Path Structured Multimarginal Schrödinger Bridge for Probabilistic Learning of Hardware Resource Usage by Control Software

Georgiy A. Bondar, Robert Gifford, Linh Thi Xuan Phan, Abhishek Halder

Abstract— **Solution of the path structured multimarginal Schrödinger bridge problem (MSBP) is the most-likely measure-valued trajectory consistent with a sequence of observed probability measures or distributional snapshots. We leverage recent algorithmic advances in solving such structured MSBPs for learning stochastic hardware resource usage by control software. The solution enables predicting the time-varying distribution of hardware resource availability at a desired time with guaranteed linear convergence. We demonstrate the efficacy of our probabilistic learning approach in a model predictive control software execution case study. The method exhibits rapid convergence to an accurate prediction of hardware resource utilization of the controller. The method can be broadly applied to any software to predict cyber-physical context-dependent performance at arbitrary time.**

I. Introduction

Control software in safety-critical cyber-physical systems (CPS) is often designed and verified based on platform models that do not fully capture the complexity of its deployment settings. For example, it is common to assume that the processor always operates at full speed, is dedicated to the control software, and that overheads are negligible. In practice, the hardware resources – such as last-level shared cache (LLC), memory bandwidth and processor cycles – often vary with time and depend on the current hardware state, which is a reason why we observe different execution times across different runs of the same control software [1]. This gap can lead to overly costly or unsafe design.

Measurement-based approaches and overhead-aware analysis can reduce the analysis pessimism or ensure safety [2]. The recent work [3] uses fine-grained profiles of the software execution on an actual platform to make dynamic scheduling and resource allocations. Supervisory algorithms that dynamically switch among a bank of controllers – all provably safe but some computationally more benign (and less performant) than others – depending on the resource availability also exist [4]. However, the effectiveness of these techniques is contingent on the quality of prediction of the hardware resource availability at a future instance or time horizon of interest.

Georgiy A. Bondar is with the Department of Applied Mathematics, University of California, Santa Cruz, CA 95064, USA, gbondar@ucsc.edu.
Robert Gifford and Linh Thi Xuan Phan are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA, {rgif, linhphan}@seas.upenn.edu.
Abhishek Halder is with the Department of Aerospace Engineering, Iowa State University, Ames, IA 50011, USA, ahalder@iastate.edu.
This work was supported by NSF grants 2112755, 2111688 and 1750158.

In this work, we propose to predict the resource usage by control software based on just a very small set of measurements. This approach is attractive as it can reduce measurement efforts while better handling potential variances.

A first-principle predictive model for hardware resource availability based on semiconductor physics of the specific platform is, however, unavailable. Furthermore, resources such as cache and bandwidth are not only time-varying and stochastic, but they are also statistically correlated. This makes it challenging to predict the *joint stochastic variability* of the hardware resource availability in general. The challenge is even more pronounced for control software because the computational burden then also depends on additional context, e.g., reference trajectory that the controller is tracking.

We note that for safety-critical CPS, predicting the joint stochastic hardware resource state, as opposed to predicting a lumped variable such as worst-case execution time, can open the door for designing a new class of dynamic scheduling algorithms with better performance than what is feasible today while minimizing hardware cost.

This work proposes learning a joint stochastic process for hardware resource availability from control software execution profiles conditioned on CPS contexts (to be made precise in Sec. III-A, III-B). Our proposed method leverages recent advances in *stochastic control* – specifically in the multimarginal Schrödinger bridge (MSBP) – to allow prediction of time-varying joint statistical distributions of hardware resource availability at any desired time.

Contributions: Our specific contributions are as follows.

- We show how recent algorithmic developments in solving the MSBP, enable probabilistic learning of hardware resources. This advances the state-of-the-art at the intersection of control, learning and real-time systems.
- The proposed method is statistically nonparametric, and is suitable for high-dimensional joint prediction since it avoids gridding the hardware feature/state space.
- The proposed formulation provably predicts the most likely distribution given a sequence of distributional snapshots for the hardware resource state.
- We explain that the resulting algorithm is an instance of the multimarginal Sinkhorn iteration with path structured cost that is guaranteed to converge to a unique solution, and enjoys linear rate of convergence. Its computational complexity scales linearly w.r.t. dimensions, linearly w.r.t. number of distributional snapshots, and quadratically w.r.t. number of scattered samples.

Organization: We introduce the notations and preliminaries in Sec. II. The problem formulation is detailed in Sec. III. Sec. IV summarizes the overall methodology, which is then followed by a numerical case study in Sec. V. Concluding remarks are provided in Sec. VI.

II. Notations and Preliminaries

We use unboldfaced capital letters to denote matrices and bold capital letters to denote tensors (of order three or more). Unboldfaced (resp. boldfaced) small letters are used to denote scalars (resp. vectors). Capital calligraphic letters are reserved to denote sets.

Square braces are used to denote the components. For instance, $[\boldsymbol{X}_{i_1,\ldots,i_r}]$ denotes the $(i_1,\ldots,i_r)$th component of the order r tensor $\boldsymbol{X}$, where $(i_1,\ldots,i_r) \in \mathbb{N}^r$. We use the r fold tensor product space notation $\left(\mathbb{R}^d\right)^{\otimes r} := \underbrace{\mathbb{R}^d \otimes \ldots \otimes \mathbb{R}^d}_{r \text{ times}}$.

For two tensors $\boldsymbol{X}, \boldsymbol{Y}$ of order r, we define their *Hilbert-Schmidt inner product* as

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle := \sum_{i_1,\ldots,i_r} [\boldsymbol{X}_{i_1,\ldots,i_r}]\,[\boldsymbol{Y}_{i_1,\ldots,i_r}]\,. \tag{1}$$

The operators $\exp(\cdot)$ and $\log(\cdot)$ are understood elementwise. We use $\odot$ and $\oslash$ to denote elementwise (Hadamard) multiplication and division, respectively.

For measures μ, ν defined on two Polish spaces, their product measure is denoted by $\mu \otimes \nu$. The *relative entropy* a.k.a. *Kullback-Leibler divergence* $D_{\rm KL}(\cdot\|\cdot)$ between probability measures μ and ν is

$$D_{\rm KL}(\mu\|\nu) := \begin{cases} \int \log \frac{{\rm d}\mu}{{\rm d}\nu} {\rm d}\mu & \text{if} \quad \mu \ll \nu, \\ +\infty & \text{otherwise,} \end{cases} \tag{2}$$

where $\frac{{\rm d}\mu}{{\rm d}\nu}$ denotes the Radon-Nikodym derivative, and $\mu \ll \nu$ is a shorthand for "μ is absolutely continuous w.r.t. ν".

The Hilbert (projective) metric (see e.g., [5]) $d_{\rm H}(\boldsymbol{u}, \boldsymbol{v})$ between two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n_{>0}$ is

$$d_{\rm H}(\boldsymbol{u}, \boldsymbol{v}) = \log\left(\frac{\max_{i=1,\ldots,n} u_i/v_i}{\min_{i=1,\ldots,n} u_i/v_i}\right). \tag{3}$$

We use the term "control cycle" to mean one pass of a feedback control loop. Due to hardware stochasticity, each control cycle completion takes variable amount of time.

