Planning for Gene Regulatory Network Intervention

Daniel Bryce & Seungchan Kim
Department of Computer Science and Engineering
Arizona State University, Brickyard Suite 501
699 South Mill Avenue, Tempe, AZ 85281
{dan.bryce, dolchan}@asu.edu

Abstract

Modeling the dynamics of cellular processes has recently become a important research area of many disciplines. One of the most important reasons to model a cellular process is to enable high-throughput *in-silico* experiments that attempt to predict or intervene in the process. These experiments can help accelerate the design of therapies through their cheap replication and alteration. While some techniques exist for reasoning with cellular processes, few take advantage of the flexible and scalable algorithms popularized in AI research. In this domain, where scalability is crucial for feasible application, we apply AI planning based search techniques and demonstrate their advantage over existing enumerative methods.

1 Introduction

The cell maintains its functions via various interconnections and regulatory controls among genes and proteins. Therefore, it is critical, for understanding the living cell, to unravel how such cellular components as genes and proteins interact with each other. Mathematical modeling and computational simulation of cellular systems, especially gene regulatory systems, has been crux of computational systems biology, or biomedical science in general. Once a model is constructed, it can be used to predict the behavior of a cellular system under unusual conditions, to identify how a disease might develop, and/or how to intervene such development to prohibit cells from reaching undesirable states. Such models can aid biologists by quickly ruling out or confirming simple hypotheses before expensive and time-consuming wet lab experiments.

In this paper, we address the problem of planning to intervene in cellular processes, focusing on gene regulatory networks (GRNs). A GRN describes a cellular process by a set of genes and their regulatory influences over each other. GRNs focus solely on genes (omitting proteins or other molecules) to model the high level behavior of many genes versus the low level behavior of a much smaller system. Because practical GRN models are typically learned from micro-array data [Kim *et al.*, 2000], they are situated at the right level of granularity for automated parametrization. Micro-array experiments measure the activity level of thousands of genes from living tissue in terms of mRNA concentrations (the products of gene transcription used to code proteins). Correlations between observed gene activity levels help describe regulatory influences. Predictor functions characterize the regulatory influences and provide a dynamic model (e.g., when g_1 and g_2 are highly active, g_3 becomes inactive). The GRN state models the activity levels of genes and the predictor functions describe possible next states. Outside interventions (e.g., using RNA interference to suppress a gene's activity level) alter predictor functions to control the dynamics of the GRN, providing a natural planning problem. In a plan, each action models either a possible intervention or non-intervention that will change the state of the GRN.

There are a number of GRN features that affect our choice of AI planning model. First, intervention plans need only focus on horizons long enough to ensure the GRN will naturally transition to nominal states. Biological knowledge and computational simulation of GRNs [Kim *et al.*, 2002] tell us that cellular processes, left to their own, transition to (or through) stable attractor states. These states represent common cellular phenomena such as the cell cycle, division, etc. However, some states are consistent with disease, such as the metastasis of cancer. From abnormal states, planning interventions provides a method to push the evolution of the cell toward nominal attractor states. Second, the cell has an inherently hidden state because full observations are prohibitively costly and we have limited accessability [Datta *et al.*, 2004]. Planning with partial observability is important because biologists analyzing cellular processes cannot be expected to understand, nor obtain complete state information. Third, cellular processes are commonly viewed as stochastic [Rao *et al.*, 2002]. Genes are typically regulated many different ways, meaning GRNs must allow for the probabilistic selection of predictor function for each gene. Because state transitions are stochastic, provide only partial observations, and are only relevant over a limited planning horizon, we formulate our model of GRN intervention as decision theoretic planning in a finite horizon partially observable Markov decision process (POMDP).

