A POMDP Model for Safe Geological Carbon Sequestration

Anthony Corso
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA
acorso@stanford.edu

Yizheng Wang
Department of Earth and Planetary Sciences
Stanford University
Stanford, CA
yizhengw@stanford.edu

Markus Zechner
Department of Earth and Planetary Sciences
Stanford University
Stanford, CA
mzechner@stanford.edu

Jef Caers
Department of Earth and Planetary Sciences
Stanford University
Stanford, CA
jcaers@stanford.edu

Mykel Kochenderfer
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA
mykel@stanford.edu

Abstract

Geological carbon capture and sequestration (CCS), where CO_2 is stored in subsurface formations, is a promising and scalable approach for reducing global emissions. However, if done incorrectly, it may lead to earthquakes and leakage of CO_2 back to the surface, harming both humans and the environment. These risks are exacerbated by the large amount of uncertainty in the structure of the storage formation. For these reasons, we propose that CCS operations be modeled as a partially observable Markov decision process (POMDP) and decisions be informed using automated planning algorithms. To this end, we develop a simplified model of CCS operations based on a 2D spillpoint analysis that retains many of the challenges and safety considerations of the real-world problem. We show how off-the-shelf POMDP solvers outperform expert baselines for safe CCS planning. This POMDP model can be used as a test bed to drive the development of novel decision-making algorithms for CCS operations.

1 Introduction

While global warming continues to pose an existential risk to humanity, the latest IPCC report [1] suggests that reducing emissions by half before 2030 will secure a livable future. A wide variety of measures will be necessary to reduce greenhouse gas emissions, but in all significant reports, carbon capture and sequestration (CCS), where carbon is permanently removed from the atmosphere, accounts for about one-quarter to two-thirds of the cumulative emission reduction [2], [3]. Geological carbon storage, where CO_2 is stored into subsurface formations such as saline aquifers, is a promising technique due to its scalability and the recent increases in the price of CO_2 [4]–[6], and will be the focus of this work.

(a) Spillpoint model for CO_2 storage.

(b) Performance metrics for a no-uncertainty baseline against POMCPOW (ours).
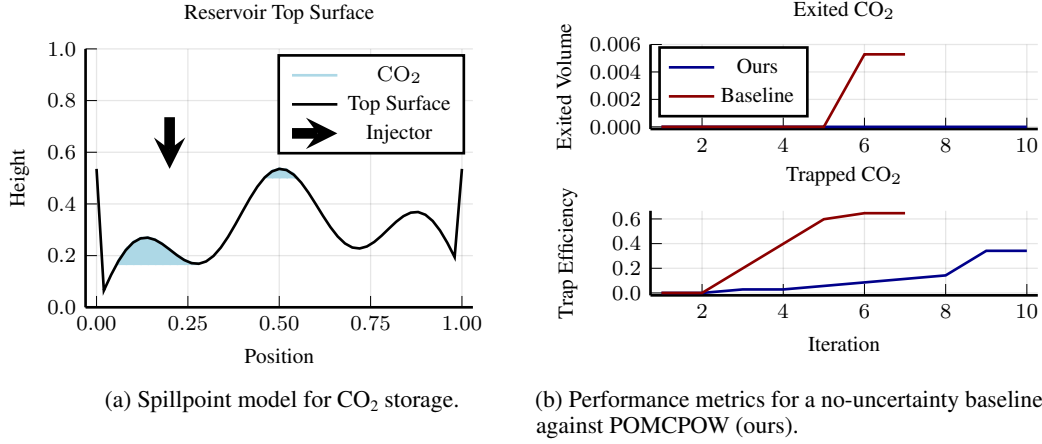
Figure 1: A simplified model for CCS operations ammenable to automated decision making.

Even though CCS projects are similar to other, well-studied, subsurface problems such as groundwater or oil and gas applications, they involve additional and unsolved challenges. First, much less is known about saline aquifers compared to oil and gas reservoirs, resulting in significantly higher subsurface uncertainty. Second, the interaction of supercritical CO_2, brine, and rock results in complex physical and chemical processes such as trapping mechanisms, mineral precipitations, and geomechanics that must be accounted for. Lastly, injection of CO_2 into geological formations comes with significant risks such as induced earthquakes, fractured cap rocks, and reactivation of faults that can lead to leakage, harming human life and the environment. Therefore, any decision made in the course of a CCS project needs to balance the trade-off between safety and the utilization of storage capacity to keep the project economically viable.

