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.

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.
In this work, we consider an instance of (4) where physical context vectors are comprised of separable cyber and physical context vectors

\[ c := \left( \begin{array}{c} c_{\text{cyber}} \\ c_{\text{phys}} \end{array} \right). \tag{4} \]

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

\[ c_{\text{cyber}} = \begin{pmatrix} \text{allocated last-level cache} \\ \text{allocated memory bandwidth} \end{pmatrix}, \tag{5} \]

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

\[ e_{\text{phys}} = y_{\text{obs}}(x) \in \text{GP}\left([x_{\text{min}}, x_{\text{max}}]\right), \tag{6} \]

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

### B. Hardware Resource State \(\xi\)

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

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

The three elements of \(\xi\) 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 \(\xi\). To highlight this flexibility, we describe the proposed approach for \(\xi \in \mathbb{R}^d\) with suitable interpretations for the specific application.

For a time interval \([0, t]\) of interest, we think of time-varying \(\xi\) 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 \(\xi(t), 0 \leq t \leq t\), 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 \(c\), the snapshot observations comprise a sequence of joint probability measures \(\{\mu_{\sigma}\}_{\sigma \in [s]}\) satisfying \(\int d\mu_{\sigma}(\xi(\tau_{\sigma})) = 1\). In other words,

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

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 \(c\) with all parameters and initial conditions fixed. So the stochasticity in \(\xi(\tau_{\sigma})\) stems from the dynamic variability in hardware resource availability.

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

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

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

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

\[ \xi(\tau) \sim \mu_{\tau} \quad \forall \tau \in [0, t]. \tag{10} \]

Without the qualifier "most likely", the problem is over-determined 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 \( \mathcal{X}_\sigma \) := support \( \{ \mu_\sigma \} \subseteq \mathbb{R}^d \) \( \forall \sigma \in [s] \), and consider the Cartesian product \( \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_s =: \mathcal{X} \subseteq \mathbb{R}^{d \times s} \). Let \( \mathcal{M}(\mathcal{X}_\sigma) \) and \( \mathcal{M}(\mathcal{X}) \) denote the collection (i.e., manifold) of probability measures on \( \mathcal{X}_\sigma \) and \( \mathcal{X} \), respectively. Define a ground cost \( C : \mathcal{X} \to \mathbb{R}_{\geq 0} \).

Following [6, Sec. 3], let
\[
\begin{align}
\mathcal{X}_{\sigma} &:= \mathcal{X}_\sigma \times \mathcal{X}_{\sigma-1} \times \mathcal{X}_{\sigma+1} \times \ldots \times \mathcal{X}_s, \\
d\xi_{\sigma} &:= d\xi(\tau) \times \ldots \times d\xi(\tau_{\sigma-1}) \times d\xi(\tau_{\sigma+1}) \times \ldots \times d\xi(\tau_s).
\end{align}
\]

For \( \varepsilon \geq 0 \) (not necessarily small), the multimarginal Schrödinger bridge problem (MSBP) is the following infinite dimensional convex program:
\[
\min_{M \in \mathcal{M}(\mathcal{X})} \int_{\mathcal{X}} \left\{ C(\xi(\tau_1), \ldots, \xi(\tau_s)) + \varepsilon \log M(\xi(\tau_1), \ldots, \xi(\tau_s)) \right\} d\xi(\tau_1) \ldots d\xi(\tau_s) \\
\text{subject to} \\
\quad M(\xi(\tau_1), \ldots, \xi(\tau_s)) d\xi_{\sigma} = \mu_\sigma \forall \sigma \in [s].
\]

In particular, \( \mathcal{M}(\mathcal{X}) \) is a convex set. The objective (12a) is strictly convex in \( M \), thanks to the \( \varepsilon \)-regularized negative entropy term \( \int_{\mathcal{X}} \varepsilon M \log M \). 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
\[
C(\xi(\tau_1), \ldots, \xi(\tau_s)) = \sum_{\sigma=1}^{s-1} C_\sigma(\xi(\tau_\sigma), \xi(\tau_{\sigma+1})).
\]

