Principle of the Gur Game

The principle of the Gur Game (1) is based on biased random walks of finite-state automata (a set of drugs). The automata describe a set of drugs with assigned concentration values and a set of rules is included for determining how the concentration of the drug switches from one value to the other. Each drug concentration is referred as a state of the automaton. The overall goal of the automata design is to have the drugs to self-organize (choose the optimal concentrations) in an attempt to maximize the overall system performances (desired biological responses indicated by biomarkers). The biological response is transformed into a reward function, which describes the probability of the drug to switch between different concentrations.

To understand the principle of the Gur Game algorithm, it is helpful to consider an example for choosing the concentration of a drug to maximize the antiviral activity (AVA). We design a finite-state automaton with only two states, i.e., one drug with two concentrations (SI Fig. 7). Assuming the two concentrations, C1 and C2, of the drug will give antiviral activities of AVA_1 and AVA_2, respectively. A reward function can be defined to map AVA_1 and AVA_2 to probabilities of r_1 and r_2 of the drug to be rewarded. For example, the percentage of cells not being infected (the AVA) can be considered to be the probability of the drug being rewarded (i.e., the reward function). At each iteration, the drug can either be awarded or penalized. If the concentration C1 is applied, then the drug will have a chance of r_1 to be rewarded. The drug then has a probability of $(1 - r_1)$ and $(1 - r_2)$ to receive a penalty in the corresponding states. If the drug is rewarded, it chooses to keep the same concentration. Otherwise, the drug concentration will be switched from one concentration to the other. Because a concentration that gives higher AVA will have higher chance to be rewarded, the drug concentration will have a higher chance to stay in a concentration with high AVA. If the drug concentration gives a low AVA, the drug will have a high chance to be switched to the other concentration. In general, this design encourages the drug to choose a concentration with high AVA because the drug has a higher chance of being rewarded.

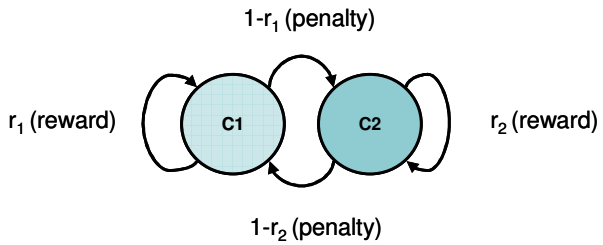


Fig. 7. A simple automaton design with two states (one drug with two possible concentrations). The drug concentration stays the same if the automaton (drug) is rewarded. Otherwise (penalized), the drug is switched to the other concentration.

The asymptotic behavior of the drug can be modeled by using the Markov chain analysis (2). We define π_1 and π_2 to be the steady-state probabilities of being in drug concentrations of C1 and C2, respectively. The fact that the probabilities sum to unity gives Eq. **1** (the drug chooses either C1 or C2). Equating the transition probabilities (switching from one concentration to the other) leads to Eq. **2**. Solving the equations gives the steady-state probability of choosing C2 in Eq. **3**.

$$\pi_1+\pi_2 = 1 \qquad \textbf{[1]}$$

$$\pi_1(1-r_1) = \pi_2(1-r_2) \qquad \textbf{[2]}$$

$$\pi_2 = \frac{1-r_1}{2-r_2-r_1} \qquad \textbf{[3]}$$

If we assume drug concentrations, C1 and C2, give AVA of 0.4 and 0.8 and the AVA is directly defined as the reward probability (i.e., $r_1 = 0.4$ and $r_2 = 0.8$,), the steady state probability of the drug to choose the concentration C2 will be 0.75, whereas the probability of choosing the concentration C1 is 0.25. Therefore, the drug chooses C2 (high AVA) three times more often than C1 (low AVA). In other words, the drug self-organizes to spend more time in a concentration with a higher AVA level.