This work is not the first to address the problem of planning interventions for GRNs, but it is the first to apply AI planning techniques. Existing research [Datta *et al.*, 2004] enumerates all possible plans and then uses a dynamic programming algorithm to find the optimal finite horizon plan. Our planner is based on the AO* algorithm [Nilsson, 1980],

which uses upper bounds on plan quality (reward) for pruning. We evaluate our planner on several formulations of a random GRN with different reward functions to see the benefit pruning. We also evaluate on the WNT5A GRN [Weeraratna *et al.*, 2002], which [Datta *et al.*, 2004] use for intervention planning. WNT5A is a gene that plays a significant role in the development of melanoma and is known to induce the metastasis of melanoma when highly active. On the WNT5A problems studied by [Datta *et al.*, 2004], our planner performs significantly better than the enumeration algorithm of [Datta *et al.*, 2004]. Our study formalizes this interesting planning domain as a finite horizon POMDP and shows the gains of applying relatively standard AI techniques.

We start by describing a simple example of planning interventions in a GRN. We then briefly define the components of a finite horizon POMDP and a GRN intervention problem. With these definitions, we map the intervention problem to the POMDP. The solution to the POMDP is a conditional plan, for which we define the semantics and provide two solution algorithms – AO* and enumeration [Datta *et al.*, 2004]. We compare both algorithms on several GRNs, either randomly generated or learned from actual micro-array experiments. We end with related work, a conclusion, and discussion of future work.

2 Intervention Planning Example

To clarify intervention planning, consider a small two gene network where each gene g is either active (g) or inactive ($\neg g$). There are two predictor functions $\{f_{g_1}, f_{g_2}\}$ that describe the GRN dynamics, and an intervention to suppress g_2 described by the predictor function f'_{g_2}. Using the predictor functions, we can define the following two actions:

Action	No Intervention	g_2 Intervention
Reward	0	-1
Effects (Predictors)	$f_{g_1} : g_2 \rightarrow \neg g_1, \neg g_2 \rightarrow g_1$ $f_{g_2} : \neg g_2 \rightarrow \neg g_2, g_2 \rightarrow g_2$	$f_{g_1} : g_2 \rightarrow \neg g_1, \neg g_2 \rightarrow g_1$ $f'_{g_2} : \rightarrow \neg g_2$
Observation	g_2 / $\neg g_2$	g_2 / $\neg g_2$

Intervening to suppress g_2 incurs a reward of -1 by rewriting the predictor function f_{g_2} with f'_{g_2}. We also assume that we can observe g_2 but not g_1. The goal of the intervention problem associates a reward of 10 with activating gene g_1. While the actions are deterministic (for the sake of example simplicity), we can describe stochastic actions by allowing a probabilistic choice of predictor function for each gene.

Assume that we start with the initial belief state where each GRN state is equally likely, as depicted in Figure 1. The figure shows a horizon three plan to achieve the goal of activating g_1. The first action in the plan is to not intervene; we will let the genes affect each other, and then observe g_2. Not intervening leads to the belief state $\{0.5\{\neg g_1, g_2\}, 0.5\{g_1, \neg g_2\}\}$. Using the observation of whether g_2 or $\neg g_2$ holds, the plan branches to belief states consistent with the appropriate observation. Since we do not know which value of g_2 will occur, we plan for both branches. In the first branch, we apply our g_2 intervention action to suppress g_2. In the following step, we do not intervene and having g_2 inactive will activate g_1, leading to a state satisfying the goal. In the second branch, we can apply the non intervention action indefinitely because the goal is satisfied and will remain satisfied.

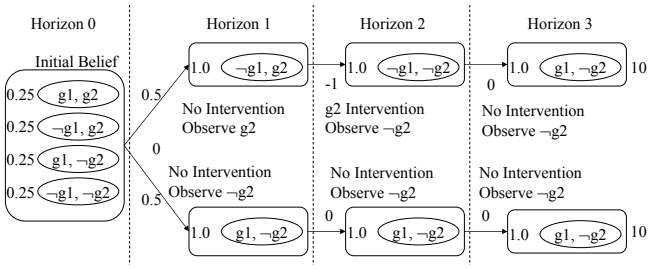

Figure 1: Example Plan.

The quality of the plan is the expected reward of all possible plan branches. The first branch, taken with 0.5 probability has a reward of 9 (because we had to intervene), and the second branch, taken with 0.5 probability has a reward of 10. Thus, the total expected reward is 9.5.

3 Decision Theoretic Planning

This section describes the basics of the finite horizon POMDP model, formally defines intervention planning, and details the formulation of intervention planning in the POMDP.