In particular, we choose the squared Euclidean distance sequential cost between two consecutive snapshot indices, i.e., \( C_\sigma(\cdot, \cdot) := \| \cdot - \|_2^2 \) \( \forall \sigma \in [s] \). 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 \( \mu_1, \mu_2 \). 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 \( C([\tau_1, \tau_2], \mathbb{R}^d) \) 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 \( C([\tau_1, \tau_2], \mathbb{R}^d) \) with time \( \tau_1 \) marginal \( \mu_1 \), and time \( \tau_2 \) marginal \( \mu_2 \). Given a symmetric ground cost (e.g., Euclidean distance) \( C : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathbb{R}_{\geq 0} \), let
\[
K(\cdot, \cdot) := \exp \left( -\frac{C(\cdot, \cdot)}{\varepsilon} \right),
\]
and consider the bimarginal Gibbs kernel
\[
K(\xi(\tau_1), \xi(\tau_2)) \mu_1 \otimes \mu_2.
\]

Then, the bimarginal SBP solves
\[
\min_{\pi \in \Pi(\mu_1, \mu_2)} \varepsilon D_{KL}(\pi \| K(\xi(\tau_1), \xi(\tau_2))) \mu_1 \otimes \mu_2.
\]

i.e., the most likely evolution of the path measure consistent with the observed measure-valued snapshots \( \mu_1, \mu_2 \).

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 \( \pi \mapsto D_{KL}(\pi \| \nu) \) for fixed \( \nu \).

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 \( C : \mathcal{X} \to \mathbb{R}_{\geq 0} \) as in (12)-(13), we generalize (14) as
\[
K(\xi(\tau_1), \ldots, \xi(\tau_s)) := \exp \left( -\frac{C(\xi(\tau_1), \ldots, \xi(\tau_s))}{\varepsilon} \right),
\]
and define the multimarginal Gibbs kernel
\[
K(\xi(\tau_1), \ldots, \xi(\tau_s)) \mu_1 \otimes \ldots \otimes \mu_s.
\]

Problem (16) then generalizes to
\[
\min_{\pi \in \Pi(\mu_1, \ldots, \mu_s)} \varepsilon D_{KL}(\pi \| K(\xi(\tau_1), \ldots, \xi(\tau_s))) \mu_1 \otimes \ldots \otimes \mu_s
\]
where \( \Pi(\mu_1, \ldots, \mu_s) \) denotes the collection of all path measures on \( C([\tau_1, \tau_s], \mathbb{R}^d) \) with time \( \tau_\sigma \) 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 \( \mu_\sigma \forall \sigma \in [s] \).

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

The minimizer of (12), \( M_{\text{opt}}(\xi(\tau_1), \ldots, \xi(\tau_s)) \) can be used to compute the optimal coupling between snapshot index pairs \( (\sigma_1, \sigma_2) \in [s]^2 \setminus (1 < \sigma_2) \) as
\[
\int_{\mathcal{X}_{-\sigma_1, -\sigma_2}} M_{\text{opt}}(\xi(\tau_1), \ldots, \xi(\tau_s)) d\xi_{-\sigma_1, -\sigma_2}
\]
where
\[
d\xi_{-\sigma_1, -\sigma_2} := \prod_{\sigma \in [s] \setminus \{\sigma_1, \sigma_2\}} d\xi(\tau_\sigma),
\]
\[
\mathcal{X}_{-\sigma_1, -\sigma_2} := \prod_{\sigma \in [s] \setminus \{\sigma_1, \sigma_2\}} \mathcal{X}_\sigma.
\]

This will be useful for predicting the statistics of \( \xi(\tau) \sim \mu_\tau \) 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 \( \{\xi(\tau_\sigma)\}_{\sigma=1}^n \) 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 \( C \in \mathbb{R}^{n_s}_{\geq 0} \) with components \( [C_{i_1,\ldots,i_s}] = C(\xi_{i_1},\ldots,\xi_{i_s}) \). The component \( [C_{i_1,\ldots,i_s}] \) encodes the cost of transporting unit mass for a tuple \((i_1,\ldots,i_s)\).

Likewise, consider the discrete mass tensor \( M \in \mathbb{R}^{n_s}_{\geq 0} \) with components \( [M_{i_1,\ldots,i_s}] = M(\xi_{i_1},\ldots,\xi_{i_s}) \). The component \( [M_{i_1,\ldots,i_s}] \) denotes the amount of transported mass for a tuple \((i_1,\ldots,i_s)\).