CCS operations require a large number of sequential decisions to be made under uncertainty, such as selecting aquifers from a portfolio, choosing injection rates, defining well locations, selecting information gathering campaigns, and selecting monitoring strategies. The inherent subsurface uncertainty combined with complex physio-chemical processes results in a challenging decision problem for CCS operations that require a tight coupling of information gathering and acting. For these reasons, we propose to formulate CCS as a partially observable Markov decision process (POMDP) and rely on POMDP solvers [7] to inform the decision-making process.

Related work: Quantifying subsurface uncertainty is especially important for successfully implementing safe and effective CCS operations and monitoring. Although considerable research has been devoted to developing better uncertainty quantification techniques [8], [9], significantly less attention has been paid to determining how to act on that information. Prior work has focused on non-sequential optimization approaches [10], [11], or the use of sequential approaches in other subsurface domains such as oil/gas [12] and groundwater [13].

In this work we introduce a POMDP model of CCS operations based on simplified physics in order to spur the development of more advanced solution techniques that can produce safe and reliable plans. We first describe our simplified model, which is based on spillpoint analysis, but retains many of the challenges associated with the full-scale CCS problem (fig. 1a). We then compare a variety of rule-based plans against state-of-the-art POMDP solvers and show how current off the shelf approaches can improve upon expert plans (fig. 1b). The spillpoint CCS model is implemented in the open source POMDPs.jl[1] framework.

2 Approach

POMDP Background A partially observable Markov decision processes (POMDP) [7] is a model for sequential decision making where an agent takes actions over time to maximize the accumulation of a reward. A POMDP is defined by the tuple $(S, A, T, R, O, Z, \gamma)$ where S is the set of states, A

---

[1] `https://juliapomdp.github.io/POMDPs.jl/latest/`

is the set of actions, T is a transition model, R is the reward function, O is the observation space, Z is the observation model, and γ is the discount factor. For complex dynamical systems such as CCS, the transition function is implemented as a simulator from which transitions can be drawn according to the distribution $s' \sim T(\cdot \mid s, a)$ defined by the action $a \in A$ taken in state $s \in S$. The agent is uncertain about the current state and therefore maintains a distribution over states called a belief $b(s)$. The initial belief $b_0(s)$ is updated at each timestep using observations $o \in O$ sampled with likelihood $Z(o \mid s, a, s')$. A policy π is a function that maps beliefs to actions such that $a = \pi(b)$. POMDP solvers seek to approximate the optimal policy π^* that maximizes the expected discounted sum of future rewards.

Challenges of Solving a CCS POMDP CCS can be formulated as a POMDP where the states represent the reservoir structure and the properties of the multi-phase fluid stored therein. The actions are the placement of sensors, the drilling of CO_2 injectors, and the setting of the rate of injection. The observations can come from many modalities including direct observation of the geology through drilling, as well as indirect methods such as seismic monitoring. The transition function is the defined by the complex fluid dynamics that govern the injection and evolution of the CO_2 plume.

Optimal planning for CCS operations is challenging for several reasons: the state, action and observation spaces are continuous and high dimensional, the transition function is computationally intensive, requiring hours for a single transition, and since CCS operations are safety-critical, the plan must have a high degree of reliability. To encourage the development of algorithms that can solve planning problems of this scale, we developed a simplified model of CCS operations that retains many of these challenges, while requiring far less computational effort.

A Spillpoint Model POMDP for CCS Buoyancy is one of the dominant physical forces that govern the dynamics of CO_2 injection [14]. Spillpoint models, where only the effects of buoyancy are used to compute the dynamics of injected CO_2, are therefore a simple, but realistic model and have previously been used to estimate the trapping capacity of a reservoir given the top surface geometry [14]. In this work, we design a CCS POMDP using a 2D spillpoint model as the transition function to approximate the CO_2 dynamics with low computational effort.