POMDP Model: Unlike most work on POMDPs, that find a policy over the belief space, we find a policy (conditional plan) rooted at an initial situation. The finite horizon POMDP problem P is defined as the tuple $\langle S, A, T, R, \Omega, O, h, b_I \rangle$, where S is a set of states, A is a set of actions, $T : S \times A \cup \{\perp\} \times S \cup \{\perp\} \rightarrow [0, 1]$ is a transition function, $R : S \times A \cup \{\perp\} \times S \cup \{\perp\} \rightarrow \mathbb{R}$ is the transition reward function, Ω is a set of observations, $O : S \times A \times \Omega \rightarrow [0, 1]$ is the observation function, h is the planning horizon, and $b_I : S \rightarrow [0, 1]$ is the initial belief state. We overload the symbol $\perp$ to denote both the terminal action and state signifying the end of the plan.

Intervention Planning Problem: Given our definition of the POMDP model, in the following we show how an intervention planning problem maps to the POMDP. The intervention problem is defined by the tuple $\langle \mathcal{G}, Dom, F, X, W, Y, \mathcal{O}, h \rangle$, where $\mathcal{G}$ is a set of genes, Dom is the set of activity levels for genes, F is a set of predictor functions, X is a set of interventions, W is an initial situation, Y is a goal description, $\mathcal{O}$ is a set of observations, and h is the horizon. The genes and their activity levels describe states of the POMDP, the predictor functions and interventions describe actions, interventions and the goal description define the reward function, and the initial situation, observations, and horizon map directly to their POMDP counterparts.

States: Each gene $g \in \mathcal{G}$ has an activity level from the domain Dom of values, of which we will only illustrate boolean domains $\{g, \neg g\}$, active or inactive. A state $s : \mathcal{G} \rightarrow Dom$ of the gene network maps each gene to a value $d \in Dom$. The entire set of GRN states defines the POMDP states S.

Predictor Functions and Interventions: Given a state of the gene network, the predictor functions F describe states reachable after one step. Interventions re-write predictor functions in F for specific genes to ensure the gene network transitions to specific states. Thus, each possible action in the POMDP

is described by a set of predictor functions. A non intervention simply uses F to describe the action, but an intervention action $x \in X$ replaces predictor functions in F to get a new set F_x. Each intervention $x \in X$ is a set of predictors $\{f_{g_1}, f_{g_2}, ...\}$, allowing us to define

$$F_x = F \cup x \backslash \{f_g | f_g \in F, f_{g'} \in x, g = g'\}.$$

Each predictor function f_g is defined as the mapping $f_g : Dom^{|\mathcal{G}'|} \rightarrow Dom$ from activity levels of genes in $\mathcal{G}' \subseteq \mathcal{G}$ to the activity level of gene g. The interventions described in this work contain a single predictor function f_g where $\mathcal{G}' = \emptyset$, meaning that irrespective of the state g has its activity level set deterministically.

Since each gene may be predicted by several predictor functions, where each state transition selects one probabilistically, we assign a weight $w(f_g)$ to each. While we assume the predictor functions are given, we can learn them from micro-array experiments. In the empirical analysis we evaluate GRNs where the predictor functions and weights are both generated randomly and from real micro-array experiments.

Each action a_F defined by the set of predictor functions F (similarly F_x) describes the transition function probability

$$T(s, a_F, s') = Pr(s'|F, s) = \prod_{g \in \mathcal{G}} \frac{\sum_{f_g \in F : f_g(s) = s'(g)} w(f_g)}{\sum_{f_g \in F} w(f_g)}.$$

Observations: After each step, whether by intervention or non intervention, we can observe the state of the gene network. The set $\mathcal{O} \subseteq \mathcal{G}$ defines which genes are observable (by genetic markers, physiology, etc.). The set of observations $\Omega = \{o | o \in Dom^{|\mathcal{O}|}\}$ is defined by all joint activity levels of genes in $\mathcal{O}$. In this work, we assume that observations are perfect and the same for each action, meaning that if a state s and observation o agree on the activity level of each gene, then the probability of the observation is one (i.e., $O(s, a, o) = 1$), otherwise zero (i.e., $O(s, a, o) = 0$). Observations can be noisy (i.e., $0 \leq O(s, a, o) \leq 1$) in general.

