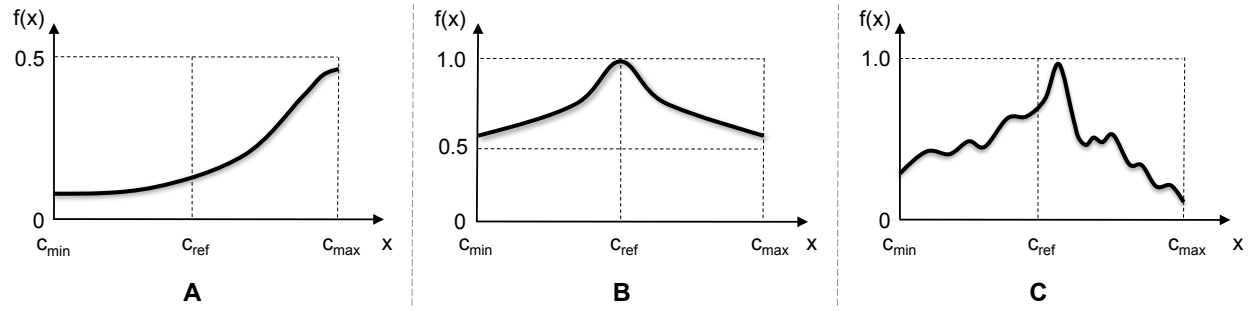

Figure 1: **Drug response functions.** (A) A normalized drug response $f(x)$ that is always below 0.5. (B) A normalized drug response $f(x)$ that is always above 0.5. (C) A drug response function with a large number of small variations.

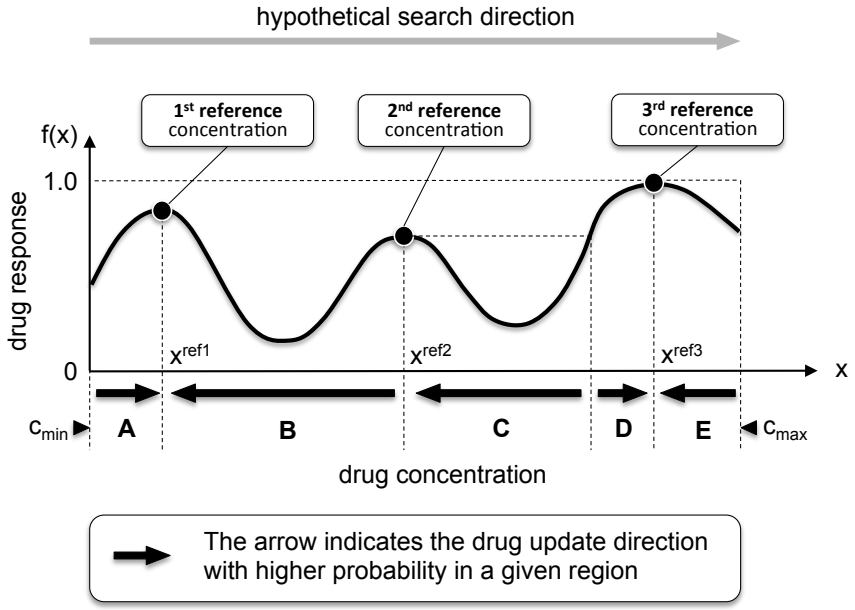

Figure 2: **Updating the reference point.** As the search for the optimal drug concentration continues from left to right (from the lowest concentration to the highest one), the reference concentration is updated from the initial drug concentration $c_{\min}$ to the local maximum points x^{ref1}, x^{ref2}, and x^{ref3}, according to this order.

