Low Complexity Control Policy Synthesis for Embodied Computation in Synthetic Cells

Ana Pervan and Todd D. Murphey

Department of Mechanical Engineering
Northwestern University
Evanston, IL 60208, USA
anapervan@u.northwestern.edu
t-murphey@northwestern.edu

Abstract. As robots become more capable, they also become more complicated—either in terms of their physical bodies or their control architecture, or both. An iterative algorithm is introduced to compute feasible control policies that achieve a desired objective while maintaining a low level of design complexity (quantified using a measure of graph entropy) and a high level of task embodiment (evaluated by analyzing the Kullback-Leibler divergence between physical executions of the robot and those of an idealized system). When the resulting control policy is sufficiently capable, it is projected onto a set of sensor states. The result is a simple, physically-realizable design that is representative of both the control policy and the physical body. This method is demonstrated by computationally optimizing a simulated synthetic cell.

1 Introduction

Many roboticists seek to create robots that are as capable as possible, assuming that optimal design is achieved when a robot can perfectly perform a task. But as robots become more capable, they often become more complicated. While there are some cases where this is acceptable, there are many where complexity is an undesired consequence of optimization. A conflict exists between equipping a robot with what is sufficient and what is necessary—what can enable a robot to succeed (and is likely complex) and what is minimally required for it to achieve its goal (and is necessarily simple). This is especially evident when designing for embodied computation—robots without any on-board, CPU-based, traditional computational capabilities.

Robots traditionally require three elements: sensors, computation, and actuators. Some robotic systems employ embodied computation to reduce weight and energy—for example fully mechanical devices, like passive dynamic walkers [9], [28] or those used in prosthetic limbs [22]—while others must *necessarily* resort to embodied computation because of scale, for example, robots on the micro- or nano-scales [13], [16], [17]. We focus on the latter case later. So although robot design is traditionally identifying which sensory, computation, and actuation elements can be combined to best achieve a goal, here we will examine a framework for designing the sensory and actuation elements of robots so that no traditional, CPU-based computation is necessary to accomplish a goal.

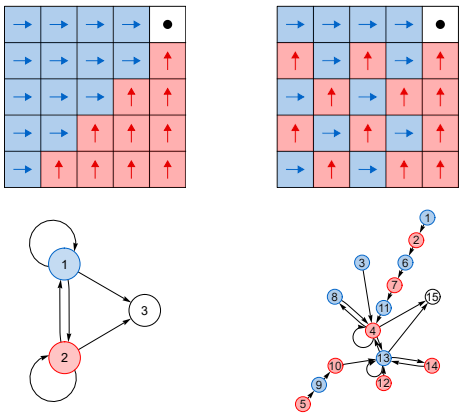

Fig. 1: A simple example of a control policy. Here the state space is a two dimensional 5×5 grid and the desired state is in the upper right of the state space. At each state, the suitable control is indicated by its assigned color. (a) A *simple* control policy with only three different states in the finite state machine. (b) A *complex* control policy, with fifteen different states in the finite state machine. Both achieve the task with different implementations.

We first examine the relationship between control policy design and physical robot body design. A control policy assigns an action for a system at each time or state. Symbolic control policies have been useful in robot control and motion planning [3], [21] for systems with limited computational power [25]. An example is shown in Fig. 1, where the system consists of three possible control modes: move right (blue), move up (red), and stay still (white) and its goal is to navigate to the upper right corner of the grid world. A simple robot placed in this environment with one of these policies in its memory would be able to achieve its task without any traditional computation, replacing logical operators such as inequalities with physical comparators to relate sensor states to control actions.

Figure 1 illustrates the difference in complexity between these two policies. The policy in Fig. 1 (a) requires a combination of sensors that are capable of differentiating between three different regions of the state space, and the policy in Fig. 1 (b) requires sensors that are able to discern between fifteen different regions. It is simpler to physically implement the robot design implied by the control policy in Fig. 1 (a) than the policy in Fig. 1 (b).

We pose the physical design problem as the projection of a policy onto an admissible set of physical sensor-actuator interconnections, the complexity of which must be managed during policy iteration. This complexity is a measure of logical interconnections between sensor states and control modes (which correspond to arrows on the graphs in Fig. 1). Methods from switched systems literature are used to organize these sensor-actuator connections so that they change minimally as a function of state. This is challenging because [4] shows that optimal solutions for discrete switched systems will chatter with probability 1, and chattering implies a complex policy. That

is, when choosing among a finite number of sensor-actuator pairings, optimization of an objective function will necessarily lead to arbitrarily complex dependencies on state, and the physical implementation of such a policy would consequently be very complex (and often not physically realizable). Results from [7] show that solutions with slow mode switching are "almost" as good as chattering solutions. Properties from [7] are used here to design control policies with minimal mode switching and then project them onto a feasible sensor space, as illustrated in Fig. 2, resulting in a methodology for designing robots with embodied computation.

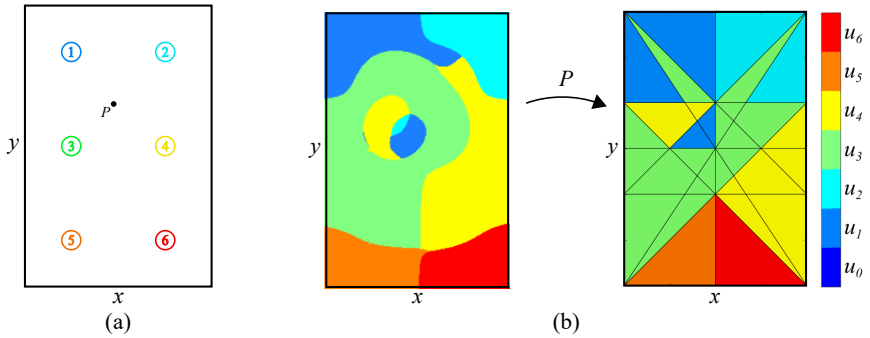

Fig. 2: (a) The state space and controls for the synthetic cell example system introduced in Sec. 3. At any location in the state space, the robot is able to choose one of seven different control modes: attraction to chemical potentials at the six different sources, or zero control. (b) A control policy for this system projected onto a feasible set of sensor states. (These figures will be explained in detail in Sec. 5 and Sec. 6.)