Rewards: The goal Y is a function describing desirable states. We assume that the goal maps states to real values $Y : Dom^{|\mathcal{G}|} \rightarrow \mathbb{R}$. The reward function for terminal actions and goal states is defined by the goal $R(s, \perp, \perp) = Y(s)$. We assume that the reward associated with actions is -1 for intervention actions (i.e., $R(s, a_{F_x}, s') = -1$) and 0 for non intervention (i.e., $R(s, a_F, s') = 0$).

Initial Situation: The initial situation W is a distribution over GRN states $W : Dom^{|\mathcal{G}|} \rightarrow [0, 1]$. This mapping to the POMDP initial situation is straight-forward, $b_I(s) = W(s)$.

Since the formulation has a distinct initial belief state, we do not explore more traditional POMDP value or policy iteration techniques for computing a finite horizon policy. Rather, we use the knowledge of the initial belief state to guide expansion of a conditional plan. By searching forward from the initial belief state, it is possible to focus plan construction on reachable belief states.

4 Search

This section describes our approach to solving a finite horizon POMDP. We start by defining the semantics of conditional plans and our search space. We follow with two search algorithms to find plans. The first algorithm is `AO*` [Nilsson, 1980], and the second `Datta` is based on a competing approach [Datta *et al.*, 2004] from the GRN literature.

Conditional Plans: A solution to the problem P is a conditional plan $\mathcal{P}$ of horizon h, described by a partial function $\mathcal{P} : V \rightarrow A \cup \{\perp\}$ over the belief state space graph $G = (V, E)$. A subset of the vertices $b \in V$ (which are belief states) are mapped to a "best" action a, denoted $\mathcal{P}(b) = a$. Each edge $e \in E$ directed from b to b_a^o is mapped to an action $a \in A$ and an observation $o \in \Omega$, and denoted $e(b, b_a^o) = (a, o)$. If $\mathcal{P}(b) = a$ and $e(b, b_a^o) = (a, o)$, then $\mathcal{P}(b_a^o)$ is the action to execute after executing a and receiving observation o. Throughout our discussion, we assume that the horizon is a feature of every state to ensure that the graph G is acyclic. Belief states where the horizon is equal to h have a single available action $\perp$ to signify the end of the plan, leading to a terminal $\perp$.

If $\mathcal{P}(b) = a$, and there exists an edge $e(b, b_a^o) = (a, o)$ then the successor belief state b_a^o is defined

$$b_a(s') = \sum_{s \in S} b(s) T(s, a, s')$$
$$b_a^o(s') = \alpha\, b_a(s') O(s', a, o),$$

where α is a normalization constant. If for all $s' \in S$, $b_a^o(s') = 0$ because no observation is consistent with the belief state b_a, then the belief state is not added to the graph.

The expected reward $q(a, b)$ of a plan that starts with action a at belief state b is the sum of current and future rewards:

$$q(a, b) = \sum_{s \in S} \sum_{s' \in S} b(s) T(s, a, s') R(s, a, s') + \sum_{o \in \Omega} b_a(s') O(s', a, o) \mathcal{V}(b_a^o),$$

where the expected reward for a belief state is $\mathcal{V}(b) = \max_{a \in A} q(a, b)$. Terminal vertices are assigned the expected goal reward

$$q(\perp, b) = \sum_{s \in S} b(s) R(s, \perp, \perp).$$

AO* Algorithm: We solve the finite horizon POMDP problem with `AO*` search [Nilsson, 1980] in the space of belief states. The `AO*` algorithm, listed in Figure 2, takes the planning problem as input, and iteratively constructs the belief space graph G rooted at b_I. The algorithm involves three iterated steps: expand the current plan with the `ExpandPlan` routine (line 3), collect the ancestors Z' of new vertices Z (line 4), and compute the current best partial plan (line 5). The algorithm ends when it expands no new vertices. In the following we briefly describe the sub-routines used by `AO*`.