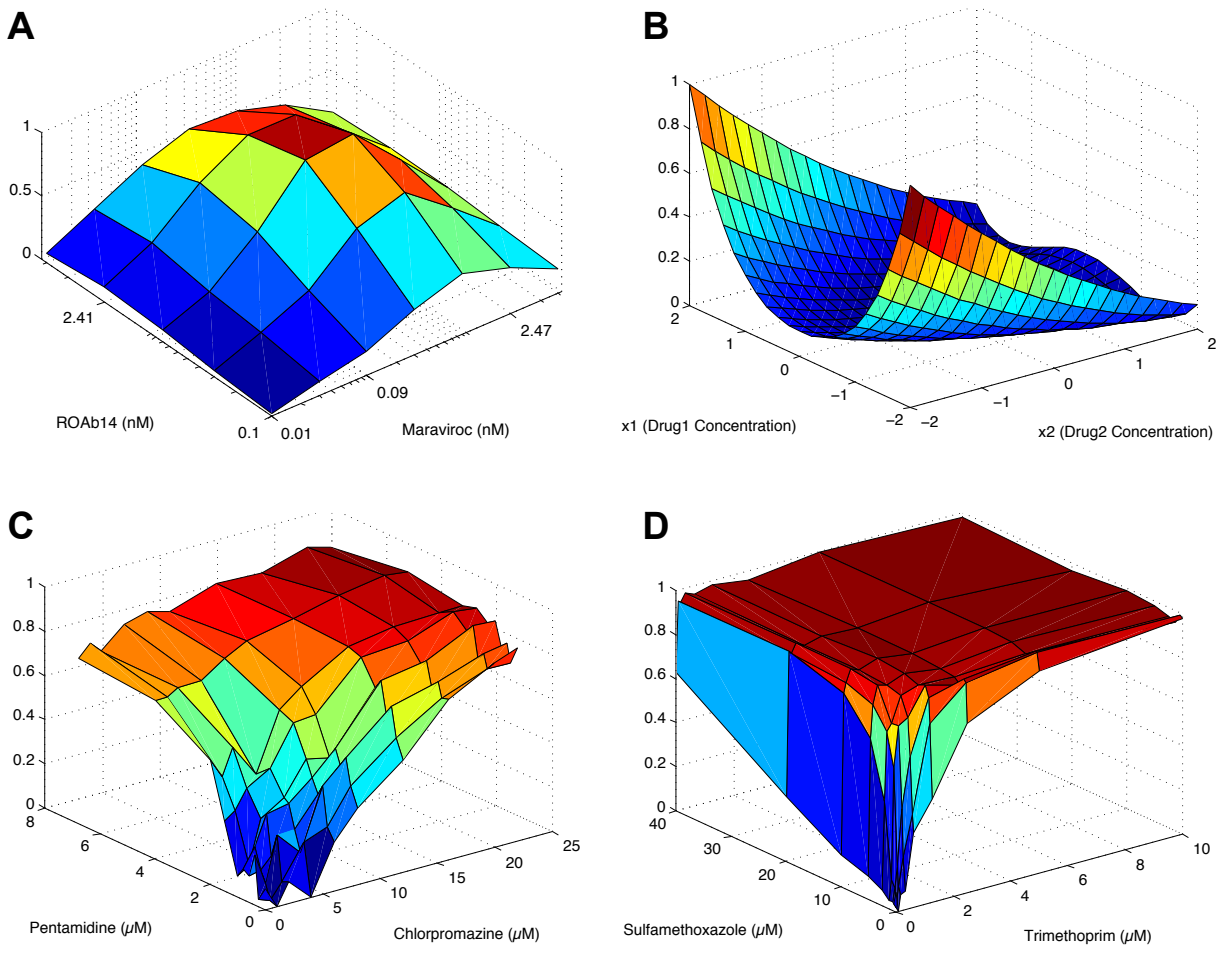

Figure 3: **Two-dimensional drug response functions.** (A) Inhibition of HIV. (B) Second De Jong function (Rosenbrock's saddle). (C) Inhibition of A549 lung carcinoma cell proliferation. (D) Inhibition of bacteria (*S. aureus*) proliferation.

Tables

Table 1: Performance for optimizing the combination of two drugs.

		Gur Game (simultaneous)		Gur Game (sequential)		Previous search algorithm [11] ($\alpha = 1$)		ARU algorithm (proposed) ($\alpha = 1$)	
		success rate	unique comb.	success rate	unique comb.	success rate	unique comb.	success rate	unique comb.
$f_{2a}(\mathbf{x})$: HIV INHIBITION	($M = 80$)	97%	13.2	95%	17.2	100%	13.4	**100%**	**12.1**
$f_{2b}(\mathbf{x})$: DE JONG (2ND)	($M = 441$)	9%	3.6	10%	4.5	99%	56.2	**99%**	**46.2**
$f_{2c}(\mathbf{x})$: CANCER INHIBITION	($M = 100$)	58%	11.3	53%	13.0	98%	13.2	**98%**	**12.4**
$f_{2d}(\mathbf{x})$: BACTERIA INHIBITION	($M = 81$)	96%	5.9	91%	6.8	100%	4.8	**100%**	**4.5**

Table 2: Performance for optimizing multi-drug cocktails.