In our model, the reservoir is defined by a top surface as shown in fig. 1a, and a porosity ρ which defines the fraction of the volume available to store CO_2. Given an injection location and volume, the model determines the amount of CO_2 trapped in each region of the reservoir. To compute CO_2 saturation, the reservoir is automatically segmented into *spill regions*, where CO_2 is trapped due to buoyant forces, and *spillpoints*, which are connections between spill regions, across which CO_2 will migrate. For example in fig. 1a, CO_2 is injected in the left spill region until the trapped volume exceeds the spill region volume, and it begins to fill the central spill region. If the spillpoint connects to a fault, then any CO_2 that migrates through that spillpoint is considered to have exited the reservoir or *leaked*. We model a fault on the left and right boundary of the reservoir domain. For additional details of the spillpoint algorithm, see appendix A.

To construct a diverse set of reservoirs, we model the top surface geometry by the function

$$h = h_{\text{lobe}} \sin(5\pi x) + h_{\text{center}} \sin(\pi x) + h_{\text{elev}} x \tag{1}$$

where x is the lateral position. The outer lobe height h_{lobe}, central lobe height h_{center}, and elevation slope h_{elev} determine the size and relationship between spill regions. The initial belief is implicitly defined by distributions over top surface parameters h_{lobe}, h_{center}, h_{elev}, and the porosity ρ.

The rest of the POMDP is defined as follows. The state is defined by the top surface height of the reservoir, the porosity, the injector location, and the amount of trapped and exited CO_2. The actions include the position at which to drill the injector, the rate of CO_2 injection, the ability to make noisy observations of the trapped CO_2, and the option to end CCS operations. The action to drill involves specifying a drill location and returns the top surface height at that location. The action to observe involves specifying sensor locations which return a measurement of the top surface height and CO_2 thickness at that location, with noise distributed normally. If no CO_2 is present, then 0 is returned for both values, providing no additional information on the top surface geometry. Additionally, we model a sensor on each of the boundaries of the domain that can noiselessly detect if CO_2 is leaking, and is included in the observation at each transition, regardless of the action taken.

The reward function includes the competing objectives of storing CO_2 while avoiding leakage. It has four components including trapped CO_2 volume, exited CO_2 volume, observation costs, and an

Table 1: Comparison of CCS policies

Method	Return	Observations	Leak Fraction	Trap Efficiency
Random	-104.38 ± 320.41	4.60 ± 6.43	0.10	0.08 ± 0.07
Best Guess MDP	-516.03 ± 549.10	0.00 ± 0.00	0.50	0.70 ± 0.27
Fixed Schedule	-97.91 ± 316.28	42.70 ± 24.57	0.10	$\mathbf{0.82 \pm 0.17}$
POMCPOW (Basic)	-2811.32 ± 6009.65	1.20 ± 1.55	0.20	0.58 ± 0.30
POMCPOW (SIR)	$\mathbf{2.24 \pm 4.04}$	3.00 ± 3.46	$\mathbf{0.00}$	0.27 ± 0.32

indicator of CO_2 leakage. Specifically, the reward is

$$R(s, a, s') = \lambda_{\text{leak}} \mathbb{1}_{\{\Delta V_{\text{exited}} > 0\}} + \lambda_{\text{exited}} \Delta V_{\text{exited}} + \lambda_{\text{trapped}} \Delta V_{\text{trapped}} + \lambda_{\text{observation}} N_{\text{obs}} \quad (2)$$

where $\mathbb{1}_{\{.\}}$ is the indicator function and V_{exited} and V_{exited} are, respectively, the changes in exited and trapped CO_2 between states s and s', and N_{obs} is the number of observation locations.