The `ExpandPlan` routine recursively walks the current plan to find unexpanded vertices (lines 2-5). Upon finding a vertex to expand, it generates all successors of the vertex (lines 7-18). Generating successors involves assigning the $\perp$ action if the vertex it is at the max horizon (line 10) or constructing the vertices reached by all action and observation combinations (lines 12-17). Notice that each vertex has its value initialized with an upper bound,

$$R^{max} = \max_{s, s' \in S, a \in A} R(s, a, s') h + \max(0, R(s, \perp, \perp)),$$

```
AO*(P)
 1: expanded(b_I) = FALSE
 2: repeat
 3:    Z = ExpandPlan(b_I, 0)
 4:    Z' = AddAncestors(Z)
 5:    Update(Z')
 6: until |Z| = 0
ExpandPlan(b, hzn)
 1: if expanded(b) then
 2:    for e(b, b^o_{P(b)}) ∈ E do
 3:       Z' = ExpandPlan(b^o_{P(b)}, hzn+1)
 4:       Z = Z ∪ Z'
 5:    end for
 6: else
 7:    expanded(b) = TRUE
 8:    Z = Z ∪ {b}
 9:    if hzn == h then
10:       E = E ∪ {e(b, ⊥) = (⊥, ⊥)}
11:    else
12:       for a ∈ A, o ∈ Ω do
13:          V = V ∪ {b^o_a}
14:          E = E ∪ {e(b, b^o_a) = (a, o)}
15:          expanded(b^o_a) = FALSE
16:          V(b^o_a) = R^max
17:       end for
18:    end if
19: end if
20: return Z
```

Figure 2: AO* Search Algorithm.

on its expected reward. The upper bound plays a role in pruning vertices from consideration in the search.

After expanding the current plan, ExpandPlan returns the set of expanded vertices Z. In order for Update (Figure 3) to find the best plan, given the new vertices, AddAncestors adds to Z' every ancestor vertex of a vertex in Z. The resulting set of vertices consists of every vertex whose value (and best action) can change after Update. The Update routine iteratively removes vertices from Z that have no descendent in Z and calls Backup until no vertices remain in Z. The Backup routine computes the value of a vertex and sets its best action. The reason Update chooses vertices with no descendent in Z is to ensure each vertex has its value updated with the updated values of its children.

AO* can often avoid computing the entire belief space graph G, leading to significant savings in problems with large horizons. By initializing vertices with an upper bound on their value it is possible to ignore vertices that have a consistently lowest upper bound. For example in Backup, if there exists an action whose q-value is always greater than the alternative actions, then the best action will never be set to one of the alternatives. Further, because the alternative actions are never considered best, ExpandSolution will never expand them. As we will explore in the empirical evaluation, the reward function has a significant effect on the number of vertex expansions. In the worst case, it is possible to expand

```
AddAncestors(Z)
 1: Z' = Z
 2: while ∃b s.t. e(b, b^o_{P(b)}) ∈ E and b^o_{P(b)} ∈ Z do
 3:    Z' = Z' ∪ b
 4: end while
 5: return Z'
Update(Z)
 1: while |Z| > 0 do
 2:    Remove b ∈ Z s.t. ¬∃b' ∈ Z where e(b, b') ∈ E
 3:    Backup(b)
 4: end while
Backup(b)
 1: for all a ∈ A ∪ {⊥} do
 2:    Compute q(a, b)
 3: end for
 4: V(b) = max_{a∈A∪{⊥}} q(a, b)
 5: P(b) = argmax_{a∈A∪{⊥}} q(a, b)
```

Figure 3: AO* Subroutines.

```
Datta(P)
 1: ExpandPlanD(b_I, 0)
 2: Update(V)
ExpandPlanD(b, hzn)
 1: if hzn == h then
 2:    E = E ∪ {e(b, ⊥) = (⊥, ⊥)}
 3: else
 4:    for a ∈ A, o ∈ Ω do
 5:       V = V ∪ {b^o_a}
 6:       E = E ∪ {e(b, b^o_a) = (a, o)}
 7:       ExpandPlanD(b^o_a, hzn+1)
 8:    end for
 9: end if
```

Figure 4: Datta Enumeration Algorithm.

the entire graph G, as would Datta.