III. Problem Formulation

A. Context $\boldsymbol{c}$

We consider a context vector $\boldsymbol{c}$ comprised of separable cyber and physical context vectors

$$\boldsymbol{c} := \begin{pmatrix} \boldsymbol{c}_{\rm cyber} \\ \boldsymbol{c}_{\rm phys} \end{pmatrix}. \tag{4}$$

In this work, we consider an instance of (4) where

$$\boldsymbol{c}_{\rm cyber} = \begin{pmatrix} \text{allocated last-level cache} \\ \text{allocated memory bandwidth} \end{pmatrix}, \tag{5}$$

where both features are allocated in blocks of some size, and

$$\boldsymbol{c}_{\rm phys} = y_{\rm des}(x) \in {\rm GP}\left([x_{\min}, x_{\max}]\right), \tag{6}$$

where GP denotes a Gaussian process over the domain $[x_{\min}, x_{\max}]$. We work with a collection of contexts with cardinality $n_{\rm context}$, i.e., a sample of contexts $\{\boldsymbol{c}^i\}_{i=1}^{n_{\rm context}}$.

B. Hardware Resource State $\boldsymbol{\xi}$

For concreteness, we define a hardware resource state or feature vector used in our numerical case study (Sec. V):

$$\boldsymbol{\xi} := \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} = \begin{pmatrix} \text{instructions retired} \\ \text{LLC requests} \\ \text{LLC misses} \end{pmatrix}. \tag{7}$$

The three elements of $\boldsymbol{\xi}$ denote the number of CPU instructions, the number of LLC requests, and the number of LLC misses in the last time unit (10 ms in our profiling), respectively.

We emphasize that our proposed method is not limited by what specific components comprise $\boldsymbol{\xi}$. To highlight this flexibility, we describe the proposed approach for $\boldsymbol{\xi} \in \mathbb{R}^d$ with suitable interpretations for the specific application.

For a time interval $[0, t]$ of interest, we think of time-varying $\boldsymbol{\xi}$ as a continuous time vector-valued stochastic process over subsets of $\mathbb{R}^d$. Suppose that $s \in \mathbb{N}, s \geq 2$ snapshots or observations are made for the stochastic state $\boldsymbol{\xi}(\tau)$, $0 \leq \tau \leq t$, at (possibly non-equispaced) instances

$$\tau_1 \equiv 0 < \tau_2 < \ldots < \tau_{s-1} < \tau_s \equiv t.$$

Consider the snapshot index set $[\![s]\!] := \{1, \ldots, s\}$. For a fixed context $\boldsymbol{c}$, the snapshot observations comprise a sequence of joint probability measures $\{\mu_\sigma\}_{\sigma \in [\![s]\!]}$ satisfying $\int {\rm d}\mu_\sigma(\boldsymbol{\xi}(\tau_\sigma)) = 1$. In other words,

$$\boldsymbol{\xi}(\tau_\sigma) \sim \mu_\sigma \quad \forall \sigma \in [\![s]\!]. \tag{8}$$

In our application, the data $\{\mu_\sigma\}_{\sigma \in [\![s]\!]}$ comes from control software execution profiles, i.e., by executing the same control software for the same $\boldsymbol{c}$ with all parameters and initial conditions fixed. So the stochasticity in $\boldsymbol{\xi}(\tau_\sigma)$ stems from the dynamic variability in hardware resource availability.

In particular, for finitely many (say n) execution profiles, we consider empirical distributions

$$\mu_\sigma := \frac{1}{n} \sum_{i=1}^{n} \delta(\boldsymbol{\xi} - \boldsymbol{\xi}^i(\tau_\sigma)), \tag{9}$$

where $\delta(\boldsymbol{\xi} - \boldsymbol{\xi}^i(\tau_\sigma))$ denotes the Dirac delta at sample location $\boldsymbol{\xi}^i(\tau_\sigma)$ where $i \in [\![n]\!]$, $\sigma \in [\![s]\!]$. At any snapshot index $\sigma \in [\![s]\!]$, the set $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$ is *scattered data*.

Given the data (8)-(9), we would like to predict the *most likely* hardware resource state statistics

$$\boldsymbol{\xi}(\tau) \sim \mu_\tau \quad \text{for any } \tau \in [0, t]. \tag{10}$$

Without the qualifier "most likely", the problem is overdetermined since there are uncountably many *measure-valued continuous curves* over $[0, t]$ that are consistent with the observed data (8)-(9).

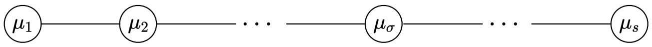

Fig. 1: The path tree for sequentially observed $\{\mu_\sigma\}_{\sigma\in[\![s]\!]}$.

C. Multimarginal Schrödinger Bridge

Let $\mathcal{X}_\sigma := \mathrm{support}\,(\mu_\sigma) \subseteq \mathbb{R}^d\ \forall\sigma\in[\![s]\!]$, and consider the Cartesian product $\mathcal{X}_1\times\mathcal{X}_2\times\ldots\times\mathcal{X}_s =: \boldsymbol{\mathcal{X}} \subseteq \left(\mathbb{R}^d\right)^{\otimes s}$. Let $\mathcal{M}\left(\mathcal{X}_\sigma\right)$ and $\mathcal{M}\left(\boldsymbol{\mathcal{X}}\right)$ denote the collection (i.e., manifold) of probability measures on $\mathcal{X}_\sigma$ and $\boldsymbol{\mathcal{X}}$, respectively. Define a ground cost $\boldsymbol{C}:\boldsymbol{\mathcal{X}}\mapsto\mathbb{R}_{\geq 0}$.

Following [6, Sec. 3], let

$$\mathrm{d}\boldsymbol{\xi}_{-\sigma} := \mathrm{d}\boldsymbol{\xi}(\tau_1)\times\ldots\times\mathrm{d}\boldsymbol{\xi}(\tau_{\sigma-1})\times\mathrm{d}\boldsymbol{\xi}(\tau_{\sigma+1})\times\ldots\times\mathrm{d}\boldsymbol{\xi}(\tau_s), \tag{11a}$$

$$\boldsymbol{\mathcal{X}}_{-\sigma} := \mathcal{X}_1\times\ldots\times\mathcal{X}_{\sigma-1}\times\mathcal{X}_{\sigma+1}\times\ldots\times\mathcal{X}_s. \tag{11b}$$

For $\varepsilon \geq 0$ (not necessarily small), the multimarginal Schrödinger bridge problem (MSBP) is the following infinite dimensional convex program:

$$\min_{\boldsymbol{M}\in\mathcal{M}(\boldsymbol{\mathcal{X}})}\int_{\boldsymbol{\mathcal{X}}}\left\{\boldsymbol{C}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s))+\varepsilon\log\boldsymbol{M}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s))\right\} \boldsymbol{M}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s))\,\mathrm{d}\boldsymbol{\xi}(\tau_1)\ldots\mathrm{d}\boldsymbol{\xi}(\tau_s) \tag{12a}$$

$$\text{subject to} \int_{\boldsymbol{\mathcal{X}}_{-\sigma}}\boldsymbol{M}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s))\,\mathrm{d}\boldsymbol{\xi}_{-\sigma} = \mu_\sigma\ \forall\sigma\in[\![s]\!]. \tag{12b}$$

In particular, $\mathcal{M}(\boldsymbol{\mathcal{X}})$ is a convex set. The objective (12a) is strictly convex in $\boldsymbol{M}$, thanks to the ε-regularized negative entropy term $\int_{\boldsymbol{\mathcal{X}}}\varepsilon\boldsymbol{M}\log\boldsymbol{M}$. The constraints (12b) are linear.

In this work, the measures $\{\mu_\sigma\}_{\sigma\in[\![s]\!]}$ correspond to sequential observation, and we therefore fix the *path structured* (Fig. 1) ground cost

$$\boldsymbol{C}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s)) = \sum_{\sigma=1}^{s-1} c_\sigma\left(\boldsymbol{\xi}(\tau_\sigma),\boldsymbol{\xi}(\tau_{\sigma+1})\right). \tag{13}$$

In particular, we choose the squared Euclidean distance sequential cost between two consecutive snapshot indices, i.e., $c_\sigma(\cdot,\cdot) := \|\cdot-\cdot\|_2^2\ \forall\sigma\in[\![s]\!]$. MSBPs with more general *tree structured* ground costs have appeared in [7].