3 Experiments

We solve the spillpoint CCS POMDP with four different approaches. The first approach (**Random**) is a random policy where the actions are selected randomly until CCS operations are stopped by chance or until leakage is detected, at which point operations are halted. The second approach (**Best Guess MDP**) we do not consider uncertainty in the state, and instead start by selecting a top surface geometry that matches a single top surface measurement and treating it as the ground truth. Then Monte Carlo tree search is used to plan an optimal injection strategy. The third approach (**Fixed Schedule**) is an expert policy with a fixed observation schedule. The action to observer is used every three timesteps, incorporating each observation into a belief update. Injection continues as long as there is no possibility of leakage according to the current belief. The last two approaches include solving the formulation with an off-the-shelf POMDP solver called POMCPOW [15]. The first version of POMCPOW (**POMCPOW (Basic)**) uses a basic particle filter as the belief update strategy while the second version (**POMCPOW (SIR)**) uses the more advanced sequential importance resampling (SIR) belief updating approach (see appendix B for details).

To evaluate the various approaches, we randomly sampled 10 realizations of the reservoir geometry (as the ground truth), and ran each algorithm on this test set, recording the following 4 metrics. The first metric is the sum of discounted rewards (**Return**) defined in equation 2, which includes terms associated with total trapped CO_2 as well as safety considerations. Additionally, we consider metrics associated with each sub component of the reward. These include the average number of observations per trajectory (**Observations**), the fraction of realizations that had leakage (**Leak Fraction**), and the fraction of trapped CO_2 compared to the maximum possible trapped volume (**Trap Efficiency**). All hyperparameters are given in appendix C.

The results of these experiments are shown in table 1. The highest return and smallest amount of leakage is achieved by POMCPOW with SIR. This algorithm behaved conservatively, however, and rarely used the *observe* action to gather more information, and therefore only reached an average of 27% trap efficiency. Ignoring uncertainty lead to high trap efficiency but caused a significant chance of leakage at 50%. The fixed observation schedule had an even higher trap efficiency (82%) but still leaked in 10% of cases. Using a naive belief updating procedure (particle filter) caused POMCPOW to become overconfident in its belief and therefore lead two cases with significant leakage. We therefore conclude that automated decision making systems can be helpful for safe CCS operations.

4 Conclusion

To address the upcoming challenges of safe geological carbon storage, we proposed to model CCS as a POMDP and use automated planning algorithms to design safe operational plans. We developed a simplified CCS formulation based on spillpoint analysis that eases the computational burden of exploring automated decision making for CCS and demonstrated the performance of several existing algorithms and baselines. We hope this model can be used to drive the development of new algorithms that support real-world CCS operations.

Acknowledgments and Disclosure of Funding

References

[1] H.-O. Pörtner, D. C. Roberts, H. Adams, C. Adler, P. Aldunce, E. Ali, R. A. Begum, R. Betts, R. B. Kerr, R. Biesbroek, *et al.*, "Climate change 2022: Impacts, adaptation and vulnerability," *IPCC Sixth Assessment Report*, 2022.

[2] International Energy Agency, *World Energy Outlook 2020*. 2020, p. 464.

[3] S. M. Benson and F. M. Orr, "Carbon dioxide capture and storage," *MRS Bulletin*, vol. 33, no. 4, pp. 303–305, 2008.

[4] B. Metz, O. Davidson, H. De Coninck, M. Loos, and L. Meyer, *IPCC special report on carbon dioxide capture and storage*. Cambridge: Cambridge University Press, 2005.

[5] T. Wilberforce, A. Baroutaji, B. Soudan, A. H. Al-Alami, and A. G. Olabi, "Outlook of carbon capture technology and challenges," *Science of the total environment*, vol. 657, pp. 56–72, 2019.

[6] I. E. Agency, "CCUS in clean energy transitions," IEA Paris, 2020.

[7] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. MIT Press, 2022.

[8] C. Scheidt, L. Li, and J. Caers, *Quantifying uncertainty in subsurface systems*. John Wiley & Sons, 2018, vol. 236.

[9] D. Lu, S. Painter, N. Azzolina, and M. Burton-Kelly, "Accurate and timely forecasts of geologic carbon storage using machine learning methods," in *NeurIPS Workshop on Tackling Climate Change with Machine Learning*, 2021.

[10] O. J. Isebor, L. J. Durlofsky, and D. E. Ciaurri, "A derivative-free methodology with local and global search for the constrained joint optimization of well locations and controls," *Computational Geosciences*, vol. 18, no. 3, pp. 463–482, 2014.