Enumeration Algorithm: In order to compare our planner to the work of [Datta *et al.*, 2004], we provide a description of their algorithm, we call Datta, in Figure 4. Unlike the iterative AO*, Datta consists of two steps: expand G with ExpandPlanD, and then update each vertex $v \in V$ with Update. The ExpandPlanD routine recursively expands G by either reaching a terminal vertex at the horizon (line 2), or generating and recursing on each child of a vertex (lines 4-8). Following ExpandPlanD, Update computes the best action for each vertex in V.

Unlike AO*, Datta is unable to prune vertices from expansion, making it insensitive to the reward function. While the result of the two algorithms is identical, the time and space required can be very different. We implemented both algorithms within our planner and demonstrate their effec-

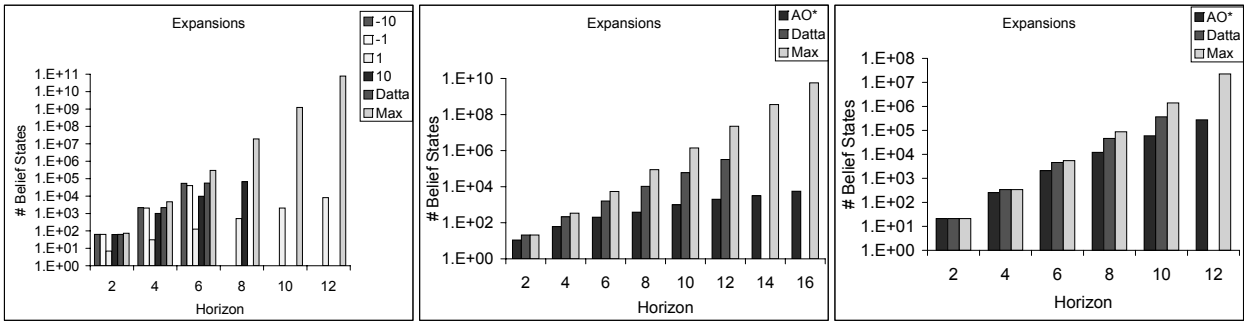

Figure 5: Number of Expanded Vertices for random GRN (left), WNT5A intervention (center), PIRIN intervention (right).

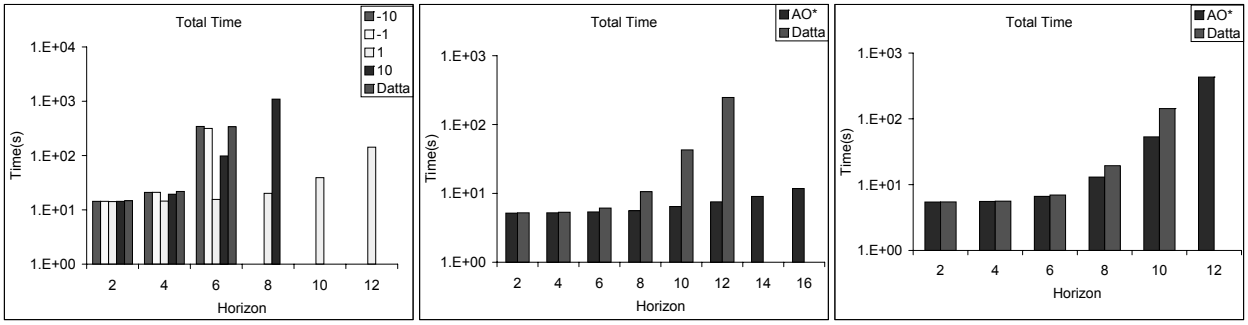

Figure 6: Total Planning Time(s) for random GRN (left), WNT5A intervention (center), PIRIN intervention (right).

tiveness on the GRN intervention planning problems. We also implemented a straight-forward version of the [Datta *et al.*, 2004] algorithm that does not make use of several efficiency improvements within the planner, such as using ADDs [Bryant, 1986] for compact action and belief state representation, as well as duplicate belief state detection.