After reviewing related work in Section 2, this methodology will be explored in terms of an extended example. The example system, called a synthetic cell, will be introduced in Section 3. The primary contributions of this work can be summarized as follows:

1. Quantitative definitions are given for design complexity and task embodiment, described in Section 4.
2. An iterative algorithm is developed in Section 5, which creates control policies with low design complexity while increasing task information.
3. A physically realizable robot design is generated by dividing the state space into regions to create a set of sensor states, illustrated in Section 6, and projecting the low complexity control policy onto this set.

These are supported by simulations of synthetic cells.

2 Related Work

This work is substantially inspired by [8], which defines design problems as relations between functionality, resources, and implementation and shows that despite being non-convex, non-differentiable, and noncontinuous, it is possible to create languages and optimization tools to define and automatically solve design problems. The optimal

solution to a design problem is defined as the solution that is minimal in resources usage, but provides maximum functionality. We apply this definition by proposing a min-max problem in which the goal is to minimize design complexity (representative of the amount of sensors and actuators required, i.e., the resources), and maximize task embodiment (i.e., the functionality of the design).

Robotic primitives are introduced in [23] as independent components that may involve sensing or motion, or both. These are implemented in this work as actuator and sensor libraries from which we allow our algorithm to choose components. Our definition of task embodiment, which is defined in Sec. 4, parallels the dominance relation proposed in [23] that compares robot systems such that some robots are stronger than others based on a sensor-centered theory of information spaces.

Similarly, our definition of design complexity (Sec. 4) parallels an existing notion of conciseness, presented in [24]. The results in [24] are motivated by circumstances with severe computational limits, specifically addressing the question of how to produce filters and plans that are maximally concise subject to correctness for a given task. This is very related to our goal of finding the simplest way to physically organize sensors and actuators so that a (computationally limited) robot can achieve a given task.

The work presented in [15] produces asymptotically optimal sampling-based methods and proposes scaling laws to ensure low algorithmic complexity for computational efficiency. These algorithms were originally developed for path planning, but we apply similar ideas for generating simple control policies. The methods described in [15] start with an optimal, infinite complexity solution, and from that develop simpler plans. Here, we start with a zero complexity policy and move towards more complex, better performing solutions—while maintaining a level of computational complexity appropriate for physical implementations of embodied computation.

3 Motivating Example: Synthetic Cells

How can we use control principles to organize sensor components, actuator components, and their interconnections to create desired autonomous behavior, without relying on traditional computation? To answer this question we will consider the extended example of a synthetic cell—a small robot that only has a finite number of possible sensor and actuator states and potential pairings between them [19]. The purpose of this example system is to show a concrete implementation of the methods in Section 5, and to better illustrate the relationship between control policy design and physical robot body design.

A synthetic cell is a mechanically designed microscopic device with limited sensing, control, and computational abilities [19]; it is essentially an engineered cell. A synthetic cell is in a chemical bath and generates movement by interacting with its environment using chemical inhibitors, and it contains simple circuits that include minimal sensors and very limited nonvolatile memory [20]. Such a device is 100μm in size or less, rendering classical computation using a CPU impossible. But these simple movement, sensory, and memory elements can potentially be combined with a series of physically realizable logical operators to enable a specific task. How can these discrete structures be algorithmically organized to combine sensing and control to accomplish an objective?

4 Design Complexity and Task Embodiment

Graph entropy [1],[10] will be used as a measure of design complexity for comparing robot designs. The complexity of a control policy is equated with the measure of entropy of its resulting finite state machine.

A finite state machine consists of a finite set of states (nodes), a finite set of inputs (edges), and a transition function that defines which combinations of nodes and edges lead to which subsequent nodes [26]. The finite set of nodes that the system switches between are the *control modes*, and the edges—inputs to the system which cause the control modes to change—are the *state observations* (Fig. 1(a)).

Finite state machines and their corresponding adjacency matrices are generated numerically, by simulating a synthetic cell forward for one time step, and recording control modes assigned at the first and second states. These control mode transitions are counted and normalized into probabilities, and the resulting adjacency matrix A is used in the entropy calculation,

$$h = -\sum_i A(i)\log(A(i)) \tag{1}$$

which results in a complexity measure h for each robot design. This measure of complexity is more informative than other metrics (e.g., simply counting states) because it is a function of the *interconnections* between states—which is what we want to minimize in the physical design.

We define *task embodiment* as the amount of information about a task encoded in a robot's motion (not to be confused with embodiment found in human-robot interaction [14], [29]). The focus is on how much information a body encodes about a task, so that the design update can be characterized in terms of moving task information from the centralized computations in the control calculations to embedded computation in the physical body. This can be evaluated using the K-L divergence (2) between a distribution representing the task, P, and a distribution representing trajectories of the robot design, Q [5],

$$D_{KL}(P\|Q) = -\sum_i P(i)\log\left(\frac{Q(i)}{P(i)}\right). \tag{2}$$

To define the goal task distribution P, a model predictive controller (MPC) is used to simulate the task execution of an idealized robot (e.g., with a computer on board and perfect state measurements). The same method is used to generate a distribution Q that represents the robot design—this time implementing the generated control policy in place of the optimal controller. We use (2) to compare these two distributions: a low measure of K-L divergence indicates that the distributions are similar, and implies a high level of task embodiment, and therefore a better robot design.