		Gur Game (simultaneous)		Gur Game (sequential)		Previous search algorithm [11] ($\alpha = 1$)		ARU algorithm (proposed) ($\alpha = 1$)	
		success rate	unique comb.	success rate	unique comb.	success rate	unique comb.	success rate	unique comb.
$f_{3a}(\mathbf{x})$	($M = 11^3$)	1%	4.3	2%	5.5	100%	105.3	**100%**	**74.0**
$f_{3b}(\mathbf{x})$	($M = 11^3$)	83%	229.4	58%	204.8	100%	88.5	**100%**	**79.4**
$f_{4a}(\mathbf{x})$	($M = 11^4$)	20%	823.9	11%	666.6	100%	177.9	**100%**	**136.8**
$f_{4b}(\mathbf{x})$	($M = 11^4$)	52%	706.7	24%	520.9	100%	117.9	**100%**	**91.6**
$f_{5a}(\mathbf{x})$	($M = 11^5$)	8%	2.1	2%	4.8	100%	138.1	**100%**	**80.6**
$f_{5b}(\mathbf{x})$	($M = 11^5$)	89%	1013.4	54%	976.2	100%	252.9	**100%**	**216.8**
$f_{6a}(\mathbf{x})$	($M = 11^6$)	90%	1269.1	44%	1260.8	100%	191.9	**100%**	**178.1**
$f_{6b}(\mathbf{x})$	($M = 11^6$)	90%	446.7	40%	1033.2	100%	238.1	**100%**	**190.1**

Table 3: Performance for optimizing the combination of two drugs in the presence of noise.

	Noise level	Search type	Performance metric	Gur Game (simultaneous)	Gur Game (sequential)	Previous search algorithm [11] ($\alpha = 1$)	ARU algorithm (proposed) ($\alpha = 1$)
$f_{2a}(\mathbf{x})$	**(2%)**	A	success rate	97%	96%	100%	**100%**
			unique comb.	12.5	17.0	13.3	**11.6**
		B	success rate	97%	95%	100%	**100%**
			iterations	37.8	45.2	25.8	**20.0**
	(5%)	A	success rate	97%	96%	100%	**100%**
			unique comb.	12.6	17.1	13.3	**11.8**
		B	success rate	97%	95%	100%	**100%**
			iterations	38.0	45.4	26.6	**20.2**
	(8%)	A	success rate	97%	96%	100%	**100%**
			unique comb.	12.6	17.0	13.3	**12.0**
		B	success rate	97%	95%	100%	**100%**
			iterations	38.2	45.4	26.8	**20.4**
$f_{2b}(\mathbf{x})$	**(2%)**	A	success rate	10%	10%	99%	**99%**
			unique comb.	3.9	4.2	57.0	**45.1**
		B	success rate	10%	10%	99%	**99%**
			iterations	4.0	4.4	148.5	**120.2**
	(5%)	A	success rate	9%	9%	98%	**99%**
			unique comb.	4.1	4.5	62.9	**52.2**
		B	success rate	9%	9%	98%	**99%**
			iterations	4.3	4.7	172.1	**143.8**
	(8%)	A	success rate	8%	9%	97%	**98%**
			unique comb.	4.1	4.7	66.2	**55.6**
		B	success rate	9%	9%	97%	**98%**
			iterations	4.4	4.9	198.1	**167.3**
$f_{2c}(\mathbf{x})$	**(2%)**	A	success rate	61%	54%	98%	**98%**
			unique comb.	10.9	12.7	13.1	**12.5**
		B	success rate	60%	54%	98%	**98%**
			iterations	33.1	36.0	35.9	**35.2**
	(5%)	A	success rate	71%	66%	98%	**98%**
			unique comb.	11.4	13.2	12.9	**12.4**
		B	success rate	60%	54%	98%	**98%**
			iterations	33.2	35.4	36.7	**36.0**
	(8%)	A	success rate	88%	83%	98%	**98%**
			unique comb.	11.7	13.4	12.6	**12.0**
		B	success rate	60%	54%	98%	**98%**
			iterations	33.4	34.3	37.4	**37.0**
$f_{2d}(\mathbf{x})$	**(2%)**	A	success rate	100%	100%	100%	**100%**
			unique comb.	4.9	6.0	4.3	**4.1**
		B	success rate	96%	90%	100%	**100%**
			iterations	18.3	19.7	8.2	**7.7**
	(5%)	A	success rate	100%	100%	100%	**100%**
			unique comb.	4.9	5.7	4.3	**4.1**
		B	success rate	96%	91%	100%	**100%**
			iterations	18.4	19.6	8.1	**7.5**
	(8%)	A	success rate	100%	100%	100%	**100%**
			unique comb.	4.8	5.6	4.4	**4.2**
		B	success rate	96%	91%	100%	**100%**
			iterations	18.6	19.6	8.1	**7.1**