When the cardinality of the index set $[\![s]\!]$ equals 2, then (12) reduces to the (bi-marginal) Schrödinger bridge problem (SBP) [8], [9]. In this case, the solution of (12) gives the *most likely evolution* between two marginal snapshots μ_1,μ_2. This can be established via the large deviations [10] interpretation [11, Sec. II] of SBP using Sanov's theorem [12]; see also [13, Sec. 2.1].

Specifically, let $\mathcal{C}\left([\tau_1,\tau_2],\mathbb{R}^d\right)$ denote the collection of continuous functions on the time interval $[\tau_1,\tau_2]$ taking values in $\mathbb{R}^d$. Let $\Pi(\mu_1,\mu_2)$ be the collection of all path measures on $\mathcal{C}\left([\tau_1,\tau_2],\mathbb{R}^d\right)$ with time τ_1 marginal μ_1, and time τ_2 marginal μ_2. Given a symmetric ground cost (e.g., Euclidean distance) $C:\mathcal{X}_1\times\mathcal{X}_2\mapsto\mathbb{R}_{\geq 0}$, let

$$K(\cdot,\cdot) := \exp\left(-\frac{C(\cdot,\cdot)}{\varepsilon}\right), \tag{14}$$

and consider the *bimarginal Gibbs kernel*

$$K\left(\boldsymbol{\xi}(\tau_1),\boldsymbol{\xi}(\tau_2)\right)\mu_1\otimes\mu_2. \tag{15}$$

Then, the bimarginal SBP solves

$$\min_{\pi\in\Pi(\mu_1,\mu_2)}\varepsilon D_{\mathrm{KL}}\left(\pi\,\|\,K\left(\boldsymbol{\xi}(\tau_1),\boldsymbol{\xi}(\tau_2)\right)\mu_1\otimes\mu_2\right), \tag{16}$$

i.e., the most likely evolution of the path measure consistent with the observed measure-valued snapshots μ_1,μ_2.

Under the stated assumptions on the ground cost c, the existence of minimizer for (16) is guaranteed [14], [15]. The uniqueness of minimizer follows from strict convexity of the map $\pi\mapsto D_{\mathrm{KL}}(\pi\|\nu)$ for fixed ν.

This relative entropy reformulation, and thereby "the most likely evolution consistent with observed measures" interpretation, also holds for the MSBP (12) with $s\geq 2$ snapshots. Specifically, for $\boldsymbol{C}:\boldsymbol{\mathcal{X}}\mapsto\mathbb{R}_{\geq 0}$ as in (12)-(13), we generalize (14) as

$$\boldsymbol{K}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s)) := \exp\left(-\frac{\boldsymbol{C}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s))}{\varepsilon}\right), \tag{17}$$

and define the *multimarginal Gibbs kernel*

$$\boldsymbol{K}\left(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s)\right)\mu_1\otimes\ldots\otimes\mu_s. \tag{18}$$

Problem (16) then generalizes to

$$\min_{\pi\in\Pi(\mu_1,\ldots,\mu_s)}\varepsilon D_{\mathrm{KL}}\left(\pi\,\|\,\boldsymbol{K}\left(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s)\right)\mu_1\otimes\ldots\otimes\mu_s\right) \tag{19}$$

where $\Pi(\mu_1,\ldots,\mu_s)$ denotes the collection of all path measures on $\mathcal{C}\left([\tau_1,\tau_s],\mathbb{R}^d\right)$ with time τ_σ marginal $\mu_\sigma\ \forall\sigma\in[\![s]\!]$. The equivalence between (12) and (19) can be verified by direct computation. Thus solving (19), or equivalently (12), yields the most likely evolution of the path measure consistent with the observed measure-valued snapshots μ_σ $\forall\sigma\in[\![s]\!]$.

We propose to solve the MSBP (12) for learning the time-varying statistics of the hardware resource state $\boldsymbol{\xi}$ as in (10). We next detail a discrete formulation to numerically solve the same for scattered data $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$ where n is the number of control software execution profiles.

The minimizer of (12), $\boldsymbol{M}_{\mathrm{opt}}\left(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s)\right)$ can be used to compute the optimal coupling between snapshot index pairs $(\sigma_1,\sigma_2)\in\{[\![s]\!]^{\otimes 2}\mid\sigma_1<\sigma_2\}$ as

$$\int_{\boldsymbol{\mathcal{X}}_{-\sigma_1,-\sigma_2}}\boldsymbol{M}_{\mathrm{opt}}(\boldsymbol{\xi}(\tau_1),\ldots,\boldsymbol{\xi}(\tau_s))\,\mathrm{d}\boldsymbol{\xi}_{-\sigma_1,-\sigma_2} \tag{20}$$

where

$$\mathrm{d}\boldsymbol{\xi}_{-\sigma_1,-\sigma_2} := \prod_{\sigma\in[\![s]\!]\setminus\{\sigma_1,\sigma_2\}}\mathrm{d}\boldsymbol{\xi}(\tau_\sigma), \tag{21a}$$

$$\boldsymbol{\mathcal{X}}_{-\sigma_1,-\sigma_2} := \prod_{\sigma\in[\![s]\!]\setminus\{\sigma_1,\sigma_2\}}\mathcal{X}_\sigma. \tag{21b}$$

This will be useful for predicting the statistics of $\boldsymbol{\xi}(\tau)\sim\mu_\tau$ at any (out-of-sample) query time $\tau\in[0,t]$.

Remark 1. ***(MSBP and MOT)*** *When the entropic regularization strength $\varepsilon=0$, then the MSBP (12) reduces to the multimarginal optimal transport (MOT) problem [16], [17] that has found widespread applications in barycenter computation [18], fluid dynamics [19], [20], team matching problems [21], and density functional theory [22], [23]. Further specializing MOT with the cardinality of $[\![s]\!]$ equals 2, yields the (bimarginal) optimal transport [24] problem.*

D. Discrete Formulation of MSBP

For finite scattered data $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$ and $\{\mu_\sigma\}_{\sigma\in[\![s]\!]}$ as in (9), we set up a discrete version of (12) as follows.

With slight abuse of notations, we use the same symbol for the continuum and discrete version of a tensor. The ground cost in discrete formulation is represented by an order s tensor $\boldsymbol{C}\in(\mathbb{R}^n)^{\otimes s}_{\geq 0}$, with components $[\boldsymbol{C}_{i_1,\dots,i_s}]=\boldsymbol{C}(\boldsymbol{\xi}_{i_1},\dots,\boldsymbol{\xi}_{i_s})$. The component $[\boldsymbol{C}_{i_1,\dots,i_s}]$ encodes the cost of transporting unit mass for a tuple $(i_1,\dots,i_s)$.

Likewise, consider the discrete mass tensor $\boldsymbol{M}\in(\mathbb{R}^n)^{\otimes s}_{\geq 0}$ with components $[\boldsymbol{M}_{i_1,\dots,i_s}]=\boldsymbol{M}(\boldsymbol{\xi}_{i_1},\dots,\boldsymbol{\xi}_{i_s})$. The component $[\boldsymbol{M}_{i_1,\dots,i_s}]$ denotes the amount of transported mass for a tuple $(i_1,\dots,i_s)$.