In other words, if a task is well-embodied by a robot, only a simple control policy is necessary to execute it. Otherwise, more information, in the form of a more complex control policy, is required.

5 Control Modes Based on State

We split the design synthesis problem into two stages. First, in this section, the control policy is optimized as a function of state, keeping the number of transitions between control modes low. Then, in the next section, those control modes are projected onto sensor states to generate a design. The min-max optimization of minimizing implementation complexity while maximizing task embodiment is challenging, with many reasonable approaches. We use techniques from hybrid optimal control because of properties described next.

It was proven in [4] that optimal control of switched systems will result in a chattering solution with probability 1. Chattering is equivalent to switching control modes very quickly in time. In the case of these control policies, this translates to switching between control modes very quickly in state. As a result, an implementation of the optimal control policy would be highly complex. Instead of an optimal solution, we are looking for a "good enough," near optimal solution that results in a minimal amount of mode switching. It was shown in [7] that the mode insertion gradient (MIG), which will be discussed in Section 5.2, has useful properties in this regard, including that when the MIG is negative at a point, it is also negative for a region surrounding that point, and that a solution that switches modes slowly can be nearly as optimal as a chattering solution.

This section will first review the topics of switched systems [4], [7], [12], [30] and the use of needle variations for optimization [11], [27], [30], then develop an algorithm for building low complexity control policies. The algorithm creates a simple control policy under the assumption that the system has perfect knowledge of its state. Mapping this policy to physically realizable sensors is the subject of Section 6.

5.1 Switched Systems

A switched-mode dynamical system is typically described by state equations of the form

$$\dot{x}(t) = \{f_\sigma(x(t))\}_{\sigma\in\Sigma} \tag{3}$$

with n states $x:\mathbb{R}\to X\subseteq\mathbb{R}^n$, m control modes[1] $\sigma:\mathbb{R}\to\Sigma\subset[0,m]$, and continuously differentiable functions $\{f_\sigma:\mathbb{R}^{n+m}\to\mathbb{R}^n\}_{\sigma\in\Sigma}$ [11]. Such a system will switch between modes a finite number of times N in the time interval $[0,T]$. The control policy for this type of switched system often consists of a mode schedule containing a sequence of the switching control modes $\mathcal{S}=\{\sigma_1,...,\sigma_N\}$ and a sequence of increasing switching times $\mathcal{T}=\{\tau_1,...,\tau_N\}$ [12], [30].

In this paper, we will consider a similar switched-mode system in that the system has n states $x=[x_1,x_2,...,x_n]^T\in\mathbb{R}^n$ and m control modes $\sigma=[\sigma_1,\sigma_2,...,\sigma_m]^T\in\mathbb{R}^m$, but instead of implementing an algorithm to optimize transition *times* between modes, we optimize transition *states*, such that a robot can directly map sensory measurements of state to one of a finite number of control outputs.

[1] Typically u is denoted as a control variable, but in this case the control value u could be anything in the greater context of the control mode σ. The control mode may, in fact, consist of many different values of u, which is why u does not appear in a significant way in the posing of this problem.

5.2 Hybrid Optimal Control

Let $L:\mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable cost function, and consider the total cost J, defined by

$$J = \int_0^T L(x(t))dt. \tag{4}$$

We use the Mode Insertion Gradient (MIG) [11], [27], [30] to optimize over the choice of control mode at every state. The MIG measures the first-order sensitivity of the cost function (4) to the application of a control mode σ_i for an infinitesimal duration $\lambda \to 0^+$. The MIG $d(x)$ is evaluated using

$$d(x) = \left.\frac{dJ}{d\lambda^+}\right|_t = \rho(t)^T(f_{\sigma_i}(x(t)) - f_{\sigma_0}(x(t))), \tag{5}$$

where the adjoint variable ρ is the sensitivity of the cost function

$$\dot{\rho} = -(\frac{\partial f_{\sigma_0}}{\partial x}(x(t)))^T\rho - (\frac{\partial L}{\partial x}(x(t)))^T, \quad \rho(T) = 0. \tag{6}$$

The derivation of these equations is discussed in [11], [27], [30], but the key point is that $d(x)$ measures how much inserting a mode σ_i locally impacts the cost J. When $d(x) < 0$, inserting a mode at state x will decrease the cost throughout a volume around x, meaning a descent direction has been found for that state. The MIG can be calculated for each mode so that $d(x)$ is a vector of m mode insertion gradients: $d(x) = [d_1(x),...,d_m(x)]^T$. Therefore the best actuation mode (i.e., the mode with the direction of maximum descent) for each state x has the most negative value in the vector $d(x)$.

As long as the dynamics $f(x(t))$ are real, bounded, differentiable with respect to state, and continuous in control and time and the incremental cost, $L(x(t))$, is real, bounded, and differentiable with respect to state, the MIG is continuous [7]. Sufficient descent of the mode insertion gradient is proven in [7], where the second derivative of the mode insertion gradient is shown to be Lipschitz continuous under assumptions guaranteeing the existence and uniqueness of both x, the solution to the state equation Eq. (3), and ρ, the solution of the adjoint equation Eq. (6). Combining this with the results of [15], one can conclude that any sufficiently dense finite packing will also satisfy the descent direction throughout the volume of packing. As a result, although chattering policies may be the actual optimizers, finite coverings will generate descent throughout the state space, resulting in a non-optimal but "good enough" solution. This property provides the required guarantee that we can locally control the complexity of the policy as a function of state. This will be discussed further in Sec. 5.3.