[11] Y. Yang, J. Wu, X. Sun, J. Wu, and C. Zheng, "A niched pareto tabu search for multi-objective optimal design of groundwater remediation systems," *Journal of Hydrology*, vol. 490, pp. 56–73, 2013.

[12] G. De Paola, C. Ibanez-Llano, J. Rios, and G. Kollias, "Reinforcement learning for field development policy optimization," in *SPE Annual Technical Conference and Exhibition*, OnePetro, 2020.

[13] Y. Wang, M. Zechner, J. M. Mern, M. J. Kochenderfer, and J. K. Caers, "A sequential decision-making framework with uncertainty quantification for groundwater management," *Advances in Water Resources*, vol. 166, p. 104 266, 2022.

[14] H. Møll Nilsen, K.-A. Lie, O. Møyner, and O. Andersen, "Spill-point analysis and structural trapping capacity in saline aquifers using MRST-co2lab," *Computers and Geosciences*, vol. 75, pp. 33–43, 2015.

[15] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *International Conference on Automated Planning and Scheduling*, 2018.

A   Spillpoint Algorithm

Spillpoint analysis [14] allows for the calculation of CO_2 saturation from the geometry of the reservoir and total amount of CO_2 injected. The procedure takes as input the top surface geometry $h(x)$, the spill region s_r, and the total injected CO_2 V_{inject} and returns the CO_2 saturation at each position, the trapped volume and the exited volume. Details of the algorithm can be found in algorithm 1. The CO_2 saturation is represented a set of polyhedra that indicate the 2D area that is occupied by CO_2. The polyhedra are computed by first identifying the spill region responsible for trapping (we followed the procedure outlined by Møll Nilsen, Lie, Møyner, *et al.*), and then determining the depth of the CO_2 plume. The depth is found by using an optimizer to minimize the difference between the injected

volume and the volume of of the polyhedra for a given depth. This optimization procedure is the most expensive part of the computation.

Algorithm 1 Algorithm for determining CO_2 depth using spillpoint analysis for mesh m and volume V

1: **function** INJECT($h, s_r, V_{\text{inject}}$)
2:      $V_{\text{trap}} = 0$
3:      $SRs = \emptyset$ $\triangleright$ Spill regions that are full
4:      $Sat = \emptyset$ $\triangleright$ Polyhedra representing saturated volume
5:      **while** $V_{\text{trap}} < V$ $\triangleright$ While there is remaining CO_2 volume
6:          $V_{\text{remain}} = V_{\text{inject}} - V_{\text{trap}}$ $\triangleright$ Compute remaining CO_2
7:          $V_{\text{sr}} \leftarrow$ SPILLREGIONVOLUME(s_r) $\triangleright$ Max storage of spill region
8:          **if** $V_{\text{sr}} > V_{\text{remain}}$
9:              $Sat \leftarrow Sat \bigcup$ GETPOLYS($h, s_r, V_{\text{remain}}$)
10:             **break**
11:          **else**
12:              $Sat \leftarrow Sat \bigcup$ GETPOLYS(h, s_r, V_{sr})
13:              $V_{\text{trap}} \leftarrow V_{\text{trap}} + V_{\text{sr}}$ $\triangleright$ Count the new trapped CO_2
14:              $SRs \leftarrow SRs \bigcup s_r$ $\triangleright$ Record full spill region
15:              $sr \leftarrow$ UPHILLSPILLREGION(h, s_r) $\triangleright$ Get the connecting spill region
16:              **if** $s_r \in$ FAULTS(h) $\triangleright$ If next spill region is a fault
17:                  $V_{\text{exit}} \leftarrow V_{\text{remain}}$
18:                  **break**
19:              **if** $s_r \in SRs$ $\triangleright$ Adjacent spill regions are full
20:                  $h \leftarrow$ MERGESPILLREGIONS(h, s_r) $\triangleright$ Merge into larger region
21:                  $V_{\text{trap}} \leftarrow 0$
22:                  $SRs = \emptyset$
23:                  $Sat = \emptyset$
24:      **return** $Sat, V_{\text{trap}}, V_{\text{exited}}$

B   Belief Updating

After taking action a, the prior belief $b(s)$ is updated to the posterior belief $b(s')$ with the observation o according to Bayes' rule [7]

$$b(s') = O(o \mid a, s') \int_s T(s' \mid s, a)b(s)ds \tag{3}$$

When the transition function, observation model, or belief representation is too complex to do an analytical belief update, we rely on approximate methods such as particle filters.