For any $\sigma\in[\![s]\!]$, the empirical marginals $\boldsymbol{\mu}_\sigma\in\mathbb{R}^n_{\geq 0}$ are supported on the finite sets $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$. We denote the projection of $\boldsymbol{M}\in(\mathbb{R}^n)^{\otimes s}_{\geq 0}$ on the σth marginal as $\mathrm{proj}_\sigma(\boldsymbol{M})$. Thus $\mathrm{proj}_\sigma:(\mathbb{R}^n)^{\otimes s}_{\geq 0}\mapsto\mathbb{R}^n_{\geq 0}$, and is given componentwise as

$$\left[\mathrm{proj}_\sigma(\boldsymbol{M})_j\right]=\sum_{i_1,\dots,i_{\sigma-1},i_{\sigma+1},\dots,i_s}\boldsymbol{M}_{i_1,\dots,i_{\sigma-1},j,i_{\sigma+1},\dots,i_s}. \quad (22)$$

Likewise, denote the projection of $\boldsymbol{M}\in(\mathbb{R}^n)^{\otimes s}_{\geq 0}$ on the (σ_1,σ_2)th marginal as $\mathrm{proj}_{\sigma_1,\sigma_2}(\boldsymbol{M})$, i.e., $\mathrm{proj}_{\sigma_1,\sigma_2}:(\mathbb{R}^n)^{\otimes s}_{\geq 0}\mapsto\mathbb{R}^{n\times n}_{\geq 0}$, and is given componentwise as

$$\left[\mathrm{proj}_{\sigma_1,\sigma_2}(\boldsymbol{M})_{j,\ell}\right] = \sum_{i_\sigma\mid\sigma\in[\![s]\!]\setminus\{\sigma_1,\sigma_2\}}\boldsymbol{M}_{i_1,\dots,i_{\sigma_1-1},j,i_{\sigma_1+1},\dots,i_{\sigma_2-1},\ell,i_{\sigma_2+1},\dots,i_s}. \quad (23)$$

We note that (22) and (23) are the discrete versions of the integrals in (12b) and (20), respectively.

With the above notations in place, the discrete version of (12) becomes

$$\min_{\boldsymbol{M}\in(\mathbb{R}^n)^{\otimes s}_{\geq 0}}\ \langle\boldsymbol{C}+\varepsilon\log\boldsymbol{M},\boldsymbol{M}\rangle \quad (24a)$$

$$\text{subject to}\quad \mathrm{proj}_\sigma(\boldsymbol{M})=\boldsymbol{\mu}_\sigma\quad\forall\sigma\in[\![s]\!]. \quad (24b)$$

The primal formulation (24) has n^s decision variables, and is computationally intractable. Recall that even for the bimarginal ($s=2$) case, a standard approach [25] is to use Lagrange duality to notice that the optimal mass matrix M_{opt} is a diagonal scaling of $K:=\exp(-C/\varepsilon)\in\mathbb{R}^{n\times n}_{>0}$, i.e., $M_{\mathrm{opt}}=\mathrm{diag}(\boldsymbol{u}_1)K\mathrm{diag}(\boldsymbol{u}_2)$ where $\boldsymbol{u}_1:=\exp(\boldsymbol{\lambda}_1/\varepsilon)$, $\boldsymbol{u}_2:=\exp(\boldsymbol{\lambda}_2/\varepsilon)$, and $\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2\in\mathbb{R}^n$ are the Lagrange multipliers associated with respective bimarginal constraints $\mathrm{proj}_1(M)=\boldsymbol{\mu}_1$, $\mathrm{proj}_2(M)=\boldsymbol{\mu}_2$. The unknowns $\boldsymbol{u}_1,\boldsymbol{u}_2$ can be obtained by performing the Sinkhorn iterations

$$\boldsymbol{u}_1\leftarrow\boldsymbol{\mu}_1\oslash(K\boldsymbol{u}_2), \quad (25a)$$

$$\boldsymbol{u}_2\leftarrow\boldsymbol{\mu}_2\oslash\left(K^\top\boldsymbol{u}_1\right), \quad (25b)$$

with guaranteed linear convergence [26] wherein the computational cost is governed by two matrix-vector multiplications.

The duality result holds for the multimarginal ($s\geq 2$) case. Specifically, the optimal mass tensor in (24) admits a structure $\boldsymbol{M}_{\mathrm{opt}}=\boldsymbol{K}\odot\boldsymbol{U}$ where $\boldsymbol{K}:=\exp(-\boldsymbol{C}/\varepsilon)\in(\mathbb{R}^n)^{\otimes s}_{>0}$, $\boldsymbol{U}:=\otimes_{\sigma=1}^s\boldsymbol{u}_\sigma\in(\mathbb{R}^n)^{\otimes s}_{>0}$, $\boldsymbol{u}_\sigma:=\exp(\boldsymbol{\lambda}_\sigma/\varepsilon)$, and $\boldsymbol{\lambda}_\sigma\in\mathbb{R}^n$ are the Lagrange multipliers associated with the respective multimarginal constraints (24b). The unknowns $\boldsymbol{u}_\sigma$ can, in principle, be obtained from the multimarginal Sinkhorn iterations [27]

$$\boldsymbol{u}_\sigma\leftarrow\boldsymbol{u}_\sigma\odot\boldsymbol{\mu}_\sigma\oslash\mathrm{proj}_\sigma(\boldsymbol{K}\odot\boldsymbol{U})\ \ \forall\sigma\in[\![s]\!], \quad (26)$$

which generalize (25). However, computing $\mathrm{proj}_\sigma(\boldsymbol{K}\odot\boldsymbol{U})$ requires $\mathcal{O}(n^s)$ operations. Before describing how to avoid this exponential complexity (Sec. III-F), we point out the convergence guarantees for (26).

E. Convergence for Multimarginal Sinkhorn Iterations

The iterations (26) can either be derived as alternating Bregman projections [27] or via block coordinate dual ascent [6]. Following either viewpoints leads to guaranteed linear convergence of (26); see [28], [7, Thm. 3.5].

More recent works have also established [29] guaranteed convergence for the continuous formulation (12) with linear rate of convergence [30].

F. Multimarginal Sinkhorn Iterations for Path Structured $\boldsymbol{C}$

We circumvent the exponential complexity in computing $\mathrm{proj}_\sigma(\boldsymbol{K}\odot\boldsymbol{U})$ in (26) by leveraging the path structured ground cost (13). This is enabled by a key result from [6], rephrased, and reproved below in slightly generalized form.

Proposition 1. *([6, Prop. 2]) Consider the discrete ground cost tensor $\boldsymbol{C}$ in* (24) *induced by a path structured cost* (13) *so that $[\boldsymbol{C}_{i_1,\dots,i_s}]=\sum_{\sigma=1}^{s-1}\left[C^{\sigma\to\sigma+1}_{i_\sigma,i_{\sigma+1}}\right]$ where the matrix $C^{\sigma\to\sigma+1}\in\mathbb{R}^{n\times n}_{\geq 0}$ encodes the cost of transporting unit mass between each source-destination pair from the source set $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$ to the destination set $\{\boldsymbol{\xi}^i(\tau_{\sigma+1})\}_{i=1}^n$.*

Let $K^{\sigma\to\sigma+1}:=\exp(-C^{\sigma\to\sigma+1}/\varepsilon)\in\mathbb{R}^{n\times n}_{\geq 0}$, $\boldsymbol{K}:=\exp(-\boldsymbol{C}/\varepsilon)\in(\mathbb{R}^n)^{\otimes s}_{>0}$, $\boldsymbol{U}:=\otimes_{\sigma=1}^s\boldsymbol{u}_\sigma\in(\mathbb{R}^n)^{\otimes s}_{>0}$.