Figure 3 illustrates differences in complexity as a result of optimizing using the mode insertion gradient. The magnitude of the curves is the default control (the control we are comparing to) minus the step size (a scaling factor, and also the line search parameter) multiplied by the MIG. Therefore the magnitude of these plots correspond to the amount of reduction in cost that can be achieved by locally employing each control mode at the state x. The complex policy illustrated in Fig. 3 (c), occurs in simulation in the chattering policy of Fig. 4. This happens when there is similar utility in employing more than one mode in a region—there is only marginal benefit in choosing one control mode over another, which results in increased complexity.

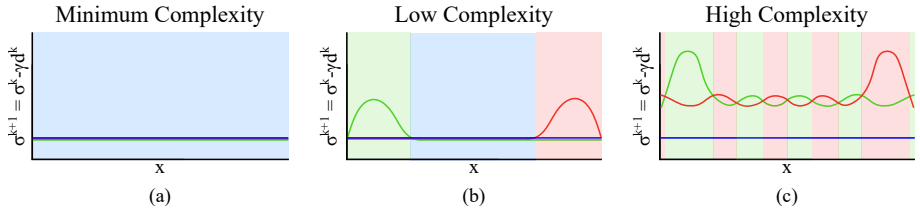

Fig. 3: The curves show $\sigma^{k+1}=\sigma^k-\gamma d^k$ for three different control modes σ_1 (blue), σ_2 (green), and σ_3 (red), where γ is the line search parameter and the background colors indicate which mode is assigned to state x in the control policy. As step size γ increases from left to right, the magnitude of $\gamma d^k(x)$ surpasses that of the default control $\sigma^k(x)$ (here the default control is $\sigma_1=0$). (a) Minimum Complexity: Only one control mode is assigned throughout the entire state space. (b) Low Complexity: A few control modes are employed, indicating that the cost function can be reduced by including these extra control modes. (c) High Complexity: The control mode switches often but the magnitudes of the values are very similar, indicating that there is no significant decrease in cost between these modes—despite the large increase in complexity (i.e., chattering).

5.3 Iterative Algorithm

An algorithm is introduced that can reduce the complexity of a control policy in as little as one iteration, based on the work in [6], [7].

Algorithm 1 Iterative Optimization

$k=0$
Choose default policy $\sigma^0(x)$
Calculate initial cost $J(\sigma^0(x))$
Calculate initial complexity h^0
Calculate initial descent direction $d^0(x)$
$h^{k-1}=\infty$
while $h^k<h^{k-1}+\epsilon_h$
  while $J(\sigma^k(x))<J(\sigma^{k+1}(x))+\epsilon_J$
    Re-simulate $\sigma^{k+1}(x)=\sigma^k(x)-\gamma d^k(x)$
    Compute new cost $J(\sigma^{k+1}(x))$
    Increment step size γ
  end while
  Calculate new complexity h^{k+1}
  Calculate $d^{k+1}(x)$
  $k=k+1$
end while

Note that the superscripts in this algorithm are not exponents, but indices of loop increments.

In this line search algorithm $\sigma=[\sigma_1,\sigma_2,...,\sigma_m]^T$ is a vector of all control modes and descent direction $d(x)=[d_1(x),d_2(x),...,d_m(x)]^T$ is the gradient of inserting any mode with respect to a default mode $\sigma^0(x)$ at a particular state. The default control

may be chosen arbitrarily, but for simplicity we will show an example using a null default policy (in Fig. 5).

The cost $J(\sigma^k(x))$ of the entire policy is approximated by simulating random initial conditions forward in time and evaluating the total cost function for time T. We use cost J rather than task embodiment D_{KL} as the objective function because the line search in the algorithm is a function of $d(x) = \frac{dJ}{d\lambda}$. This way, the algorithm decreases the objective function (plus tolerance ϵ_J) each iteration. After choosing a default policy $\sigma^0(x)$ (which can be anything from a constant mode throughout the state space to a highly complex policy), computing the initial cost $J(\sigma^0(x))$, and calculating the initial entropy h^0 (using Eq. (1)) the initial descent direction $d^0(x)$ is calculated for the set of points in S, as described in Sec. 5.2. A line search [2] is performed to find the maximum step size γ that generates a reduction in cost in the descent direction $d^k(x)$, and then the policy $\sigma^k(x)$ is updated to the policy $\sigma^{k+1}(x)$. The new design complexity h^{k+1} and descent directions $d^{k+1}(x)$ are calculated, and this is repeated until the cost can no longer be reduced without increasing the complexity beyond the threshold defined by ϵ_h.

The tolerances ϵ_h and ϵ_J are design choices based on how much one is willing to compromise between complexity and performance. In the example illustrated in Sec. 5.4, the value for ϵ_h is significant because it represents the allowable increase in complexity—how much complexity the designer is willing to accept for improved task embodiment.

This algorithm enforces low design complexity, meaning it will not result in chattering outputs. The work in [7] showed that if $d(x(\tau)) < 0$ then there exists an $\epsilon > 0$ such that $d(x(t)) < 0 \ \forall \ t \in [\tau - \epsilon, \tau + \epsilon]$. Since $d(x)$ is continuous in x (as discussed in Section 5.2), $d(x_0) < 0$ implies that there exists an $\epsilon > 0$ such that $d(x) < 0 \ \forall \ x \in B_\epsilon(x_0)$. Note that each point in $B_\epsilon(x_0)$ does not necessarily have the same mode of *maximum* descent, but they do each have a *common* mode of descent.