The same idea can be extended to the general case of a drug with multiple concentrations. In general, the drug moves to the system state toward the center of the state space if penalized, and away from the center if rewarded (see SI Fig. 8). This design allows the drug to "detect" promising trends of drug concentrations. SI Fig. 8 shows a drug with $2n$ states (from $-n$ to $+n$) and each state is assigned with a different concentration. If the drug is rewarded, the automaton moves from state i to $i + 1$ if i is positive or from i to $i - 1$ if i is negative. The automaton stays in state n or –n if it is in either one of those states. For a penalty, it moves from state 1 to –1 or *vice versa* if it is at one of the state, otherwise it moves from state i to $i - 1$ if i is positive, or from i to $i + 1$ if i is negative. If a low drug concentration has a high AVA (being reward), the automaton attempts to further improve the performance by searching for even lower concentrations. If the drug concentration goes too low and the drug has a high chance to be penalized, the drug concentration will be driven back to a relatively higher concentration. In the case of multiple drugs, the drugs collectively search for a mixture with high AVA (being reward). This framework provides a very rapid and robust control approach for optimizing drug combinations. For complete details, the reader is referred to original discussions of the Gur Game (1-4).

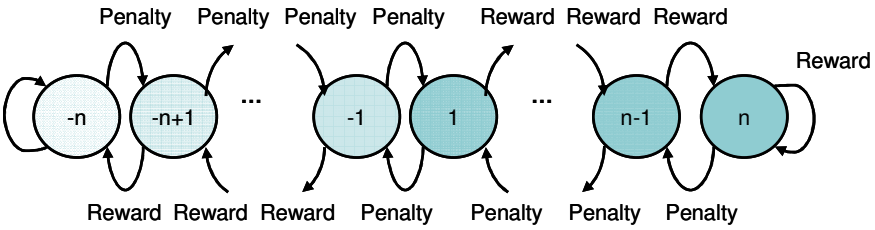


Fig. 8. An automaton design with multiple concentrations.

Implementation of the Gur Game

Flowchart Representation of the Gur Game. SI Fig. 9 shows a flowchart representation of the Gur Game applied in this study. The study started with initiation of the concentrations of a mixture. Then, an experiment using the drug mixture was performed to estimate the reward value (the probability of the drug being rewarded or penalized). In the VSV experiment, the reward value was estimated according to the percentage of cells not expressing GFP (i.e., not infected by

the virus). In the NF-κB experiment, the peak GFP intensities were normalized to map the system state from 0 to 1 with high GFP intensities corresponding to values nearer 1. The GFP value was the average fluorescence intensity of individual cells measured by a 16 bit cooled CCD camera (Photometric CH350L). Intensity values of individual cells were measured by using ImageJ. The GFP data are the average response of ≈100 cells. The data are normalized to be the reward value. The normalization value was determined experimentally and iteratively such that the reward value is between 0 and 1. A random number from 0 to 1 was generated and was compared with the reward value for each automaton. A set of predefined rules of the automata design were used to determine the drug concentrations in the subsequent iteration to probabilistically drive the drugs toward viral inhibition or high NF-κB activity. The concentrations of the drugs were then determined for the next iteration. The process is repeated for each drug during each iteration. The process can be terminated when the system states are spending a large portion of time in some states and/or when a preset performance of the system is reached. Otherwise, a new iteration is initiated and the process repeats.