Then (22) *and* (23) *can be expressed as*

$$\mathrm{proj}_\sigma(\boldsymbol{K}\odot\boldsymbol{U})=\left(\boldsymbol{u}_1^\top K^{1\to 2}\prod_{j=2}^{\sigma-1}\mathrm{diag}(\boldsymbol{u}_j)K^{j\to j+1}\right)^\top\odot\boldsymbol{u}_\sigma\odot \left(\left(\prod_{j=\sigma+1}^{s-1}K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right)K^{s-1\to s}\boldsymbol{u}_s\right)\ \forall\sigma\in[\![s]\!], \quad (27)$$

and

$$\mathrm{proj}_{\sigma_1,\sigma_2}(\boldsymbol{K}\odot\boldsymbol{U})=\mathrm{diag}\left(\boldsymbol{u}_1^\top K^{1\to 2}\prod_{j=2}^{\sigma_1-1}\mathrm{diag}(\boldsymbol{u}_j)K^{j\to j+1}\right) \mathrm{diag}(\boldsymbol{u}_{\sigma_1})\prod_{j=\sigma_1+1}^{\sigma_2}\left(K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right) \mathrm{diag}\left(\left(\prod_{j=\sigma_2+1}^{s-1}K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right)K^{s-1\to s}\boldsymbol{u}_s\right) \forall(\sigma_1,\sigma_2)\in\{[\![s]\!]^{\otimes 2}\mid\sigma_1<\sigma_2\}. \quad (28)$$

Proof. The proof strategy is to write the Hilbert-Schmidt inner product $\langle \boldsymbol{K}, \boldsymbol{U}\rangle$ in two different ways.

First, recall that $\boldsymbol{K} := \exp(-\boldsymbol{C}/\varepsilon) \in (\mathbb{R}^n)^{\otimes s}_{>0}$ and $\boldsymbol{U} := \otimes_{\sigma=1}^{s} \boldsymbol{u}_\sigma \in (\mathbb{R}^n)^{\otimes s}_{>0}$. So following (1), we have

$$\begin{aligned}\langle \boldsymbol{K}, \boldsymbol{U}\rangle &= \sum_{i_1,\dots,i_s} \left(\prod_{j=2}^{s}\left[K^{j-1\to j}_{i_{j-1},i_j}\right]\right)\prod_{j=1}^{s}(\boldsymbol{u}_j)_{i_j}\\ &= \sum_{i_1,\dots,i_s}(\boldsymbol{u}_1)_{i_1}\prod_{j=2}^{s}\left[K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right]_{i_{j-1},i_j}\\ &= \boldsymbol{u}_1^\top\left(\prod_{j=2}^{s-1}K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right)K^{s-1\to s}\boldsymbol{u}_s,\end{aligned}$$

and (27) follows from [6, Lemma 1 in Appendix 1].

Next, notice that we can alternatively write

$$\begin{aligned}\langle \boldsymbol{K}, \boldsymbol{U}\rangle = \boldsymbol{u}_1^\top &\left(\prod_{j=2}^{\sigma_1-1}K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right)K^{\sigma_1-1\to\sigma_1}\\ &\left(\prod_{j=\sigma_1}^{\sigma_2-1}\mathrm{diag}(\boldsymbol{u}_j)K^{j\to j+1}\right)\mathrm{diag}(\boldsymbol{u}_{\sigma_2})\\ &\left(\prod_{j=\sigma_2+1}^{s-1}K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right)K^{s-1\to s}\boldsymbol{u}_s.\end{aligned}$$

Then (28) follows from [6, Lemma 2 in Appendix 1]. ■

Remark 2. *Unlike [6, Prop. 2], our data $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$ are scattered, i.e., not on a fixed grid, hence the need for superscripts $\sigma \to \sigma+1$ for the time-varying matrices in our Prop. 1. In contrast, the corresponding matrices in [6, Prop. 2] are independent of σ.*

Remark 3. *We note that substituting* (27) *into* (26) *cancels the (elements of) positive vectors $\boldsymbol{u}_\sigma$ $\forall\sigma\in[\![s]\!]$ from the corresponding numerators and denominators. This further simplifies our multimarginal Sinkhorn recursions to*

$$\begin{aligned}\boldsymbol{u}_\sigma \leftarrow \boldsymbol{\mu}_\sigma \oslash &\left(\left(\boldsymbol{u}_1^\top K^{1\to 2}\prod_{j=2}^{\sigma-1}\mathrm{diag}(\boldsymbol{u}_j)K^{j\to j+1}\right)^\top\right.\\ &\left.\odot\left(\left(\prod_{j=\sigma+1}^{s-1}K^{j-1\to j}\mathrm{diag}(\boldsymbol{u}_j)\right)K^{s-1\to s}\boldsymbol{u}_s\right)\right)\ \forall\sigma\in[\![s]\!].\end{aligned} \tag{29}$$

Remark 4. ***(From exponential to linear complexity in** s**)*** *We note that* (29) *involves $s-1$ matrix-vector multiplications each of which has $\mathcal{O}(n^2)$ complexity. So the computational complexity for* (29) *becomes $\mathcal{O}\left((s-1)n^2\right)$ which is linear in s, i.e., a significant reduction from earlier $\mathcal{O}\left(n^s\right)$ complexity mentioned at the end of Sec. III-D.*

Remark 5. ***(Linear complexity in** d**)*** *The dimension d of the vector $\boldsymbol{\xi}$ only affects the construction of the time-varying Euclidean distance matrices $C^{\sigma\to\sigma+1}$ $\forall\sigma\in[\![s-1]\!]$ in Prop. 1, which has total complexity $\mathcal{O}(sd)$. Once constructed, the recursions* (29) *are independent of d.*

We next outline how the solution tensor $\boldsymbol{M}_{\mathrm{opt}} = \boldsymbol{K}\odot\boldsymbol{U}$ obtained from the converged Sinkhorn iterations can be used together with (28), to make stochastic predictions of the most likely hardware resource state in the form (10).

G. Predicting Most Likely Distribution

For the ground cost (13) resulting from sequential information structure (Fig. 1), we utilize (28) to decompose $\boldsymbol{M}_{\mathrm{opt}} = \boldsymbol{K}\odot\boldsymbol{U}$ of (24) into bimarginal transport plans

$$M^{\sigma_1\to\sigma_2} := \mathrm{proj}_{\sigma_1,\sigma_2}(\boldsymbol{M}_{\mathrm{opt}}) = \mathrm{proj}_{\sigma_1,\sigma_2}(\boldsymbol{K}\odot\boldsymbol{U}). \tag{30}$$

Further, when $\boldsymbol{C}$ is squared Euclidean, as we consider here, the maximum likelihood estimate for μ_τ in (10) for a query point $\tau\in[0,t]$, is (see [6, Sec. 2.2])

$$\hat{\mu}_\tau := \sum_{i=1}^{n}\sum_{j=1}^{n}\left[M^{\sigma\to\sigma+1}_{i,j}\right]\delta(\boldsymbol{\xi}-\hat{\boldsymbol{\xi}}(\tau,\boldsymbol{\xi}^i(\tau_\sigma),\boldsymbol{\xi}^j(\tau_{\sigma+1}))) \tag{31}$$

where $\sigma\in[\![s]\!]$ such that $\tau\in[\tau_\sigma,\tau_{\sigma+1}]$, and

$$\hat{\boldsymbol{\xi}}(\tau,\boldsymbol{\xi}^i(\tau_\sigma),\boldsymbol{\xi}^j(\tau_{\sigma+1})) := (1-\lambda)\boldsymbol{\xi}^i(\tau_\sigma)+\lambda\boldsymbol{\xi}^j(\tau_{\sigma+1}), \tag{32a}$$

$$\lambda := \frac{\tau-\tau_\sigma}{\tau_{\sigma+1}-\tau_\sigma}\in[0,1]. \tag{32b}$$

IV. Overall Algorithm

The methodology proposed in Sec. III is comprised of the following three overall steps.