The MIG serves as a descent direction for a volume in the state space, rather than just at a point. This property allows us to assign one control mode throughout a neighborhood so that instead of choosing the optimal control mode (the direction of maximum descent) at each point and causing chattering, we select a good control mode (a direction of descent) throughout a volume and maintain relative simplicity in the policy. Figure 4 shows a control policy that is the result of assigning the *optimal* control mode at each point, which results in chattering.

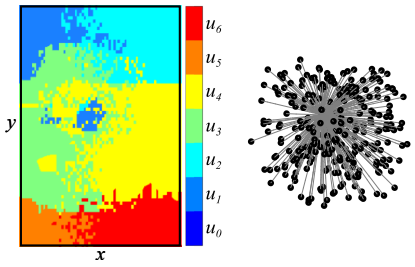

Fig. 4: A chattering control policy. The corresponding graph has entropy $h = 7.6035$.

5.4 Examples

For this example, the configuration space of the synthetic cell is defined by its position in two dimensional space (x,y). We define the control authority as the ability to be attracted toward a specific chemical potential. So at any location (x,y) the robot may choose a control mode $\sigma \in \{\sigma_0,\sigma_1,\sigma_2,\sigma_3,\sigma_4,\sigma_5,\sigma_6\}$, where σ_0 is zero control, and the other six modes are a potential (with dynamics $\ddot{x} = \frac{1}{r_n^2}\frac{s_n - x}{||s_n - x||}$, where r_n is the distance from the synthetic cell to source n and s_n is the location of source n). The control synthesis problem is to schedule σ in space, based on an objective (in this case, to approach a point P)[2]. The state space, desired point, and chemical sources are all shown in Fig. 2 (a).

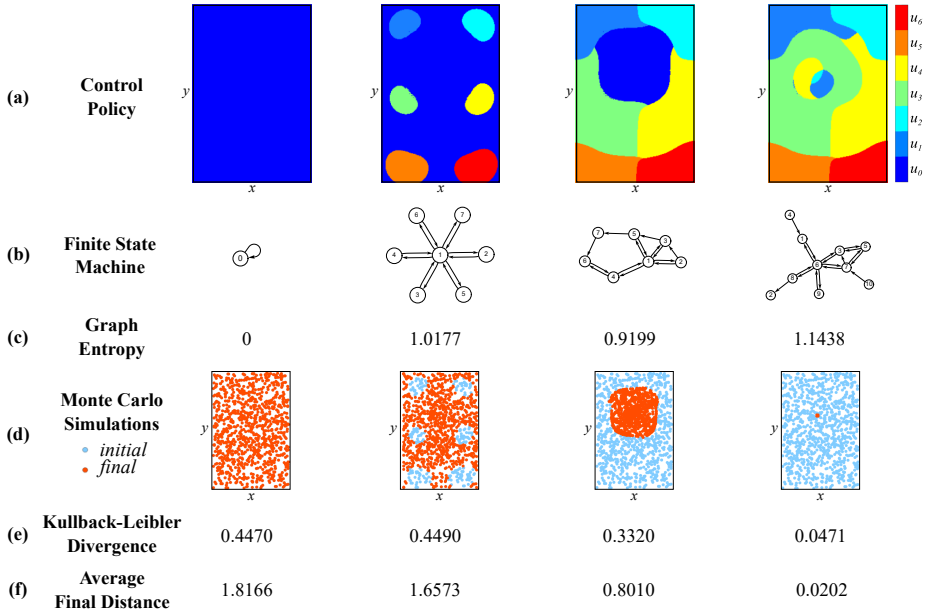

Fig. 5: Control policies for the system in Fig. 2, starting from a null initial policy. (a) The control policies, mapping state to control for various iterations of the line search. (b) A finite state machine representation of each policy, representative of the complexity of the system. (c) The design complexity value calculated from Eq. (1). (d) 1000 Monte Carlo simulations illustrate the results of random initial conditions using the associated control policies. (e) The Kullback-Leibler divergence between the goal task distribution and the distribution generated by the control policy, from Eq. (2). (f) The average final distance of the final states of (d) from the desired point P.

[2] Point P is slightly off center, to avoid adverse effects of symmetry. This is reflected in the asymmetry of the resulting control policies (e.g. red and orange not being perfectly even).

Figure 5 begins with an initial control policy of zero control throughout the state space, and increases design complexity and task embodiment until the line search algorithm converges to a new control policy. Monte Carlo simulations were performed with 1000 random initial conditions, shown in row (d) and the average distance of the final points from the desired point is shown in row (f). Most interesting are the trends in rows (c) and (e). These correspond to the min-max problem posed earlier, in which we attempt to minimize design complexity, computed using graph entropy (c), and maximize task embodiment, calculated using K-L divergence (e). The graph entropy in row (c) increases as the K-L divergence in row (e) increases. This shows the entropy must increase (from 0) to ensure some amount of task embodiment.

Synthetic cells can encode these simplified control policies by physically combining their movement, sensory, and memory elements with a series of logical operators, as discussed next in Section 6.

6 Projecting Policies onto Discrete Sensors

Section 5 described synthetic cells with perfect state measurement. In this section, implementations using discrete sensors will be explored. In some cases, it may be possible to *create* sensors that are able to detect exactly where a robot should switch between control modes (e.g. a sensor that can perfectly sense the boundary between the green and orange regions of the control policy). It is also possible that a designer may start with a *fixed* library of sensors, in which case the state space should first be divided into sensed regions, and then control modes should be assigned. Probably the most likely case, and the one we will examine in the section, is that a designer has a variety of sensors to choose from, and will want to use some subset of them.