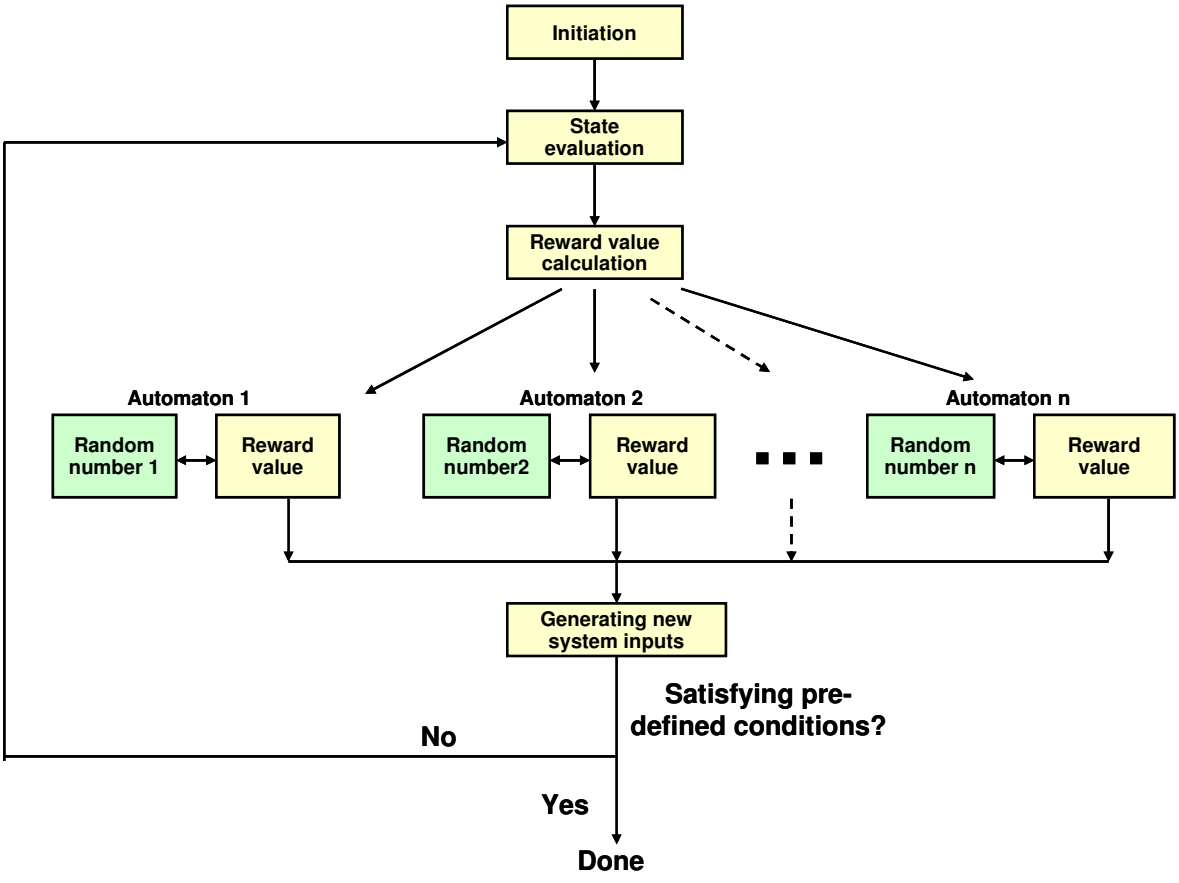


Fig. 9. Flowchart representation of the Gur Game algorithm.

Drug Concentrations for the Antiviral Experiment. We have designed four sets of experiments for searching the antiviral drug combinations. The automata design is shown in SI Tables 2 to 4. Set 1 and set 2 have 6 states (concentrations) for each automaton (drug). Set 3 and set 4 have the same search space, which has 10 states for each automaton. Each state represents a drug concentration shown in the SI Tables 2 to 4 and the corresponding automata are shown below. We have also applied different initial conditions in the experiments (SI Table 5). For set 1 and set 2, initial concentrations were zero for all agents and random initializations were applied for set 3 and set 4.

Tables 2-4. Automata design for antiviral drug cocktails.

Table 2. Search space for set 1

State	-3	-2	-1	1	2	3
IFNα, pg/ml	*0*	*1.3*	*6.5*	*13*	*65*	*130*
IFNβ, ng/ml	*0*	*0.1*	*0.5*	*1*	*5*	*10*
IFNγ, ng/ml	*0*	*10*	*50*	*100*	*500*	*1000*

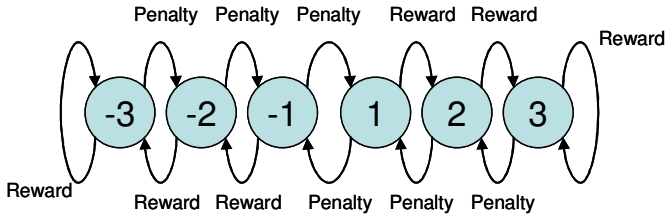


Table 3. Search space for set 2

State	-3	-2	-1	1	2	3
IFNα, pg/ml	*0*	*0.65*	*1.3*	*6.5*	*13*	*65*
IFNβ, ng/ml	*0*	*0.05*	*0.1*	*0.5*	*1*	*5*
IFNγ, ng/ml	*0*	*5*	*10*	*50*	*100*	*500*
Puromycin, µg/ml	*0*	*0.125*	*0.25*	*1.25*	*2.5*	*12.5*