Step 1. Given a collection of contexts (Sec. III-A) $\{\boldsymbol{c}^i\}_{i=1}^{n_{\mathrm{context}}}$, execute the control software over $[0,t]$ to generate hardware resource state sample snapshots (Sec. III-B) $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$, and thereby empirical μ_σ as in (9) for all $\sigma\in[\![s]\!]$, conditional on each of the n_{context} context samples.

Step 2. Using data from **Step 1**, construct Euclidean distance matrices $C^{\sigma\to\sigma+1}$ from the source set $\{\boldsymbol{\xi}^i(\tau_\sigma)\}_{i=1}^n$ to the destination set $\{\boldsymbol{\xi}^i(\tau_{\sigma+1})\}_{i=1}^n$ $\forall\sigma\in[\![s-1]\!]$. Perform recursions (29) until convergence (error within desired tolerance).

Step 3. Given a query context $\boldsymbol{c}$ and time $\tau\in[0,t]$, return most likely distribution $\hat{\mu}_\tau$ using (31).

Remark 6. *For the three steps mentioned above,* ***Step 1*** *is data generation,* ***Step 2*** *is probabilistic learning using data from* ***Step 1****, and* ***Step 3*** *is prediction using the learnt model.*

V. Numerical Case Study

In this Section, we illustrate the application of the proposed method for a vehicle path tracking control software. All along, we provide details for the three steps in Sec. IV.

Control Software. We wrote custom software[1] in C language implementing path following nonlinear model predictive controller (NMPC) for a kinematic bicycle model (KBM) [32], [33] of a vehicle with four states (x,y,v,ψ) and two control inputs (a_c,δ), given by $\dot{x}=v\cos(\psi+\beta), \dot{y}=v\sin(\psi+\beta), \dot{v}=a_c, \dot{\psi}=\frac{v}{\ell_{\text{rear}}}\sin\beta$, where the sideslip angle $\beta=$

[1]Git repo: `https://github.com/abhishekhalder/CPS-Frontier-Task3-Collaboration`

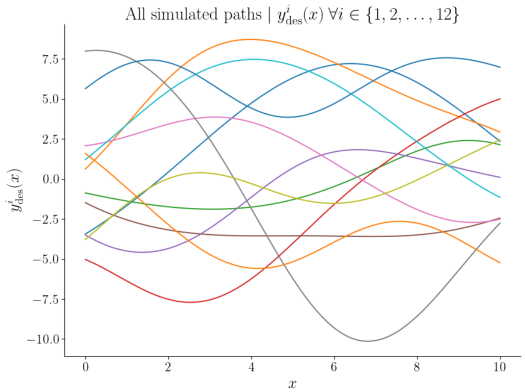

Fig. 2: All 12 paths used in profiling the NMPC software (Sec. V), generated by GP sampling via Scikit-learn [31] over the domain $[0, 10]$ using mean zero and variance 10.

$\arctan\left(\frac{\ell_{\text{rear}}}{\ell_{\text{front}}+\ell_{\text{rear}}}\tan\delta\right)$. The 4×1 state vector comprises of the inertial position (x, y) for the vehicle's center-of-mass, its speed v, and the vehicle's inertial heading angle ψ. The 2×1 control vector comprises of the acceleration a_c, and the front steering wheel angle δ.

The parameters ℓ_{front} , ℓ_{rear} are the distances of the vehicle's center-of-mass to the front and rear axles, respectively.

The NMPC was designed to track a desired path given as a sequence of $N = 200$ waypoint tuples $\left\{(x_d^{(i)}, y_d^{(i)}, v_d^{(i)})\right\}_{i=1}^{N}$, i.e., a sequence of desired positions and speeds (desired speed profile was numerically estimated from the desired waypoint profiles). At every control step (at most every 100 ms), using the IPOPT nonlinear program solver [34], the NMPC solved an optimization problem to minimize the sum of the crosstrack, ψ, and v errors, along with the magnitude and rapidity in change of the control inputs, over a period of time from 0 to the time horizon $H_p = 4$, subject to control magnitude and slew rate constraints. For formulation details and control performance achieved, see [35]. For implementation and parameter values, we refer the readers to the Git repository in the footnote.

While closing the loop with KBM with computed control values requires minimal computational overhead, the NMPC is computationally demanding. In the case where multiple vehicle controllers are available, it is of practical interest to predict the hardware resource usage for the NMPC for one to several control cycles, conditional on the CPS context $\boldsymbol{c}$ (Sec. III-A) at a given time. For this we 'profile' the NMPC, meaning we run the software many times for different values of $\boldsymbol{c}$ as in (4), measuring time evolution of the hardware resource state $\boldsymbol{\xi}$ as in (7). We use these profiles to generate marginals μ_σ as in (9), thus completing **Step 1** in Sec. IV.

We next provide details on generating control software execution profiles for our specific case study.

Generating Execution Profiles. To gather the execution profiles for our NMPC control software, we used an Ubuntu 16.04.7 Linux machine with an Intel Xeon E5-2683 v4 CPU. We leveraged Intel's Cache Allocation Technology (CAT) [36] and Memguard [37] to control allocation of LLC partitions and memory bandwidth available to the control software, respectively. Both LLC partitions and memory bandwidth were allocated in blocks of 2MB.

Utilizing these resource partitioning mechanisms, we ran our application on an isolated CPU and used the Linux perf tool [38], version 4.9.3, to sample $\boldsymbol{\xi}$ every 10 ms.

For each run of our application, we set the cache and memory bandwidth to a static allocation and pass as input a path for the NMPC to follow, represented as an array of desired (x, y) coordinates. We then execute the control software for $n_c := 5$ uninterrupted "control cycles", wherein the NMPC gets the KBM state, makes a control decision, and updates the KBM state.

We profile over 12 unique desired paths to track, denoted $\{y^i_{\text{des}}(x)\}_{i=1}^{12}$, and 5 unique vectors of $\{\boldsymbol{c}^i_{\text{cyber}}\}_{i=1}^{5}$, comprising $n_{\text{context}} = 12 \times 5 = 60$ samples for $\boldsymbol{c}$. Conditional on each of these 60 context samples $\{\boldsymbol{c}^i\}_{i=1}^{n_{\text{context}}=60}$, we run the software for 500 profiles for each unique $\boldsymbol{c}$ for a total of 30,000 profiles.

The sample paths $\{y^i_{\text{des}}(x)\}_{i=1}^{12}$ in (6) were all generated for $x \in [0, 10]$ using a GP with mean zero and variance 10 [31], and are shown in Fig. 2.

Our vectorial samples $\{\boldsymbol{c}^i_{\text{cyber}}\}_{i=1}^{5}$ in (5) were $[1, 1]^\top$, $[5, 5]^\top$, $[10, 10]^\top$, $[15, 15]^\top$, and $[20, 20]^\top$, where each entry represents the number of cache/memory bandwidth partitions from 1 to 20. These values were selected to broadly cover the range of possible hardware contexts.

Control cycle	Mean	Standard deviation
#1	0.1181	0.0076
#2	0.2336	0.0106
#3	0.3495	0.0127
#4	0.4660	0.0143
#5	0.5775	0.0159

TABLE I: The means and standard deviations of the end times for the $n_c = 5$ control cycles data shown in Fig. 4.

s_{int}	W_1	W_2	W_3	W_4	W_5
0	2.0489	-	-	-	-
1	2.2695	1.1750	-	-	-
2	5.7717	0.9163	0.3794	-	-
3	2.2413	1.6432	1.2345	0.6010	-
4	0.6372	1.2691	0.9176	0.6689	0.2111

TABLE II: Number of intracycle marginals s_{int} vs. Wasserstein distances W_j as in (33). All entries are scaled up by 10^4.