Table 4: Performance for optimizing the combination of three drugs in the presence of noise.

	Noise level	Search type	Performance metric	Gur Game (simultaneous)	Gur Game (sequential)	Previous search algorithm [11] ($\alpha = 1$)	ARU algorithm (proposed) ($\alpha = 1$)
$f_{3a}(\mathbf{x})$	**(2%)**	A	success rate	1%	3%	99%	**100%**
			unique comb.	2.4	7.3	110.6	**77.3**
		B	success rate	1%	3%	99%	**100%**
			iterations	2.8	10.9	201.1	**139.6**
	(5%)	A	success rate	1%	3%	99%	**100%**
			unique comb.	2.4	7.4	111.7	**78.4**
		B	success rate	1%	3%	99%	**100%**
			iterations	2.5	10.1	201.6	**144.0**
	(8%)	A	success rate	1%	3%	99%	**100%**
			unique comb.	2.5	7.7	113.3	**80.5**
		B	success rate	1%	3%	99%	**100%**
			iterations	2.3	9.4	210.9	**151.7**
$f_{3b}(\mathbf{x})$	**(2%)**	A	success rate	86%	69%	99%	**99%**
			unique comb.	224.3	201.6	110.8	**93.3**
		B	success rate	83%	59%	99%	**99%**
			iterations	367.1	419.1	211.5	**205.3**
	(5%)	A	success rate	89%	72%	99%	**99%**
			unique comb.	222.3	201.1	116.6	**106.2**
		B	success rate	83%	59%	98%	**98%**
			iterations	359.3	439.9	225.4	**222.9**
	(8%)	A	success rate	90%	74%	97%	**98%**
			unique comb.	225.9	200.9	126.4	**114.3**
		B	success rate	82%	60%	97%	**98%**
			iterations	359.4	431.3	249.1	**246.8**

Table 5: Performance for optimizing the combination of four drugs in the presence of noise.

	Noise level	Search type	Performance metric	Gur Game (simultaneous)	Gur Game (sequential)	Previous search algorithm [11] ($\alpha = 1$)	ARU algorithm (proposed) ($\alpha = 1$)
$f_{4a}(\mathbf{x})$	**(2%)**	A	success rate	21%	13%	96%	**98%**
			unique comb.	798.9	711.7	393.9	**327.4**
		B	success rate	21%	12%	96%	**99%**
			iterations	941.9	1032.1	558.3	**452.3**
	(5%)	A	success rate	21%	14%	90%	**95%**
			unique comb.	816.3	653.6	473.1	**398.3**
		B	success rate	21%	13%	90%	**95%**
			iterations	895.6	1022.9	675.4	**581.2**
	(8%)	A	success rate	24%	14%	85%	**92%**
			unique comb.	858.7	681.6	505.7	**433.6**
		B	success rate	23%	13%	85%	**92%**
			iterations	997.1	1008.9	720.8	**648.5**
$f_{4b}(\mathbf{x})$	**(2%)**	A	success rate	62%	41%	100%	**100%**
			unique comb.	634.5	523.1	138.0	**103.1**
		B	success rate	51%	26%	100%	**100%**
			iterations	932.4	903.0	236.9	**182.8**
	(5%)	A	success rate	75%	68%	100%	**100%**
			unique comb.	610.2	468.0	231.1	**150.9**
		B	success rate	50%	25%	100%	**100%**
			iterations	855.9	921.2	411.0	**258.8**
	(8%)	A	success rate	86%	82%	98%	**100%**
			unique comb.	525.8	430.1	314.2	**215.9**
		B	success rate	50%	24%	94%	**100%**
			iterations	835.3	979.2	602.0	**393.8**

Table 6: Performance for optimizing the combination of five drugs in the presence of noise.