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

\[
\text{proj}_\sigma(M)_{i,i_1}\ldots,i_{\sigma-1},i_{\sigma+1},\ldots,i_s = \sum_{i_1,\ldots,i_{\sigma-1},i_{\sigma+1},\ldots,i_s} M_{i_1,\ldots,i_{\sigma-1},i,\ldots,i_{\sigma+1},\ldots,i_s}.
\]

Likewise, denote the projection of \( M \in \mathbb{R}^{n_s}_{\geq 0} \) on the \((\sigma_1,\sigma_2)\)th marginal as \( \text{proj}_{\sigma_1,\sigma_2}(M) \), i.e., \( \text{proj}_{\sigma_1,\sigma_2} : (\mathbb{R}^{n_s}_{\geq 0} \mapsto \mathbb{R}^{n_{\sigma_1} \times n_{\sigma_2}}_{\geq 0} \), and is given componentwise as

\[
\text{proj}_{\sigma_1,\sigma_2}(M)_{i,j} = \sum_{i_1,\ldots,i_{\sigma_1-1},i_{\sigma_1},i_{\sigma_2+1},\ldots,i_{\sigma_2},\ldots,i_s} M_{i_1,\ldots,i_{\sigma_1-1},i,\ldots,i_{\sigma_1},i_{\sigma_2+1},\ldots,i_{\sigma_2},\ldots,i_s}.
\]

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

\[
\min_{M \in \mathbb{R}^{n,s}_{\geq 0}} \langle C + \varepsilon \log M, M \rangle \quad \text{subject to} \quad \text{proj}_\sigma(M) = \mu_\sigma \quad \forall \sigma \in [s].
\]

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 \( M_{\text{opt}} \) is a diagonal scaling of \( K := \exp(-(C/\varepsilon)) \in \mathbb{R}^{n \times n} \), i.e., \( M_{\text{opt}} = \text{diag}(u_1)K\text{diag}(u_2) \) where \( u_1 := \exp(\lambda_1/\varepsilon) \), \( u_2 := \exp(\lambda_2/\varepsilon) \), and \( \lambda_1,\lambda_2 \in \mathbb{R}^n \) are the Lagrange multipliers associated with respective bimarginal constraints \( \text{proj}_1(M) = \mu_1, \text{proj}_2(M) = \mu_2 \). The unknowns \( u_1, u_2 \) can be obtained by performing the Sinkhorn iterations

\[
u_1 \leftarrow \mu_1 \odot (Ku_2), \quad (25a)\\
\nu_2 \leftarrow \mu_2 \odot (K^\top u_1), \quad (25b)
\]

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 \( M_{\text{opt}} = K \odot U \) where \( K := \exp(-(C/\varepsilon)) \in \mathbb{R}^{n_s \times n_s}_{\geq 0} \), \( U := \otimes_{\sigma=1}^s u_\sigma \in \mathbb{R}^{n_s}_{\geq 0} \), \( u_\sigma := \exp(\lambda_\sigma/\varepsilon) \), and \( \lambda_\sigma \in \mathbb{R}^n \) are the Lagrange multipliers associated with the respective multimarginal constraints (24b). The unknowns \( u_\sigma \) can, in principle, be obtained from the multimarginal Sinkhorn iterations [27]

\[
\nu_\sigma \leftarrow u_\sigma \odot \mu_\sigma \odot \text{proj}_\sigma(K \odot U) \quad \forall \sigma \in [s],
\]

which generalizes (25). However, computing \( \text{proj}_\sigma(K \odot U) \) requires \( O(n_s^s) \) 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 C