Applying the Proposed Algorithm. Given a query context $\boldsymbol{c}$, we determine the closest CPS context for which profiling data is available, using the Euclidean distance between cyber context vectors (5), and the Fréchet distance [41] between physical context curves (6). In this case study, we consider a query context with closest $\boldsymbol{c}_{\text{cyber}} = \left[15, 15\right]^\top$ and closest $\boldsymbol{c}_{\text{phys}} = y^1_{\text{des}}(x)$. Profiling data for this $\boldsymbol{c}$ is shown in Fig. 3.

For each of the $n = 500$ profiles, we are given the end times for each of the $n_c = 5$ control cycles. We then determine the statistics of the cycle end times (Fig. 4) and compute the empirical distributions of $\boldsymbol{\xi}$ at the means (Table I) of the control cycle start/end time boundaries. For empirical distributions at times between cycle boundaries, we let s_{int} be the number of marginals equispaced-in-time

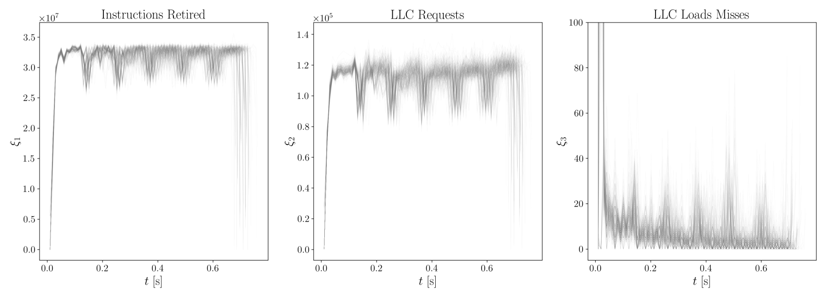

Fig. 3: Components of the measured feature vector $\boldsymbol{\xi}$ in (7) for all of the five control cycles for 500 executions of the NMPC software, where $\boldsymbol{c} = [15, 15, y^1_{\mathrm{des}}(x)]$.

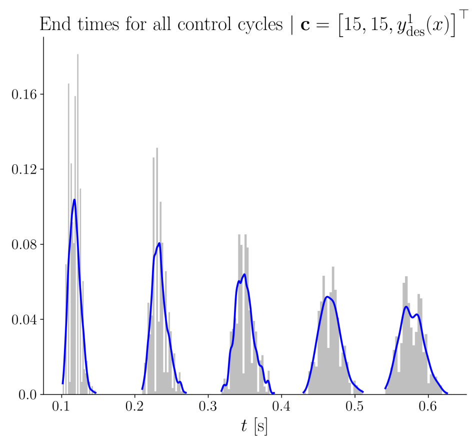

Fig. 4: Normalized histograms (*gray filled*) and kernel density estimates (KDEs) (*solid line*) for the end times of all of the five control cycles for 500 executions of the NMPC software conditioned on a fixed CPS context $\boldsymbol{c}$ shown above. The KDEs used Gaussian kernel with bandwidths computed via cross validation [39], [40].

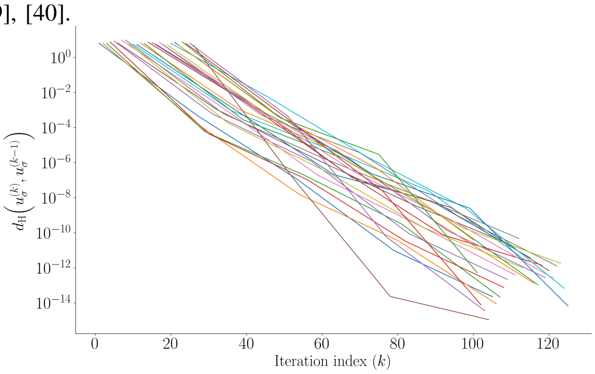

Fig. 5: Linear convergence of Sinkhorn iterations (29) for $s_{\mathrm{int}} = 4$ w.r.t. the Hilbert's projective metric d_{H} in (3) between $u_{\sigma \in [\![s]\!]}$ at iteration indices k and $k-1$.

between each cycle boundary. We then set $\tau_{\sigma \in [\![s]\!]}$ from the means in Table I, where $s := 1 + n_c(s_{\mathrm{int}} + 1)$ and $\tau_{\sigma(s_{\mathrm{int}}+1)+1}$ is the sampled mean end time for the σth control cycle.

Our distributions are as per (9), where $\boldsymbol{\xi}^i(\tau_\sigma)$ is the sample of the hardware resource state (7) at time $\approx \tau_\sigma$ (within 5ms) for profile i given context $\boldsymbol{c}$.

We set $\varepsilon = 0.1$ and solve the discrete MSBP (24) with squared Euclidean cost $\boldsymbol{C}$ using (29). Fig. 5 shows that the Sinkhorn iterations converge linearly (Sec. III-E). We emphasize here that the computational complexity of proposed algorithm is minimal, thanks to the path structure of

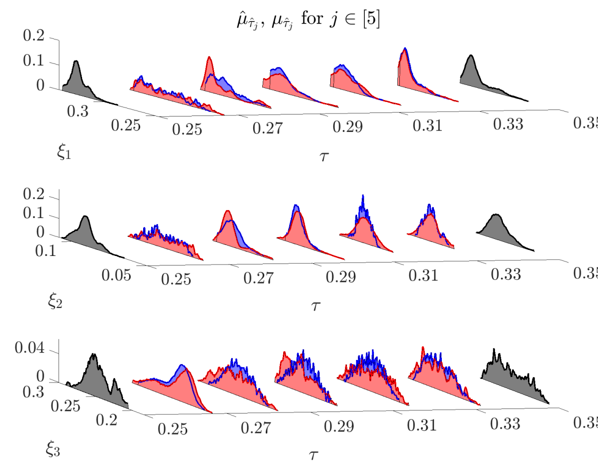

Fig. 6: Predicted $\hat{\mu}_{\hat{\tau}_j}$ (*blue*) vs. measured $\mu_{\hat{\tau}_j}$ (*red*) at times $\hat{\tau}_{j \in [\![5]\!]}$ during the 3rd control cycle with $s_{\mathrm{int}} = 4$. Distributions at the control cycle boundaries are in *black*.

the information. In particular, we solve the MSBP (24) with $n^s = 500^{26}$ decision variables in approx. 10 s in MATLAB on an Ubuntu 22.04.2 LTS Linux machine with an AMD Ryzen 7 5800X CPU.

Fig. 6 compares predicted versus observed empirical distributions. Specifically, Fig. 6 shows $s_{\mathrm{int}} + 1 = 5$ distributional predictions $\hat{\mu}_{\hat{\tau}_j}$ at times $\hat{\tau}_j$, temporally equispaced throughout the duration of the 3rd control cycle, i.e., between $\tau_{2(s_{\mathrm{int}}+1)+1}$ and $\tau_{3(s_{\mathrm{int}}+1)+1}$, with

$$\hat{\tau}_j = \tau_{2(s_{\mathrm{int}}+1)+1} + \left(\frac{\tau_{3(s_{\mathrm{int}}+1)+1} - \tau_{2(s_{\mathrm{int}}+1)+1}}{s_{\mathrm{int}} + 2}\right) j,$$

where $j \in [\![s_{\mathrm{int}} + 1]\!]$. We used (31) with $\sigma = 2(s_{\mathrm{int}} + 1) + j$, since $\hat{\tau}_j \in [\tau_{2(s_{\mathrm{int}}+1)+j}, \tau_{2(s_{\mathrm{int}}+1)+j+1}]$.