	Noise level	Search type	Performance metric	Gur Game (simultaneous)	Gur Game (sequential)	Previous search algorithm [11] ($\alpha = 1$)	ARU algorithm (proposed) ($\alpha = 1$)
$f_{5a}(\mathbf{x})$	**(2%)**	A	success rate	8%	9%	100%	**100%**
			unique comb.	2.1	4.4	139.3	**122.5**
		B	success rate	9%	11%	100%	**100%**
			iterations	2.1	6.0	172.4	**154.9**
	(5%)	A	success rate	9%	11%	100%	**100%**
			unique comb.	3.9	7.9	142.1	**129.1**
		B	success rate	9%	12%	100%	**100%**
			iterations	108.3	38.6	177.1	**155.6**
	(8%)	A	success rate	10%	13%	100%	**100%**
			unique comb.	7.0	20.2	144.5	**131.4**
		B	success rate	9%	13%	100%	**100%**
			iterations	191.4	70.8	182.3	**156.5**
$f_{5b}(\mathbf{x})$	**(2%)**	A	success rate	89%	55%	100%	**100%**
			unique comb.	917.9	1026.3	407.1	**343.9**
		B	success rate	90%	55%	100%	**100%**
			iterations	999.8	1325.1	516.5	**444.3**
	(5%)	A	success rate	90%	57%	98%	**100%**
			unique comb.	932.8	1002.7	507.7	**463.6**
		B	success rate	90%	56%	99%	**99%**
			iterations	1004.7	1332.7	656.9	**562.4**
	(8%)	A	success rate	91%	59%	97%	**98%**
			unique comb.	959.5	971.2	578.8	**534.8**
		B	success rate	90%	56%	99%	**99%**
			iterations	1015.2	1341.0	735.0	**668.6**

Table 7: Performance for optimizing the combination of six drugs in the presence of noise.

	Noise level	Search type	Performance metric	Gur Game (simultaneous)	Gur Game (sequential)	Previous search algorithm [11] ($\alpha = 1$)	ARU algorithm (proposed) ($\alpha = 1$)
$f_{6a}(\mathbf{x})$	**(2%)**	A	success rate	91%	43%	100%	**100%**
			unique comb.	1280.0	1214.1	476.8	**432.6**
		B	success rate	90%	43%	100%	**100%**
			iterations	1352.6	1662.6	531.4	**503.6**
	(5%)	A	success rate	90%	44%	99%	**99%**
			unique comb.	1262.3	1247.2	621.0	**598.3**
		B	success rate	90%	45%	100%	**100%**
			iterations	1396.8	1675.9	763.7	**736.1**
	(8%)	A	success rate	90%	46%	98%	**98%**
			unique comb.	1204.7	1302.4	723.2	**698.8**
		B	success rate	90%	46%	98%	**98%**
			iterations	1412.2	1681.9	875.2	**834.3**
$f_{6b}(\mathbf{x})$	**(2%)**	A	success rate	91%	43%	100%	**100%**
			unique comb.	509.9	971.0	341.4	**237.7**
		B	success rate	94%	44%	100%	**100%**
			iterations	1240.7	1646.0	436.5	**293.5**
	(5%)	A	success rate	90%	42%	100%	**100%**
			unique comb.	473.8	970.7	349.9	**279.8**
		B	success rate	94%	44%	100%	**100%**
			iterations	1278.6	1704.9	480.8	**324.6**
	(8%)	A	success rate	89%	42%	100%	**100%**
			unique comb.	454.4	969.9	457.4	**353.2**
		B	success rate	94%	44%	100%	**100%**
			iterations	1333.1	1775.5	545.3	**391.5**