IMPaCT: A Parallelized Software Tool for IMDP Construction and Controller Synthesis with Convergence Guarantees

Ben Wooding and Abolfazl Lavaei
School of Computing, Newcastle University

Newcastle University

Discrete-Time Stochastic Control Systems

$\mathbf{\Sigma} = (X, U, W, \varsigma, f)$:

- $X \subseteq \mathbb{R}^n$: State set
- $U \subseteq \mathbb{R}^m$: Input set
- $W \subseteq \mathbb{R}^p$: Disturbance set
- ς: A sequence of i.i.d. random variables from a sample space $\mathbf{\Omega}$ to a measurable set V_ς

$$\varsigma := \{\varsigma(k) : \Omega \to V_\varsigma,\ k \in \mathbb{N}\};$$

- $f : X \times U \times W \times V_\varsigma \to X$: Transition map

Evolution of the state of $\mathbf{\Sigma}$:

$$\mathbf{\Sigma} : x(k+1) = f(x(k), \nu(k), w(k), \varsigma(k)), \qquad k \in \mathbb{N},$$

- Default support for additive normal distributions
- Capacity for any arbitrary distribution via a custom user-defined distribution

Parallel Construction of IMDPs

Constructing IMDP Matrices $\hat{T}_{\min}, \hat{T}_{\max}, \hat{R}_{\min}, \hat{R}_{\max}, \hat{A}_{\min}, \hat{A}_{\max}$

1. Construct finite abstraction via gridding procedure
2. Label target states $\hat{r} \in \mathcal{R}$, avoid states $\hat{a} \in \mathcal{A}$, and remaining states $\hat{x}$
3. $\hat{T}_{\min}$ is **min** CDF for state-to-state transitions $\hat{x} \to \hat{x}$
4. $\hat{R}_{\min}$ is **min** CDF for state-to-target-states transitions $\hat{x} \to \mathcal{R}$
5. $\hat{A}_{\min}$ is **min** CDF for state-to-avoid-states transitions $\hat{x} \to \mathcal{A}$
6. Compute $\hat{T}_{\max}$, $\hat{R}_{\max}$, and $\hat{A}_{\max}$ similarly but with **max** CDF

Complexity Analysis

$$\mathcal{O}\left(\frac{2\kappa d}{\text{THREADS}}\right), \quad \text{for} \quad d = (n_{x_i}^n \times n_{u_j}^m \times n_{w_k}^p) \times n_{x_i}^n,$$

where κ is complexity of nonlinear optimization algorithm chosen from NLopt list.

Low-Cost Abstraction

Since

$$\hat{T}_{\max}(i,j) = 0 \implies \hat{T}_{\min}(i,j) = 0,$$

by computing $\hat{T}_{\max}$ before $\hat{T}_{\min}$, we can avoid costly optimizations for $\hat{T}_{\min}$. We do the same low-cost abstraction for $\hat{R}_{\min}$ and $\hat{A}_{\min}$.

Parallel Controller Synthesis with Convergence Guarantees

Interval Iteration for Infinite Horizon Convergence

The interval iteration algorithm solves two Bellman equations:

$$\begin{cases} V_0' = \delta_1 \hat{R} + \delta_2 \hat{A} + \hat{T} V_0, \\ V_1' = \delta_1 \hat{R} + \delta_2 \hat{A} + \hat{T} V_1, \end{cases}$$

to find the new probabilities of satisfying the specification.

- δ_1 and δ_2 are specification dependent
- V_0 and V_1 are updated to V_0' and V_1' each iteration
- it terminates when the two vectors converge, $\|V_1 - V_0\|_\infty \leq \varepsilon$