We circumvent the exponential complexity in computing \( \text{proj}_\sigma(K \odot U) \) 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 \( C \in \mathbb{R}^{n_s}_{\geq 0} \) induced by a path structured cost (13) so that \( \{\xi(\tau_\sigma)\}_{i=1}^n \) to the destination set \( \{\xi(\tau_\sigma)\}_{i=1}^n \). Let \( K := \exp((C - \sigma_1,\sigma_1 + 1)/\varepsilon) \in \mathbb{R}^{n \times n}_{\geq 0} \), \( K := \exp(-(C - \sigma_1,\sigma_1 + 1)/\varepsilon) \in \mathbb{R}^{n \times n}_{\geq 0} \). Then (22) and (23) can be expressed as

\[
\text{proj}_\sigma(K \odot U) = \left( \left( \sum_{j=1}^{s-1} K^{j-1 \rightarrow j} \text{diag}(u_j) \right) K^{s-1 \rightarrow s} \right)^\top u_\sigma \quad (27)
\]

and

\[
\text{proj}_{\sigma_1,\sigma_2}(K \odot U) = \left( \prod_{j=\sigma_1+1}^{s-1} K^{j-1 \rightarrow j} \text{diag}(u_j) \right) K^{s-1 \rightarrow s} u_\sigma \quad \forall(\sigma_1,\sigma_2) \in \{[s] \odot 2 \mid \sigma_1 < \sigma_2 \}. (28)
\]
Proof. The proof strategy is to write the Hilbert-Schmidt inner product \( \langle K, U \rangle \) in two different ways.

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

\[
\langle K, U \rangle = \sum_{i_1, \ldots, i_s} \left( \prod_{j=2}^{s} \left[ K_{i_{j-1}, i_j}^{j-1} \right] \right) \prod_{j=1}^{s} (u_{j})_{i_j}
\]

\[
= \sum_{i_1, \ldots, i_s} (u_{1})_{i_1} \prod_{j=2}^{s} \left[ K_{i_{j-1}, i_j}^{j-1} \text{diag}(u_{j}) \right]_{i_{j-1}, i_j}
\]

\[
= u_{1}^T \left( \prod_{j=2}^{s-1} K_{i_{j-1}, i_j}^{j-1} \text{diag}(u_{j}) \right) K^{s-1 \rightarrow s} u_{s},
\]

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

Next, notice that we can alternatively write

\[
\langle K, U \rangle = u_{1}^T \left( \prod_{j=2}^{s-1} K_{i_{j-1}, i_j}^{j-1} \text{diag}(u_{j}) \right) K^{s-1 \rightarrow s} u_{s},
\]

(29)

and (27) follows from [6, Lemma 2 in Appendix 1].

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

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

\[
\mu_{\tau} = u_{1}^T \left( \prod_{j=2}^{s-1} K_{i_{j-1}, i_j}^{j-1} \text{diag}(u_{j}) \right) K^{s-1 \rightarrow s} u_{s}, \quad \forall \sigma \in [s].
\]

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

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

Remark 5. (Linear complexity in \( d \)) The dimension \( d \) of the vector \( \xi \) only affects the construction of the time-varying Euclidean distance matrices \( C_{\sigma \rightarrow \sigma + 1} \) \( \forall \sigma \in [s-1] \) 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 \( M_{\text{opt}} = K \otimes U \) 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 \( M_{\text{opt}} = K \otimes U \) of (24) into binormal transport plans

\[
M_{\sigma \rightarrow \sigma + 2} := \text{proj}_{\sigma_1, \sigma_2}(M_{\text{opt}}) = \text{proj}_{\sigma_1, \sigma_2}(K \otimes U).
\]

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

\[
\hat{\mu}_{\tau} := \sum_{i=1}^{n} \sum_{j=1}^{n} [M_{i,j}^{\sigma \rightarrow \sigma + 1}] \delta(\xi - \hat{\xi}(\tau, \xi^{(\sigma)}(\tau), \xi^{(\sigma + 1)}(\tau)))
\]

(31)

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

\[
\hat{\xi}(\tau, \xi^{(\sigma)}(\tau), \xi^{(\sigma + 1)}(\tau)) = (1 - \lambda)\xi^{(\sigma)}(\tau) + \lambda\xi^{(\sigma + 1)}(\tau), \quad \lambda := \frac{\tau - \tau_{\sigma}}{\tau_{\sigma + 1} - \tau_{\sigma}} \in [0, 1].
\]

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) \( \{c_i\}_{i=1}^{n} \), execute the control software over \( [0, t] \) to generate hardware resource state sample snapshots (Sec. III-B) \( \{\xi^{(\tau)}(\sigma)\}_{\tau=1}^{n} \), and thereby empirical \( \mu_{\sigma} \) as in (9) for all \( \sigma \in [s] \), conditional on each of the \( n_{\text{context}} \) context samples.