5 Empirical Evaluation

To test the feasibility of using our planner to solve GRN intervention problems, we experiment with a random GRN and the WNT5A GRN [Weeraratna *et al.*, 2002; Datta *et al.*, 2004]. The following table summarizes the features of the GRNs:

	$\|\mathcal{G}\|$	$\|Dom\|$	$\|F\|$	$\|X\|$	$\|\mathcal{O}\|$
Random	7	2	14	3	1
WNT5A	7	2	14	1	1

Both networks use seven genes (each with two activity levels), two predictor functions per gene, each with two genes as predictors. The predictor functions were selected randomly in the random GRN and learned from micro-array data (measurements of mRNA concentrations in a cell that indicate gene activity levels) in the WNT5A GRN. In order to learn predictor functions, we use a coefficient of determination (COD) statistic to measure the strength of correlation between predictor and target genes from normalized and discretized data [Kim *et al.*, 2000]. We use the two predictor functions with highest COD for each gene, setting their weight equal to the COD.

Within both GRNs we study several intervention problems. In the random GRN, we vary the goal W to assign different rewards to terminal states, while assigning interventions a negative reward of one. This illustrates the ability of `AO*` to prune the search space in comparison with enumeration. In the WNT5A GRN, we reproduce two intervention problems studied by [Datta *et al.*, 2004]. The first directly intervenes to suppress WNT5A (which happens to be the goal) and observes the PIRIN gene. The second attempts to indirectly control WNT5A by PIRIN (a predictor gene of WNT5A) intervention. Both WNT5A problems use the same reward function, assigning interventions a negative reward of one and the goal (activating WNT5A) a negative reward of three. We use negative reward for the goal to maintain consistency with the [Datta *et al.*, 2004] model. Thus plans avoid activating WNT5A, which is equivalent to pursuing suppression. Both WNT5A networks use the initial belief state where each gene is set to an activity level with probability proportional to its observed frequency in the data.

Our planner is implemented in C++ and ran on a 2.8GHz P4 Linux machine with 1GB of RAM. Each experiment was given a twenty minute time limit. For our planner binary and gene regulatory network encodings, several of which we did not have space to describe, please visit: http://verde.eas.asu.edu/GRN.

Random GRN: The leftmost plot in Figure 5 depicts the number of expanded vertices (including terminal actions) in `AO*`, `Datta`, and the maximum possible (Max). The results for `AO*` are indexed by a number indicating the reward associated with the goal, since `Datta` is insensitive to reward. Max represents the number of vertices expanded in a search tree (versus a graph), similar to the original implementation of [Datta *et al.*, 2004]. Because we implemented the `Datta`

algorithm in our planner, like AO* it finds duplicate belief states. Without duplicate detection, Datta would expand as many vertices as Max. The leftmost plot in Figure 6 shows the associated total planning time (which is proportional to the number of expanded vertices). Missing bars indicate the instance was cut off by reaching the 20 minute time limit.

We notice several important points in these results. AO* is sensitive to the goal reward function, expanding much fewer vertices than Datta in some cases. Despite AO* using dynamic programming over its partial solution many times, it never takes more time than Datta. As a result, when it is able to prune vertices, AO* scales much better.

WNT5A GRN: The center plots in Figures 5 and 6 show results in the WNT5A GRN intervening WNT5A and observing PIRIN. Here AO* greatly outperforms Datta. The difference is partly due to the fact that AO* quickly recognizes that the optimal plan intervenes once at the end of each plan branch where WNT5A is not already deactivated. We also directly implemented the approach of [Datta *et al.*, 2004], which does no duplicate detection, and it was able to solve this problem to a horizon of ten, using significantly more memory and time. While our implementation of Datta is limited by time, the direct implementation is limited by space, exceeding 1 GB memory past horizon ten. The rightmost plots in Figures 5 and 6 show results in the WNT5A GRN intervening PIRIN and observing WNT5A. Finding plans in this problem requires more search, but AO* can still prune.

Discussion: We have shown that AO* is a viable approach to solving GRN intervention problems. Where the Datta algorithm quickly exceeds our time limit as the horizon increases, AO* is sensitive to reward functions and can scale to larger horizons. Using both artificial and existing GRNs of practical interest, we have seen that AO* performs well by pruning vertices based on upper bounds. Perhaps the next step is to achieve tighter upper bounds through heuristics. While we did not discuss alternative GRN formulations (e.g., with more genes, activity levels, and observations), we have studied such problems and see similar scalability.