A dynamic program is solved to acquire the optimal feasible solutions $\hat{T}$, $\hat{R}$, and $\hat{A}$, which minimize over the disturbance $\hat{w}$ and maximize over the input $\hat{u}$:

$$\max_{\hat{u} \in \hat{U}} \min_{\hat{w} \in \hat{W}} \underset{\hat{R}, \hat{A}, \hat{T}}{\text{optimize}}\ \delta_1 \hat{R}(\hat{x}, \hat{u}, \hat{w}) + \delta_2 \hat{A}(\hat{x}, \hat{u}, \hat{w}) + \sum_{\forall \hat{x}' \in \hat{X}} \delta_3(\hat{x}') \hat{T}(\hat{x}' | \hat{x}, \hat{u}, \hat{w})$$

subject to the following constraints:

$$\hat{T}_{\min}(\hat{x}'|\hat{x}, \hat{u}, \hat{w}) \leq \hat{T}(\hat{x}'|\hat{x}, \hat{u}, \hat{w}) \leq \hat{T}_{\max}(\hat{x}'|\hat{x}, \hat{u}, \hat{w}),$$
$$\hat{R}_{\min}(\hat{x}, \hat{u}, \hat{w}) \leq \hat{R}(\hat{x}, \hat{u}, \hat{w}) \leq \hat{R}_{\max}(\hat{x}, \hat{u}, \hat{w}),$$
$$\hat{A}_{\min}(\hat{x}, \hat{u}, \hat{w}) \leq \hat{A}(\hat{x}, \hat{u}, \hat{w}) \leq \hat{A}_{\max}(\hat{x}, \hat{u}, \hat{w}),$$
$$\hat{R}(\hat{x}, \hat{u}, \hat{w}) + \hat{A}(\hat{x}, \hat{u}, \hat{w}) + \sum_{\forall \hat{x}' \in \hat{X}} \hat{T}(\hat{x}'|\hat{x}, \hat{u}, \hat{w}) = 1.$$

For finite horizon specifications, the traditional value iteration approach is sufficient.