For the synthetic cell example, we will assume discrete sensing provided by a chemical comparator—a device that compares the relative strength of two chemical concentrations. From a given library of sensors, how should the combination of sensors, actuators, and logical operators be chosen so that the task is best achieved?

Figure 6 (a) shows five different individual sensors: each comparing the strength of chemical source 1 to another of the chemical sources in the environment, and how each of these sensors is able to divide the state space into two distinct regions, while Fig. 6 (b) illustrates which regions of the state space are able to be discerned using these 5 sensors *combined*. Fig. 6 (c) shows all possible combinations of comparators: all 6 chemical sources compared to each of their 5 counterparts, and therefore the maximum granularity of sensed regions in the state space using this sensor library[3].

The optimal scenario would be that these sensor regions correspond perfectly to the control regions found using the iterative algorithm in Fig. 5. Since this will almost never be the case, we must attempt to approximate our control policy using the library of sensors.

[3] Note that some comparators divide the state space in the exact same way, i.e. comparing chemical sources 1 and 3 results in the same sensed regions as comparing sources 2 and 4 (this is true for three other sets of comparators: $1/2 = 3/4 = 5/6$, $1/5 = 2/6$, and $3/5 = 4/6$).

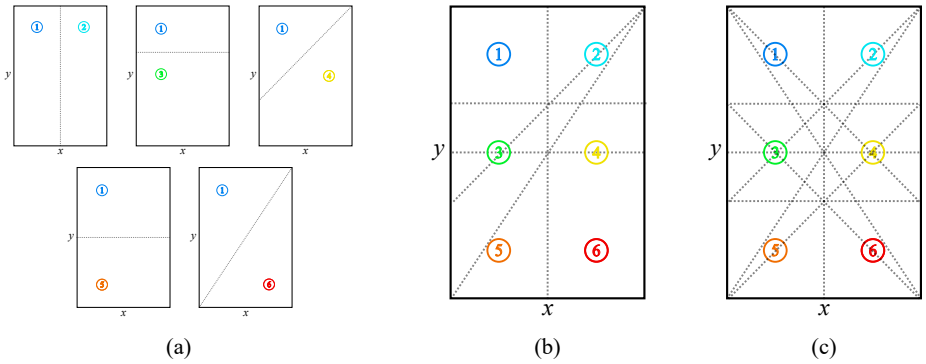

Fig. 6: (a) Illustration of five individual chemical comparators, each comparing the chemical potential of Source 1 with one of the other sources. Each sensor can tell whether the robot is on one side of an equipotential—the dotted lines—or the other. (b) Combination of the five sensors. The robot is able to sense which of these 9 regions it is in. (c) Sensor regions resulting from the combination of all possible synthetic cell comparators in this environment.

Figures 7 and 8 demonstrate how synthesized control policies combined with a library of sensors create a physical design. Figure 7 (a) shows two comparators chosen from the sensor library and how they each divide the state space into sensed regions and Fig. 7 (b) is the policy that results from projecting the final control policy found in the algorithm onto the feasible sensor space. This projection is done by dividing the state space into k different sensed regions $\mathcal{R}=[\mathcal{R}_1,...,\mathcal{R}_i,...,\mathcal{R}_k]$ and finding the volumes $v=[v_1,...,v_j,...,v_m]$ occupied by each of the m control modes in the policy. In the cases of Figures 7 and 8, k is equal to 4 and 32, respectively, and v is the n-dimensional volume corresponding to each of the 7 control modes. The control mode σ with the maximum volume in each region $\mathcal{R}_i$ is assigned to that region using Eq. (7).

$$\sigma_{\mathcal{R}_i} = \underset{j}{\text{argmax}}(v_j \cap \mathcal{R}_i) \tag{7}$$

Logical operators can be combined with sensory observations to represent the state space with more fidelity than sensors alone (e.g. a single sensor in Fig. 6(a))—so that actions can be associated with *combinations* of sensory observations (e.g. Fig. 6(b)). Figure 7 (c) illustrates the logical diagram that would be physically encoded in circuitry onto a synthetic cell so that the policy in Fig. 7 (b) could be executed.

Figure 8 is similar to Fig. 7, but illustrates the physical design corresponding to the highest fidelity control policy from the library of comparator sensors. Fig. 8 (a) shows each of the sensors in the library, including the ones that repeat sensed regions due to the symmetry in this environment. The projected control policy is shown in Fig. 8 (b) and Fig. 8 (c) illustrates the logic of the physical circuitry.

It is notable that the designs in Figures 7 and 8 are quite dissimilar. Figure 9 shows how each of the physically feasible designs compare to each other, to another

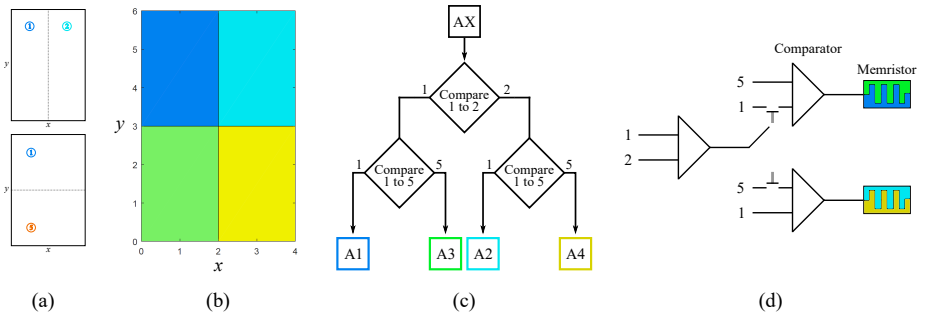

Fig. 7: Low-fidelity Design. (a) Two sensors. Top: comparing chemical Sources 1 and 2 divides the state space into left and right. Bottom: comparing Sources 1 and 5 divides the space into top and bottom. (b) Control policy from Fig. 5 projected onto the sensed regions. (c) Logical decision diagram for this system. (d) Circuit diagram for physical synthetic cell design.