From Fig. 6 it is clear that the measure-valued predictions, while largely accurate, are prone to error in cases where the software resource usage behavior changes in bursts too short to be appear in our observations. It follows that increasing the number of snapshots should yield an improvement in overall accuracy. In this example, we achieve this by increasing s_{int}. Table II reports the Wasserstein distances $W(\cdot,\cdot)$ between the corresponding predicted and measured distributions:

$$W_j := W(\hat{\mu}_{\hat{\tau}_j}, \mu_{\hat{\tau}_j}) \quad \forall j \in [\![s_{\mathrm{int}} + 1]\!]. \tag{33}$$

We computed each of these W_j as the square root of the optimal value of the corresponding linear program that results from specializing (24) with $s = 2$, $\varepsilon = 0$.

VI. Concluding Remarks

We apply recent algorithmic advances in solving the MSBP to learn stochastic hardware resource usage by control software. The learnt model demonstrates accurate nonparametric measure-valued predictions for the joint hardware resource state at a desired time conditioned on CPS context. The formulation and its solution comes with a maximum likelihood guarantee in the space of probability measures, and the algorithm enjoys a guaranteed linear convergence rate.

References

[1] G. Bernat, A. Colin, and S. Petters, "WCET analysis of probabilistic hard real-time systems," in *23rd IEEE Real-Time Systems Symposium, 2002 (RTSS)*, 2002, pp. 279–288.

[2] M. Lv, N. Guan, Y. Zhang, Q. Deng, G. Yu, and J. Zhang, "A survey of WCET analysis of real-time operating systems," in *2009 International Conference on Embedded Software and Systems*, 2009, pp. 65–72.

[3] R. Gifford, N. Gandhi, L. T. X. Phan, and A. Haeberlen, "Dna: Dynamic resource allocation for soft real-time multicore systems," in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 196–209.

[4] K. Zhang, J. Sprinkle, and R. G. Sanfelice, "Computationally aware switching criteria for hybrid model predictive control of cyber-physical systems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 479–490, 2016.

[5] P. J. Bushell, "Hilbert's metric and positive contraction mappings in a Banach space," *Archive for Rational Mechanics and Analysis*, vol. 52, pp. 330–338, 1973.

[6] F. Elvander, I. Haasler, A. Jakobsson, and J. Karlsson, "Multi-marginal optimal transport using partial information with applications in robust localization and sensor fusion," *Signal Processing*, vol. 171, p. 107474, 2020.

[7] I. Haasler, A. Ringh, Y. Chen, and J. Karlsson, "Multimarginal optimal transport with a tree-structured cost and the Schrodinger bridge problem," *SIAM Journal on Control and Optimization*, vol. 59, no. 4, pp. 2428–2453, 2021.

[8] C. Léonard, "A survey of the Schrödinger problem and some of its connections with optimal transport," *Discrete and Continuous Dynamical Systems-Series A*, vol. 34, no. 4, pp. 1533–1574, 2014.

[9] Y. Chen, T. T. Georgiou, and M. Pavon, "Stochastic control liaisons: Richard Sinkhorn meets Gaspard Monge on a Schrodinger bridge," *Siam Review*, vol. 63, no. 2, pp. 249–313, 2021.

[10] A. Dembo and O. Zeitouni, *Large deviations techniques and applications*. Springer Science & Business Media, 2009, vol. 38.

[11] H. Follmer, "Random fields and diffusion processes," *Ecole d'Ete de Probabilites de Saint-Flour XV-XVII, 1985-87*, 1988.

[12] I. N. Sanov, *On the probability of large deviations of random variables*. United States Air Force, Office of Scientific Research, 1958.

[13] M. Pavon, G. Trigila, and E. G. Tabak, "The data-driven Schrödinger bridge," *Communications on Pure and Applied Mathematics*, vol. 74, no. 7, pp. 1545–1573, 2021.

[14] I. Csiszár, "I-divergence geometry of probability distributions and minimization problems," *The annals of probability*, pp. 146–158, 1975.

[15] J. M. Borwein, A. S. Lewis, and R. D. Nussbaum, "Entropy minimization, DAD problems, and doubly stochastic kernels," *Journal of Functional Analysis*, vol. 123, no. 2, pp. 264–307, 1994.

[16] L. Rüschendorf and L. Uckelmann, "On the n-coupling problem," *Journal of multivariate analysis*, vol. 81, no. 2, pp. 242–258, 2002.

[17] B. Pass, "Multi-marginal optimal transport: theory and applications," *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, vol. 49, no. 6, pp. 1771–1790, 2015.

[18] M. Agueh and G. Carlier, "Barycenters in the Wasserstein space," *SIAM Journal on Mathematical Analysis*, vol. 43, no. 2, pp. 904–924, 2011.

[19] Y. Brenier, "Generalized solutions and hydrostatic approximation of the Euler equations," *Physica D: Nonlinear Phenomena*, vol. 237, no. 14-17, pp. 1982–1988, 2008.

[20] J.-D. Benamou, G. Carlier, and L. Nenna, "Generalized incompressible flows, multi-marginal transport and sinkhorn algorithm," *Numerische Mathematik*, vol. 142, pp. 33–54, 2019.

[21] G. Carlier, A. Oberman, and E. Oudet, "Numerical methods for matching for teams and Wasserstein barycenters," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 49, no. 6, pp. 1621–1642, 2015.

[22] G. Buttazzo, L. De Pascale, and P. Gori-Giorgi, "Optimal-transport formulation of electronic density-functional theory," *Physical Review A*, vol. 85, no. 6, p. 062502, 2012.

[23] C. Cotar, G. Friesecke, and C. Klüppelberg, "Density functional theory and optimal transportation with Coulomb cost," *Communications on Pure and Applied Mathematics*, vol. 66, no. 4, pp. 548–599, 2013.

[24] G. Peyré and M. Cuturi, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[25] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, 2013.

[26] J. Franklin and J. Lorenz, "On the scaling of multidimensional matrices," *Linear Algebra and its applications*, vol. 114, pp. 717–735, 1989.

[27] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré, "Iterative Bregman projections for regularized transportation problems," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. A1111–A1138, 2015.

[28] H. H. Bauschke and A. S. Lewis, "Dykstras algorithm with Bregman projections: A convergence proof," *Optimization*, vol. 48, no. 4, pp. 409–427, 2000.

[29] S. D. Marino and A. Gerolin, "An optimal transport approach for the Schrödinger bridge problem and convergence of Sinkhorn algorithm," *Journal of Scientific Computing*, vol. 85, no. 2, p. 27, 2020.

[30] G. Carlier, "On the linear convergence of the multimarginal Sinkhorn algorithm," *SIAM Journal on Optimization*, vol. 32, no. 2, pp. 786–794, 2022.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[32] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[33] S. Haddad, A. Halder, and B. Singh, "Density-based stochastic reachability computation for occupancy prediction in automated driving," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 6, pp. 2406–2419, 2022.

[34] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[35] https://github.com/abhishekhalder/CPS-Frontier-Task3-Collaboration/blob/master/Codes/kbm_sim/Documentation_KinematicBicycle_Controllers.pdf, accessed: 2023-09-29.

[36] Intel Corporation, "Improving real-time performance by utilizing cache allocation technology," Apr. 2015, White Paper.

[37] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memory bandwidth management for efficient performance isolation in multi-core platforms," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 562–576, Feb 2016.

[38] "perf(1) — linux manual page," https://man7.org/linux/man-pages/man1/perf.1.html, accessed: 2023-09-29.

[39] A. W. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," *Biometrika*, vol. 71, no. 2, pp. 353–360, 1984.

[40] P. Hall, J. Marron, and B. U. Park, "Smoothed cross-validation," *Probability theory and related fields*, vol. 92, no. 1, pp. 1–20, 1992.

[41] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," 1994. [Online]. Available: https://api.semanticscholar.org/CorpusID:16010565