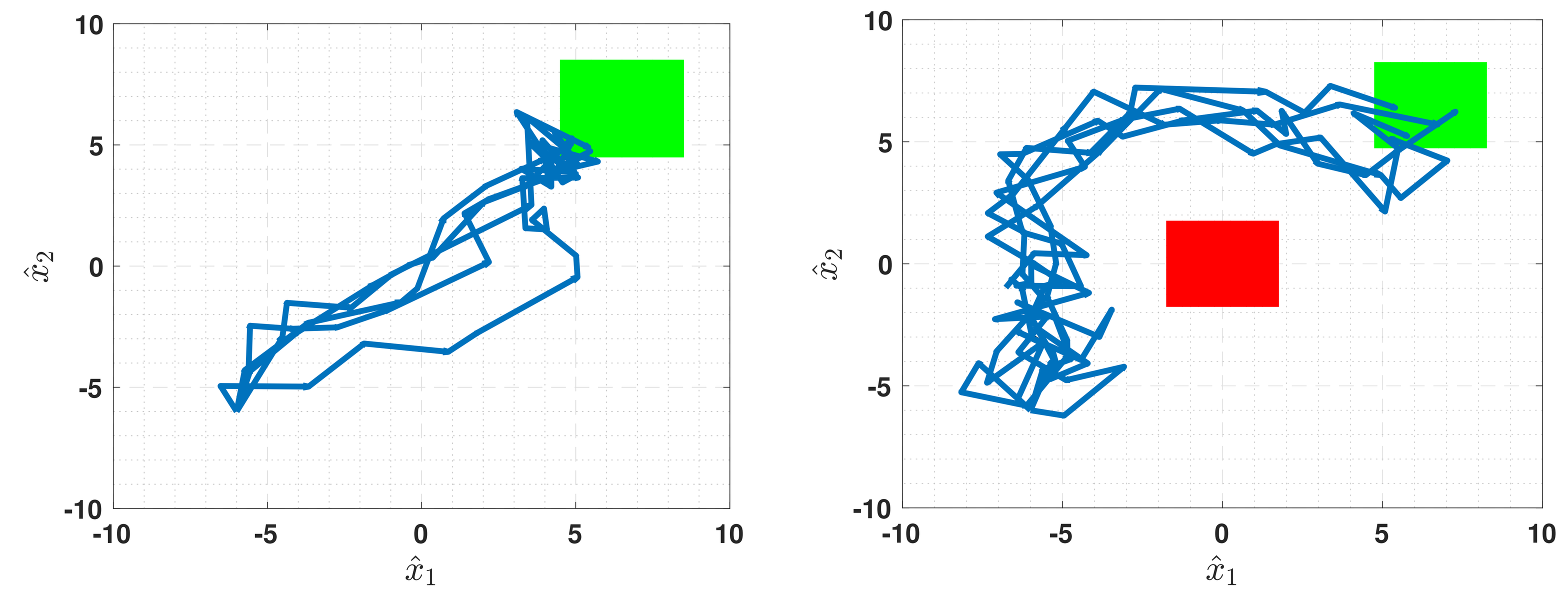
Loading and Saving

IMPaCT uses HDF5, which has native support in MATLAB, R, Python, etc. In loading, the IMDP can be constructed elsewhere, but synthesized with IMPaCT.

Main Contributions

- First tool to construct IMCs/IMDPs for large-scale discrete-time stochastic systems while providing convergence guarantees;
- Use constructed IMCs/IMDPs for formal verification and controller synthesis - safety, reachability, and reach-while-avoid properties over (in)finite horizon;
- Leverages interval iteration for convergence guarantees of optimal controller over infinite horizons;
- Implemented in C++ and runs in parallel using AdaptiveCpp based on SYCL. It provides automatic cross-platform flexibility, serving as a strong foundation for CPU and GPU implementations;
- We leverage IMPaCT across diverse real-world applications over (in)finite time horizons.

IMPaCT Examples: Diverse Real-World Applications



(a) 2D Robot - Reachability (b) 2D Robot - Reach-Avoid

Figure: 2D Robot case study fulfilling reachability and reach-avoid properties with different noise realizations.