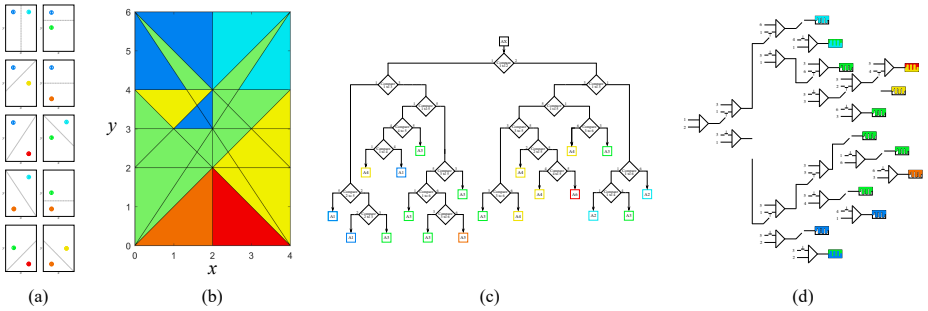

Fig. 8: High-fidelity Design. (a) Ten sensors. The equipotential lines demonstrate how the device can use chemical comparators to estimate its location in the environment. (b) Control policy from Fig. 5 projected onto the sensed regions. (c) Logical decision diagram for this system. (d) Circuit diagram for physical synthetic cell design.

physically feasible design, and to the control policy with perfect knowledge of state. The high-fidelity design in the middle of Fig. 9 captures much of the structure of the sensor-agnostic policy, and the results are evident in the relatively low K-L divergence. The medium-fidelity design uses fewer sensors than the high-fidelity one and therefore does not embody the task quite as well. The low-fidelity design has a higher K-L divergence than even the null policy in Fig. 5. This is because during the projection, the control mode assigned to each sensor region corresponded to a chemical potential that was inside that same region, ensuring that no simulated synthetic cell was ever able to escape its quadrant. The fact that all three of these designs were projected from the same original control policy demonstrates that, as expected, the success of the resulting robot design is highly sensitive to choice of sensors.

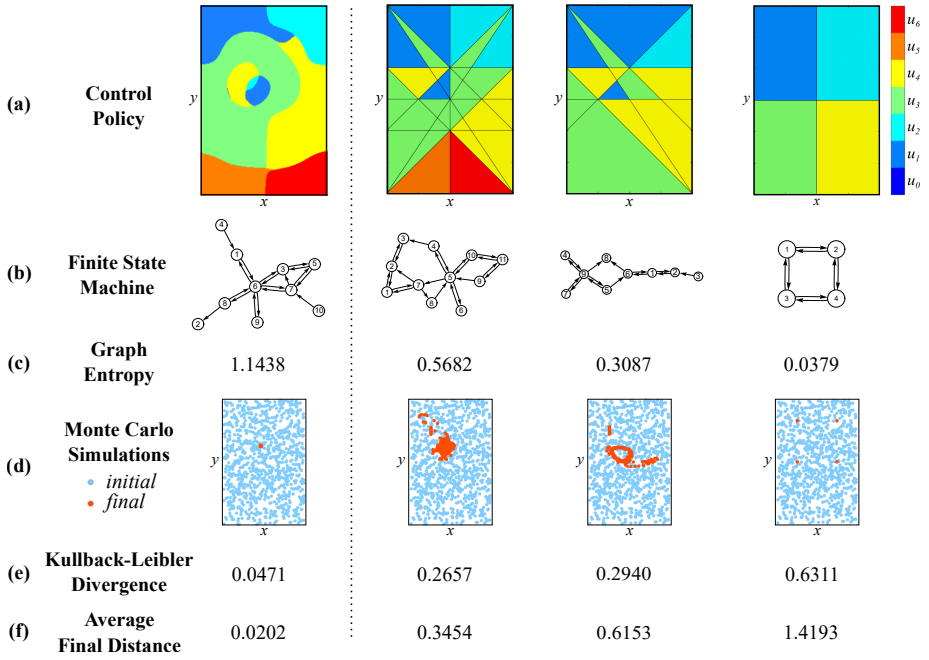

Fig. 9: Left: The policy generated when perfect knowledge of state was assumed. Right: A high-fidelity design using all of the (10 distinct) sensors in the sensor library, as shown in Fig. 8. A medium-fidelity design, using five sensors. A low-fidelity design, using only two of the sensors from the sensor library, as shown in Fig. 7.

7 Discussion

In this work we addressed the question of designing robots while minimizing complexity and maximizing task embodiment. We demonstrated our method of solving this min-max problem, which included an iterative algorithm resulting in a control policy assuming perfect sensing, and then projecting that policy onto a discrete space of sensed regions resulting from a library of sensors. This is not necessarily an optimal design pipeline for all robot design problems. In some cases it may be possible to find a simple control policy assuming perfect sensing, and then create sensors that best align with that policy. Conversely, there may be some instances where there is a fixed library of sensors, in which case one would first divide the state space into discrete regions and then find the optimal control mode in each of those regions.

In future work, this algorithm will be implemented for a wider range of dynamical systems, specifically higher order systems. The algorithm will be tested with different modifications, including using D_{KL} in the objective function so that task embodiment is the actual object of the optimization, rather than a correlated consequence. Finally, this methodology will be validated by using control policies computed by this method to design and create actual, physical synthetic cells.