Ribavirin, µg/ml	*0*	*0.5*	*1*	*5*	*10*	*50*

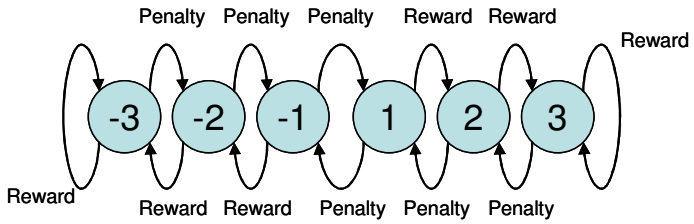


Table 4. Search space for set 3 and set 4

State	-5	-4	-3	-2	-1	1	2	3	4	5
IFNα, pg/ml	*0*	*0.65*	*1.3*	*3.9*	*7.8*	*15.6*	*32.5*	*65*	*130*	*260*
IFNβ, ng/ml	*0*	*0.05*	*0.1*	*0.3*	*0.6*	*1.2*	*2.5*	*5*	*10*	*20*
IFNγ, ng/ml	*0*	*5*	*10*	*30*	*60*	*120*	*250*	*500*	*1000*	*2000*
Puromycin, µg/ml	*0*	*0.125*	*0.25*	*0.75*	*1.5*	*3*	*6.25*	*12.5*	*25*	*50*
Ribavirin, µg/ml	*0*	*0.5*	*1*	*3*	*6*	*12*	*25*	*50*	*100*	*200*

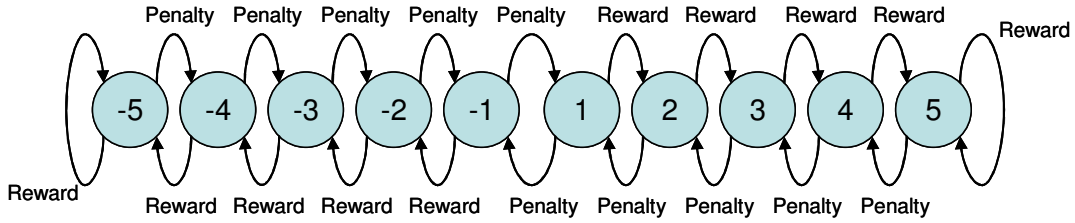


Table 5. Initial conditions of the experiment.

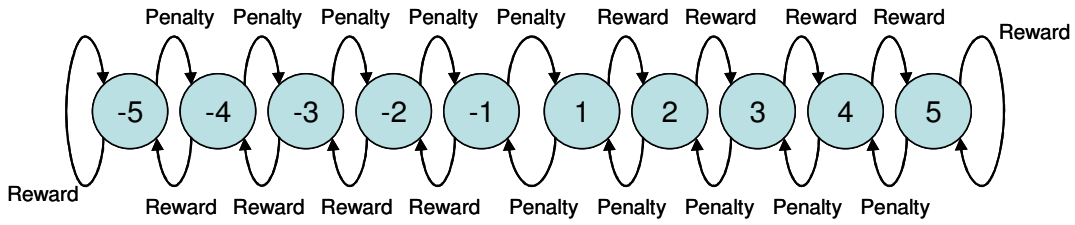
	Set 1	Set 2	Set 3	Set 4
IFNα, pg/ml	*0*	*0*	*7.8*	*1.3*
IFNβ, ng/ml	*0*	*0*	*0.6*	*1.2*
IFNγ, ng/ml	*0*	*0*	*60*	*1000*
Puromycin, µg/ml	-	*0*	*1.5*	*0.75*
Ribavirin, µg/ml	-	*0*	*6*	*12*

Cytokine Concentrations for Regulating NF-κB Activity. In this Gur Game implementation, we assigned a different drug concentration to each state of the automaton. This allows for a search technique that rapidly moves around the search space and is less likely to be trapped in any one state. All cytokines were represented by automata with 10 discrete states (0, 0.25, 0.5, 1,

2.5, 5, 10, 25, 50, and 100) ng/ml, which span across three orders of magnitude, in the NF-κB experiment (SI Table 6). The states of automata were designed for maximizing the range of cytokine concentrations being tested while maintaining the resolution for searching the optimal cytokine concentration. During each iteration, a random number was generated for each automaton. The numbers were compared with the reward value. If the reward value was greater than the random number, the automaton was rewarded. Otherwise, the automaton was penalized. The automaton then decided the state in the next iteration according to the automaton design. In general, the state moves toward the center if penalized and away from the center if rewarded.