6 Related Work

Planning in cellular processes is a relatively new area of research, with some initial work on problem formulation as well as solution algorithms. There also exists some preliminary works on representing cellular processes without specific emphasis on planning interventions. The closest body of work for planning interventions [Datta *et al.*, 2004] models the problem as a finite horizon control problem.

Some recent works in the AI community have focussed on simply representing cellular processes. One approach, discovers signal transduction pathways with a deterministic classical planner [Khan *et al.*, 2003]. Further along this vein, [Tran and Baral, 2005] model change in cellular processes as exogenous actions, termed triggers.

7 Conclusion & Future Work

We have presented a formulation of GRN intervention as decision theoretic planning. Our planner relies on several advances in AI planning to perform efficient reasoning. As a result, we have improved the scalability of planning interventions in GRNs over previous work.

In the future, we would like to decouple the representation of cellular process dynamics from our intervention actions. We think the right approach is to use exogenous actions or processes. While recent work in deterministic planning [McDermott, 2003] has discussed models for controlling exogenous processes and triggers [Tran and Baral, 2005], less thought has been given to such processes in a probabilistic setting (with the exception of [Blythe, 1994]). Alternatively, research in simulating cellular processes [Ramsey *et al.*, 2005] has considered probabilistic exogenous processes, but not under the control of a planner.

Acknowledgements: This work was supported in part by NSF grant IIS-0308139, by ONR grant N000140610058, the ARCS foundation, and an IBM faculty award. We thank Ashish Choudhary, Edward Dougherty, William Cushing, and Subbarao Kambhampati for helpful discussions.

References

[Blythe, 1994] J. Blythe. Planning with external events. In *Proceedings of UAI'04*, 1994.

[Bryant, 1986] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[Datta *et al.*, 2004] A. Datta, A. Choudhary, M. Bittner, and E. Dougherty. External control in markovian genetic regulatory networks: the imperfect information case. *Bioinformatics*, 20(6):924–930, 2004.

[Khan *et al.*, 2003] S. Khan, K. Decker, W. Gillis, and C. Schmidt. A multi-agent system-driven ai planning approach to biological pathway discovery. In *Proceedings of ICAPS'03*, 2003.

[Kim *et al.*, 2000] S. Kim, E. Dougherty, M. Bittner, Y. Chen, K. Sivakumar, P. Meltzer, and J. Trent. General nonlinear framework for the analysis of gene interaction via multivariate expression arrays. *Journal of Biomedical Optics*, 5(4):411–424, 2000.

[Kim *et al.*, 2002] S. Kim, H. Li, E. Dougherty, N. Cao, Y. Chen, M. Bittner, and E. Suh. Can markov chain models mimic biological regulation? *J. of Biological Systems*, 10(4):337–357, 2002.

[McDermott, 2003] D.V. McDermott. The formal semantics of processes in pddl. ICAPS Workshop on PDDL, 2003.

[Nilsson, 1980] N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, 1980.

[Ramsey *et al.*, 2005] S. Ramsey, D. Orrell, and H. Bolouri. Dizzy: Stochastic simulation of large-scale genetic regulatory networks. *J. Bioinformatics Comput. Biol*, 3(2):415–436, 2005.

[Rao *et al.*, 2002] C.V. Rao, D.M. Wolf, and A.P. Arkin. Control, exploitation and tolerance of intracellular noise. *Nature*, 420:231–237, November 2002.

[Tran and Baral, 2005] N. Tran and C. Baral. Issues in reasoning about interaction networks in cells: necessity of event ordering knowledge. In *Proceedings of AAAI'05*, 2005.

[Weeraratna *et al.*, 2002] A. T. Weeraratna, Y. Jiang, G. Hostetter, K. Rosenblatt, P. Duray, M. Bittner, and J. M. Trent. Wnt5a signaling directly affects cell motility and invasion of metastatic melanoma. *Cancer Cell*, 1(3):279–88, 2002.