Bibliography

[1] K. Anand and G. Bianconi. Entropy measures for networks: Toward an information theory of complex topologies. *Physical Review E*, Oct 2009.

[2] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, pages 1–3, 1966.

[3] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, pages 61–70, Mar 2007.

[4] S. C. Bengea and R. A. DeCarlo. Optimal control of switching systems. *Automatica*, pages 11–27, Jan 2005.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] T. M. Caldwell and T. D. Murphey. Projection-based optimal mode scheduling. In *IEEE Conference on Decision and Control*, Dec 2013.

[7] T. M. Caldwell and T. D. Murphey. Projection-based iterative mode scheduling for switched systems. *Nonlinear Analysis: Hybrid Systems*, pages 59–83, Aug 2016.

[8] A. Censi. A mathematical theory of co-design. *ArXiv e-prints https://arxiv.org/abs/1512.08055*, 2015.

[9] S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, pages 1082–1085, 2005.

[10] M. Dehmer and A. Mowshowitz. A history of graph entropy measures. *Journal of Information Science*, pages 57–78, Jan 2011.

[11] M. Egerstedt, Y. Wardi, and H. Axelsson. Transition-time optimization for switched-mode dynamical systems. *IEEE Transactions on Automatic Control*, pages 110–115, Jan 2006.

[12] M. Egerstedt, Y. Wardi, and H. Axelsson. Optimal control of switching times in hybrid systems. In *International Conference on Methods and Models in Automation and Robotics, Miedzyzdroje*, 2003.

[13] B. Esteban-Fernández de Ávila, P. Angsantikul, D. E. Ramírez-Herrera, F. Soto, H. Teymourian, D. Dehaini, Y. Chen, L. Zhang, and J. Wang. Hybrid biomembrane functionalized nanorobots for concurrent removal of pathogenic bacteria and toxins. *Science Robotics*, 2018.

[14] U. Huzaifa, C. Bernier, Z. Calhoun, G. Heddy, C. Kohout, B. Libowitz, A. Moenning, J. Ye, C. Maguire, and A. LaViers. Embodied movement strategies for development of a core-located actuation walker. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 176–181, June 2016.

[15] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, pages 846–894, 2011.

[16] J. Li, O. E. Shklyaev, T. Li, W. Liu, H. Shum, I. Rozen, A. C. Balazs, and J. Wang. Self-propelled nanomotors autonomously seek and repair cracks. *Nano Letters*, pages 7077–7085, 2015.

[17] S. Li, Q. Jiang, S. Liu, Y. Zhang, Y. Tian, C. Song, J. Wang, Y. Zou, G. J. Anderson, J. Han, Y. Chang, Y .Liu, C. Zhang, L. Chen, G. Zhou, G. Nie, H. Yan, B. Ding, and Y. Zhao. A DNA nanorobot functions as a cancer therapeutic in response to a molecular trigger in vivo. *Nature Biotechnology*, pages 258–264, 2018.

[18] L. Liebenwein, C. Baykal, I. Gilitschenski, S. Karaman, and D. Rus. Sampling-based approximation algorithms for reachability analysis with provable guarantees. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2018.

[19] P. Liu, A. T. Liu, D. Kozawa, J. Dong, J. F. Yang, V. B. Koman, M. Saccone, S. Wang, Y. Son, M. H. Wong, and M. S. Strano Autoperforation of 2D materials for generating two-terminal memristive Janus particles. *Nature Materials*, pages 1005–1012, 2018.

[20] P. Liu, A. L. Cottrill, D. Kozawa, V. B. Koman, D. Parviz, A. T. Liu, J. F. Yang, T. Q. Tran, M. H. Wong, S. Wang, and M. S. Strano. Emerging trends in 2D nanotechnology that are redefining our understanding of "nanocomposites". *Nano Today*, 2018.

[21] A. Mavrommati and T. D. Murphey. Automatic synthesis of control alphabet policies. In *IEEE International Conference on Automation Science and Engineering (CASE)*, Aug 2016.

[22] V. N. Murthy Arelekatti and A. G. Winter, V. Design and preliminary field validation of a fully passive prosthetic knee mechanism for users with transfemoral amputation in india. *Journal of Mechanisms and Robotics*, pages 350–356, 2015.

[23] J. M. O'Kane and S. M. LaValle. Comparing the power of robots. *The International Journal of Robotics Research*, pages 5–23, Jan 2008.

[24] J. M. O'Kane and D. A. Shell. Concise Planning and Filtering: Hardness and Algorithms. *IEEE Transactions on Automation Science and Engineering*, pages 1666–1681, Oct 2017.

[25] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, pages 2508–2516, 2008.

[26] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, pages 206–230, 1987.

[27] M. Shahid Shaikh and P. E. Caines. On the optimal control of hybrid systems: Optimization of trajectories, switching times, and location schedules. *Hybrid Systems: Computation and Control*. Springer Berlin Heidelberg, 2003.

[28] R. Tedrake, T. W. Zhang, M. Fong, and H. S. Seung. Actuating a simple 3D passive dynamic walker. In *IEEE International Conference on Robotics and Automation*, Apr 2004.

[29] J. Wainer, D. J. Feil-Seifer, D. A. Shell, and M. J. Mataric. Embodiment and human-robot interaction: A task-based perspective. In *IEEE International Symposium on Robot and Human Interactive Communication*, Aug 2007.

[30] X. Xu and P. J. Antsaklis. Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. *International Journal of Control*, pages 1406–1426, 2002.