The basic particle filter algorithm is shown in algorithm 2. The belief is represented by a number of *particles* which are samples of the state. Each update of the belief includes simulating the transition function for each particle to get a sample of the next state, computing the observation likelihood and then resampling according to the likelihood weights. The challenge with basic particle filtering is *particle depletion* where the diversity of particles is reduced due to many of the particle having low likelihood weights. The problem is exacerbated with the number of belief updates and can require an intractably large number of initial state samples.

An improvement to basic particle filtering, known as sequential importance resampling (SIR), iteratively adapts a proposal distribution to be closer to the posterior. The technique is outlined in algorithm 3. States are sampled from a proposal distribution that is initialized to the current belief. During each iteration samples are drawn from the proposal and weighted by the observation model and the importance weight. The proposal distribution is refit to these weighted samples and used in the next iteration, with the goal of being closer to the posterior distribution and producing better samples. Finally, a resampling step produces the final belief.

Our implementation of SIR makes several design decisions that were crucial to good performance. First, the proposal distribution is a kernel density estimate over the parameters that define the top

Algorithm 2 Basic particle filter for belief b (with N particles), after action a and observation o

1: **function** BASICPARTICLEFILTERUPDATE(b, N, a, o)
2:     Sample transitions $s_i' \sim T(\cdot \mid s_i, a)$ where $b = \{s_i\}_{i=1}^N$
3:     Compute likelihood weights $w_i \leftarrow O(o \mid a, s_i')$
4:     $b' \leftarrow$ Resample N particles from $\{s_i'\}_{i=1}^N$ with weights $\{w_i\}_{i=1}^N$
5:     **return** b'

surface geometry. Second, we incorporate all prior observations (not just the most recent) into the particle weight to ensure a good match with the observed data. Third, if there is insufficient particle diversity when fitting the next proposal, we fit to the *elite samples* defined by the top 30th percentile of samples in terms of their weights. Fourth, at each iteration we include 50% samples from the prior distribution to mitigate the effect of poor proposals. Lastly, we run SIR until there are at least $N/2$ unique particles when resampling or we reach the pre-specified computational budget for the belief update.

Algorithm 3 Sequential Importance Resampling (SIR) particle filter for belief b (with N particles), after action a and observation o, taking N_k samples per iteration.

1: **function** SIRPARTICLEFILTERUPDATE(b, N, a, o)
2:     $q_1(s) \leftarrow b(s)$
3:     $\mathcal{S} \leftarrow \emptyset$
4:     $\mathcal{W} \leftarrow \emptyset$
5:     **for** $k \in 1 \dots K$
6:         Sample N_k states from proposal $s_i \sim q_k$
7:         Compute importance weight $w_i \leftarrow b(s)/q_k(s)$
8:         Sample transitions $s_i' \sim T(\cdot \mid s_i, a)$
9:         Include likelihood weight $w_i \leftarrow w_i \cdot O(o \mid a, s_i')$
10:         Store particles $\mathcal{S} \leftarrow \mathcal{S} \bigcup s_i$
11:         Store weights $\mathcal{W} \leftarrow \mathcal{W} \bigcup w_i$
12:         $q_k \leftarrow$ fit $\{s_i'\}_{i=1}^{N_k}$ with weights $\{w_i\}_{i=1}^{N_k}$
13:     $b' \leftarrow$ Resample N particles from $\mathcal{S}$ with weights $\mathcal{W}$
14:     **return** b'

Comparison between the basic particle filtering approach and SIR is shown in fig. 2. For the same number of particles representing the belief, SIR leads to a better estimate of the posterior distribution while the basic particle filter suffers from particle depletion.