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

Step 3. Given a query context \( c \) and time \( \tau \in [0, t] \), return the 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 \( (x, y, v, \psi) \) and two control inputs \( (a_c, \delta) \), given by \( \dot{x} = v \cos(\psi + \beta), \dot{y} = v \sin(\psi + \beta), \dot{v} = a_c, \dot{\psi} = \frac{a_c}{v} \sin(\beta) \), where the sideslip angle \( \beta =

---

1Git repo: https://github.com/abhinavkhalder/CP5-Frontiers-Task3-Collaboration
partitions and memory bandwidth available to the control
CAT [36] and Memguard [37] to control allocation of LLC
16.04.7 Linux machine with an Intel Xeon E5-2683 v4
profiles for our NMPC control software, we used an Ubuntu
execution profiles for our specific case study.

16.04.7 Linux machine with an Intel Xeon E5-2683 v4
profiles for our specific case study.

We next provide details on generating control software
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 ξ 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_{des}(x)\}_{i=1}^{12}, and 5 unique vectors of \{c_{cyber}^i\}_{i=1}^{5}, comprising
n_{context} = 12 \times 5 = 60 samples for c. Conditional on
each of these 60 context samples \{c^i\}_{i=1}^{5}, we runler software for 500 profiles for each unique c for a total of
30,000 profiles.

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

Our vectorial samples \{c_{cyber}^i\}_{i=1}^{5} in (5) were \{1, 1\}^T,
\{5, 5\}^T, \{10, 10\}^T, \{15, 15\}^T, and \{20, 20\}^T, 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.

![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.](image)

number of intracycle marginals \(s_{int}\) vs. Wasserstein
distances \(W_i\) as in (33). All entries are scaled up by 10^8.

**Applying the Proposed Algorithm.** Given a query context
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 \(c_{cyber} = \{15, 15\}^T\) and closest
\(c_{phys} = y_{des}(x)\). Profiling data for this c is shown in Fig. 3.

For each of the \(n = 500\) profiles, we are given the
time ends 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 ξ 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

![TABLE I: The means and standard deviations of the end times for the \(n_c = 5\) control cycles data shown in Fig. 4.](image)

![TABLE II: Number of intracycle marginals \(s_{int}\) vs. Wasserstein
distances \(W_i\) as in (33). All entries are scaled up by 10^8.](image)
Fig. 3: Components of the measured feature vector $\xi$ in (7) for all of the five control cycles for 500 executions of the NMPC software, where $c = [15, 15, y_{\text{int}}(x)]$.

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 $c$ shown above. The KDEs used Gaussian kernel with bandwidths computed via cross validation [39, 40].

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

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

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

We set $\varepsilon = 0.1$ and solve the discrete MSBP (24) with squared Euclidean cost $C$ 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 the information. In particular, we solve the MSBP (24) with $n^s = 500$ 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 $s_{\text{int}} + 1 = 5$ distributional predictions $\hat{\mu}_{\hat{\tau}_j^i}$ at times $\hat{\tau}_j^i$, temporally equispaced throughout the duration of the 3rd control cycle, i.e., between $\tau_2(s_{\text{int}} + 1) + 1$ and $\tau_3(s_{\text{int}} + 1) + 1$, with

$$\hat{\tau}_j = \tau_2(s_{\text{int}} + 1) + 1 + \left( \tau_3(s_{\text{int}} + 1) + 1 - \tau_2(s_{\text{int}} + 1) + 1 \right) j,\]$$

where $j \in [s_{\text{int}} + 1]$. We used (31) with $\sigma = 2(s_{\text{int}} + 1) + j$, since $\hat{\tau}_j \in [\tau_2(s_{\text{int}} + 1) + j, \tau_3(s_{\text{int}} + 1) + 1]$. 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 $s_{\text{int}}$.

Table II reports the Wasserstein distances $W(\cdot, \cdot)$ between the corresponding predicted and measured distributions:

$$W_j := W(\hat{\mu}_{\hat{\tau}_j^i}, \mu_{\tau_j}) \quad \forall j \in [s_{\text{int}} + 1].$$

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 $s = 2$, $\varepsilon = 0$. 

[Images and diagrams as described in the text]
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