Table 6. Automata design for manipulating NF-κB activity

State	**-5**	**-4**	**-3**	**-2**	**-1**	**1**	**2**	**3**	**4**	**5**
TNFα, ng/ml	*0*	*0.25*	*0.5*	*1*	*2.5*	*5*	*10*	*25*	*50*	*100*
TNFβ, ng/ml	*0*	*0.25*	*0.5*	*1*	*2.5*	*5*	*10*	*25*	*50*	*100*
IL-1α, ng/ml	*0*	*0.25*	*0.5*	*1*	*2.5*	*5*	*10*	*25*	*50*	*100*
IL-1β, ng/ml	*0*	*0.25*	*0.5*	*1*	*2.5*	*5*	*10*	*25*	*50*	*100*
EGF, ng/ml	*0*	*0.25*	*0.5*	*1*	*2.5*	*5*	*10*	*25*	*50*	*100*
BAFF, ng/ml	*0*	*0.25*	*0.5*	*1*	*2.5*	*5*	*10*	*25*	*50*	*100*



Characteristics of the Gur Game Algorithm

In most biological systems, a complete model of the governing network is often not available and only partial information of the system of interest is known. The Gur Game provides a generic mechanism for regulating these complex systems without the requirement of a preassumed model of the system or knowledge of how the system performance depends on the variables being manipulated. Unlike gradient search methods, the Gur Game drives the system state to higher performance probabilistically, instead of deterministically. The probabilistic characteristic

of the Gur Game allows the system state to "escape" from local optima and search for the global optimum in the search space. For the same reason, the algorithm performs robustly in a noisy environment. This robustness is critical for controlling biological systems, which are intrinsically noisy (5).

Another important characteristic of the Gur Game is that the optimality is determined by the average behavior (1, 2). The automata states spend more time at states with high reward probability, i.e., good system performance. There is no dynamic difference between the transient behavior of the system and its steady-state behavior. This feature enables the system to respond to a changing population and/or reward function. It is interesting to compare this characteristic with the robustness in cellular functions (6). In general, the automata and the reward function (mapping between the system performance and reward probability) should be adjusted to fine tune the balance between the robustness, converging rate, and the ability to escape from local traps (1, 2). For distributed control and other self-organizing systems, it is preferable to have small step size such that the system locks into desire behaviors (4). The trade-off is that more steps are required for the system to reach to an optimal solution. For searching and optimization experiments, increasing the "randomness" with larger step size improves the chance that the automata will escape from local optima and rapidly search for the global solution (7).

Reward Function and Automata Design

Reward Function. A key concept in the Gur Game is the reward function, which is a global figure of merit. Basically, the reward function is a measurement of the system performance as a whole. Similar concepts can be found in other optimization approaches. For instance, it is called a fitness function in genetic algorithms (8) and an energy function in simulated annealing (9). It is one of the most important parts for the success of a closed-loop optimization experiment. The design of a reward function critically determines how well the algorithm is able to solve the problem. In the VSV experiment, the reward function is the percentage of cells not being infected (indicated by GFP expression). In the NF-κB experiment, the reward function is estimated by the NF-κB activity (indicated by the fluorescence intensity). A deep appreciation of the reward function not only provides the scientist extra freedom to improve the search but also

broadens the applicability to a wide class of biological systems. To design a reward function, a set of systemic output indicators should be defined. In most cases, phenotypic responses such as proliferation rates, gene expressions, and differentiation efficiency can be directly considered as the reward function. It should also be noted that the reward function can be nonlinearly mapped to the system performance to improve the search efficiency and the converging rate. In general, the reward function of the system can be multimodal, nonlinear, and even discontinuous.

Automata Design. As discussed, the design of the automata is an important component in the realization of the optimization process. Concentration of drugs (states of the automata) can be assigned depending on the range of the drug concentrations intended to be explored. The values can be linear, logarithmic, or other nonlinear functions depending on the nature of the problem. The number of states should be designed according to the requirements of the converging rate, randomness, and resolution. Unlike, surrogate-based optimization, experimental implementation of a closed-loop optimization scheme does not require "training" of inputs. Extension of the input ranges (drug selection and concentration) can be performed by including extra automata and states of the automata. Another advantage of the stochastic search algorithm is that the converging rate is usually maintained during extension of the input ranges.

Systems with Multiple Outputs. For optimizing multiple parameters, the vector norm or a weighted average of these parameters can be applied to optimize the overall performance. Other schemes can also be applied to account for multiple output parameters. In general, the more information, such as some key molecular components, that is known about the system, the easier to design the experiment. Unfortunately, these parameters are usually not clear and the reward functions in some problems may not be straightforward. Most cases require a slight modification of the problem to define the reward function. If necessary, multiple generations of the closed-loop optimization experiment can be performed systemically to identify the appropriate reward functions. In the current study, we have demonstrated the optimization of two different biological systems. Other examples of the implementation of the Gur Game can be found elsewhere (1, 2).