C   Experiment Hyperparameters

The parameters for the POMDP formulation and the hyperparameters for the experimental setup are included in table 2.
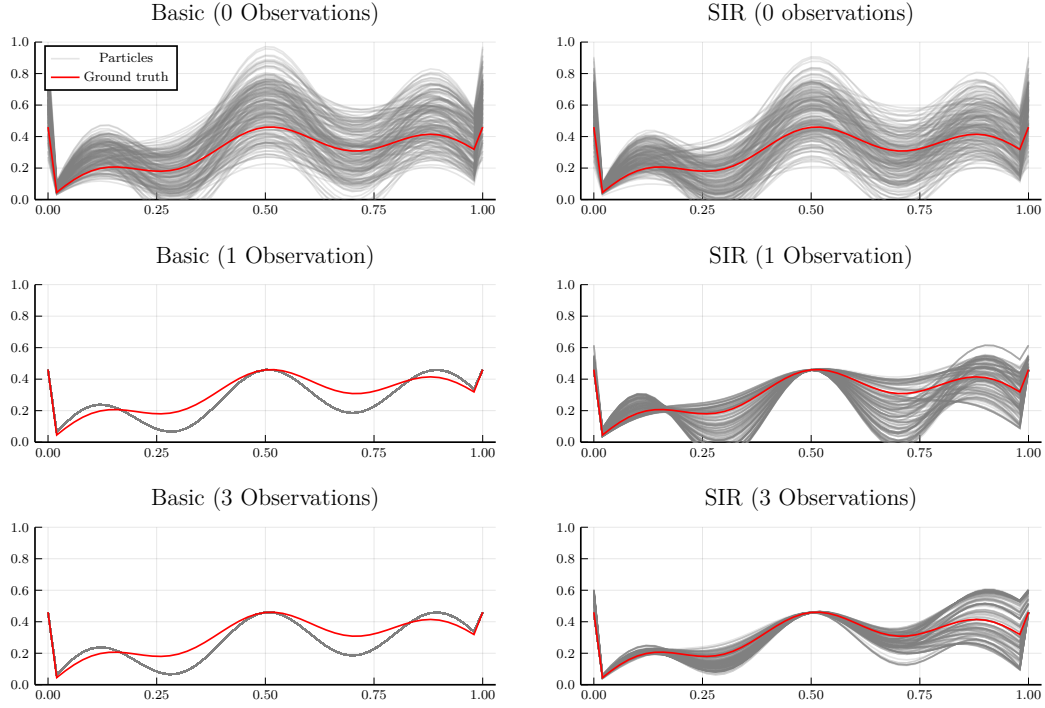
Figure 2: Comparison between basic particle filtering (left column) and SIR (right column) after different numbers of observations.

Table 2: POMDP and Solver Hyperparameters

Variable Description	Value
Discount Factor (γ)	0.9
λ_{leaked}	-1000.0
λ_{exited}	-1000.0
λ_{trapped}	100.0
λ_{obs}	-0.1
CO_2 height std. dev. (σ_{height})	0.01
CO_2 thickness std. dev. (σ_{sat})	0.01
Observation configurations	$\{0.25\!:\!0.25\!:\!0.75, 0.125\!:\!0.125\!:\!0.875\}$
Injection rates	$\{0.01, 0.07\}$
Drill Locations	$\{0.1\!:\!0.1\!:\!0.9\}$
Distribution of h_{lobe}	$\mathcal{U}(0.05, 0.25)$
Distribution of h_{center}	$\mathcal{U}(0.05, 0.5)$
Distribution of h_{elev}	$\mathcal{U}(0.05, 0.5)$
Distribution of normalized porosity (ρ)	$\mathcal{U}(0.5, 1.5)$
Exploration Coefficient for UCB	20.0
Observation widening exponent (α_{obs})	0.3
Observation widening coefficient (k_{obs})	10.0
Action widening exponent (α_{act})	0.5
Action widening coefficient (k_{act})	10.0
Number of tree queries (N_{query})	5000
Estimation value for leaf nodes	$0.1\times$ optimal return
Particles for Basic Particle filter	2000
Particles for SIR (N)	200
Samples per iteration for SIR	100
Max CPU time for belief update	60 seconds