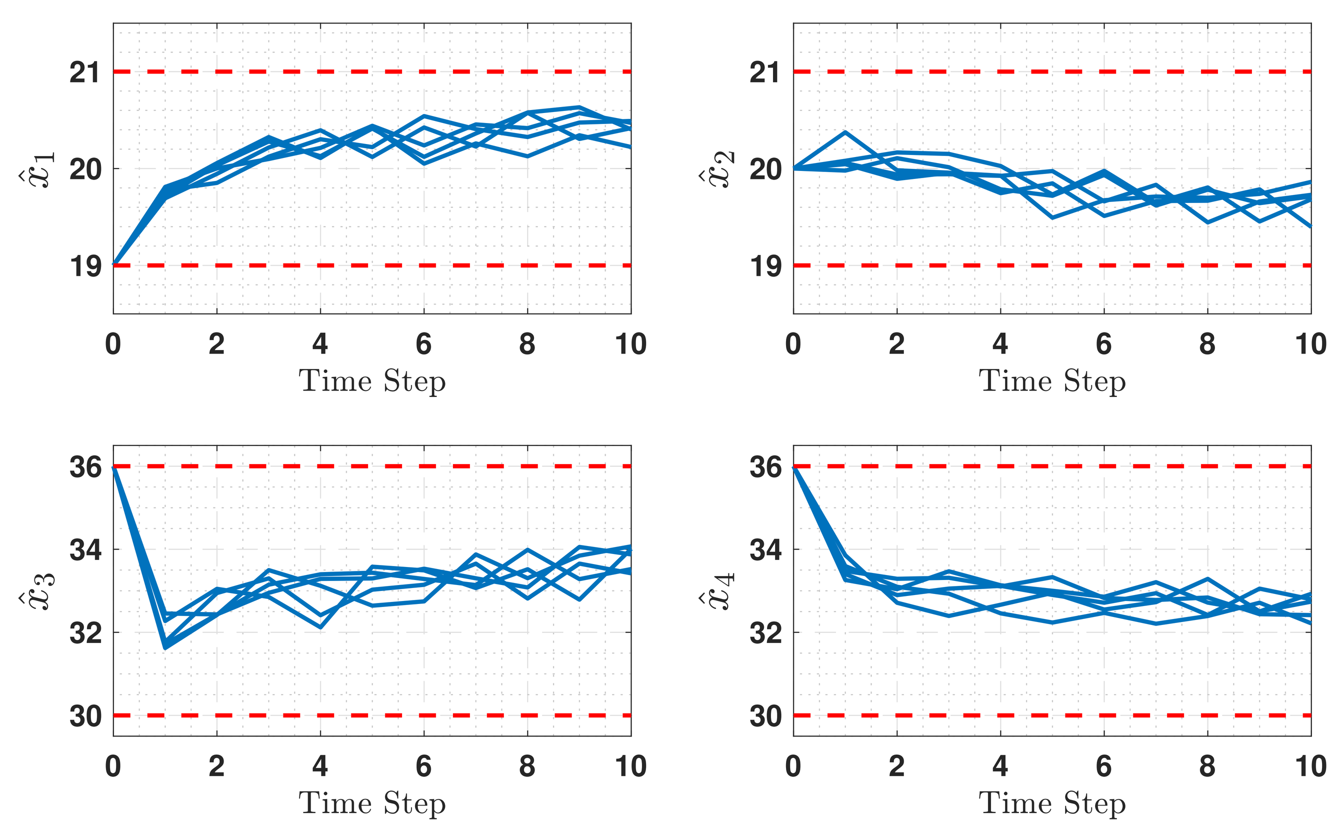


Figure: 4D Building Automation System fulfilling safety properties within **10** time steps, with **5** different noise realizations.

Benchmarking: CPU Abstraction and Synthesis

Table: Computation times are in seconds and memory usages in MB, unless otherwise specified, for a high performance computer with 2 AMD EPYC 7702 CPUs and 2TB RAM. **S** for safety, **R** for reachability, and **R − A** for reach-while-avoid. We signify the synthesis times using the GLPK Library with "a" and the synthesis times based on the sorting method with "b". Note that "*" indicates possible absorbing states.

Case Study	Spec	$\|\hat{X}\|$	$\|\hat{U}\|$	$\|\hat{W}\|$	$\|\hat{X} \times \hat{U} \times \hat{W}\|$	Abstraction	mem	Synthesis[a]	Synthesis[b]	mem
2D Robot	**R**	441	121	0	53,361	2.60	368	7.34	8.53	0.02
2D Robot	**R**	441	121	11	586,971	22.9	3.8GB	65.7	330	0.02
2D Robot	**R − A**	1,681	441	0	741,321	87.5	17.7GB	1,549	1,047	0.08
2D Robot	**R − A**	1,681	441	11	8,154,531	975	195GB	5.66hr	8.46hr	0.08
3D Vehicle	**R − A**	7,938	30	0	238,140	304	21.2GB	3.69hr	286	0.61
3D Vehicle	**R − A**	15,435	99	0	1,528,065	3,527	264GB	>24hr	5,933	1.22
3D RoomTemp	**S**	9,261	36	0	333,396	80.0	49.4GB	136*	154*	0.52
4D BAS	**S**	1,225	4	0	4,900	2.23	48.1	6.37hr*	3,038*	0.07
5D RoomTemp	**S**	7,776	36	0	279,936	169	34.8GB	**97.88***	**111.5***	0.56
7D BAS	**S**	107,163	0	0	107,163	3.5hr	184GB	>24hr	>24hr	7.7
14D Case	**S**	16,384	0	0	16,384	1,686	4.3GB	623	67.7	2.1

Benchmarking: CPU vs. GPU Synthesis

Table: Execution times for controller synthesis, conducted on both a CPU (Intel i9-12900) and a GPU (NVIDIA RTX A4000), with times reported in seconds. "∗" denotes a finite horizon of **10** seconds.

Case Study	Spec					GLPK Library	Sorted LP Method	
		$\|\hat{X}\|$	$\|\hat{U}\|$	$\|\hat{W}\|$	$\|\hat{X} \times \hat{U} \times \hat{W}\|$	CPU i9	CPU i9	GPU RTX
4D BAS*	**S**	1,225	4	0	4,900	23.2	0.38	0.53
3D RoomTemp*	**S**	216	36	0	7,776	0.34	0.22	0.29
2D Robot	**R**	441	121	0	53,361	13.07	2.26	5.2
5D RoomTemp*	**S**	7,776	9	0	69,984	160	22.2	76.2
3D Vehicle	**R − A**	7,938	30	0	238,140	>3.0hr	241	490
2D Robot	**R − A**	1,681	441	0	741,321	1691	577	216