Microfluidic Channel and Cell Culture

Microfluidic Cell Culture System. The microfluidic channel is integrated into a cell culture chamber (Instec Inc, HCS60-STC20A) with temperature control at 37°C. The chamber is mounted on a fluorescence microscope (Nikon TE200) for real-time monitoring. Fluorescence images were captured with a 1024 × 1024 pixel, 16-bit cooled CCD camera (Photometric CH350L). Mini peristaltic pumps (Instech Inc, P625-10638), pressure transducer (Honeywell, ACSX05DN), and temperature probe (Omega, DP460) were connected to the chamber and were centrally controlled by a Labview software (National instruments). The entire setup was sitting inside a vertical clean bench, which prevents contamination from the environment (SI Fig. 10).

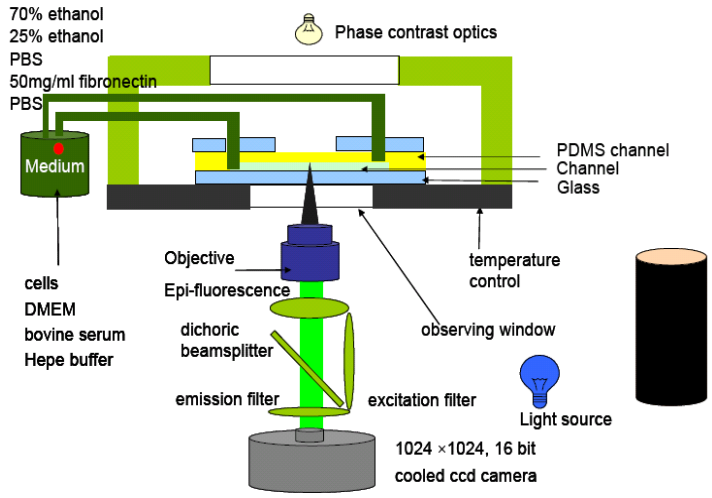


Fig. 10. Microfluidic channel and cell culture system.

Preparation Microchannel for Cell Culture. Before the experiment, the channels were sterilized by flowing 70% ethanol for 30 min. The channels were then washed with 25% ethanol, DI water, and PBS each for 10 min. The channel surface was then incubated with 200 mg/ml fibronectin for >1 h to modify the channel surface for promoting cell adhesion. Culture media (DMEM without phenol red) was flowed into the channel and the channel was then filled with a suspension of 293T cells. The cells were allowed to adhere to the channel surface for 4-6 h and medium was then perfused at a flow rate of 0.9 ml/min. At this flow rate, the cells experienced wall shear stress of 0.02-0.5 dyne/cm^2 as estimated by numerical simulation (CFDRC CFD-ACE+). This value was below the typical range of shear stress for affecting normal cell functions

or inducing other cellular responses. The cells were allowed to culture in the device overnight before the experiment. The cells could be cultured for an extended period and reached a high confluency. With normal culture conditions, the cells had similar morphologies and growth rates compared to those in cell culture dish.

1. Tsetlin ML (1963) Finite automata and the modeling of the simplest forms of behavior. *Uspekhi Matem Nauk* 8:1-26.

2. Tung B, Kleinrock L (1996) Using finite state automata to produce self-optimization and self-control. *IEEE Trans Parallel Distrib Syst* 7:439-448.

3. Tsetlin ML (1963).Finite automata and modeling the simplest forms of behavior. *Uspekhi Matem Nauk* 8:1-26.

4. Tung B (1994) Distributed control using simple automata. University of California, Los Angeles.

5. Rao CV, Wolf DM, Arkin AP (2002) Control, exploitation and tolerance of intracellular noise. *Nature* 420:231-237.

6. Stelling J, Sauer U, Szallasi Z, Doyle FJ, Doyle J (2004) Robustness of cellular functions. *Cell* 118:675-685.

7. Ho S, Nassef H, Pornsinsirirak N, Tai YC, Ho CM (2003) Unsteady aerodynamics and flow control for flapping wing flyers. *Prog Aerospace Sci* 39:635-681.

8. Forrest S (1993) Genetic algorithms—Principles of natural-selection applied to computation. *Science* 261:872-878.

9. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220:671-680.