SHAPLEYOG: FAST ADAPTIVE SHAPLEY VALUE ESTIMATION WITH ODD GAUSSIAN PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Shapley values are a central tool for attribution in machine learning. Since their exact computation requires evaluating the *cooperative game* to be explained on exponentially many *player coalitions*, they are typically estimated from a limited number of evaluations. Recently, `ShaplEIG` has shown that *adaptively* selecting the coalitions to evaluate substantially improves sample efficiency over classical estimators. This, however, comes at the cost of a sequential procedure that, in every iteration, refits a Gaussian process surrogate of the game and selects the coalition with the largest expected information gain about the Shapley values. The resulting computational overhead limits its applicability to *costly* games with low evaluation budgets. We present fast adaptive Shapley value estimation with Odd Gaussian Processes (`ShapleyOG`) to overcome this limitation. `ShapleyOG` leverages the fact that Shapley values depend only on the odd functional component of the game, and we show that under paired sampling a Gaussian process surrogate on this component alone yields the same Shapley value posterior and coalition selection criterion from half as many observations. Moreover, `ShapleyOG` refits the surrogate only on a sparse schedule and efficiently updates the selection criterion in between, and it extends quadrature-based computation schemes to the prior covariance of the Shapley values, which allows for exact or cheaper approximate computation. In extensive experiments on diverse games, `ShapleyOG` matches the estimation accuracy of `ShaplEIG` while reducing its overhead by up to three orders of magnitude. As a result, it outperforms state-of-the-art non-adaptive estimators at equal total computation time already when a single evaluation takes a tenth of a second and a budget of a few thousand evaluations suffices, thereby widening the regime in which adaptive coalition selection is applicable and pays off.

1 INTRODUCTION

Shapley values (SVs; Shapley, 1953), a principled concept from cooperative game theory for fairly distributing the jointly achieved *payoff* of a *game* among its *players*, have become a central tool for attribution in machine learning (ML). SVs are used to attribute predictions or predictive performance to features, training data points, or hyperparameters (Lundberg & Lee, 2017; Ghorbani & Zou, 2019; Wever et al., 2026), often in critical domains such as healthcare, law, or finance (Lundberg et al., 2020; Slack et al., 2021). Computing the SVs exactly requires evaluating the game's *value function* on all 2^p *coalitions* of the p players, and each evaluation may itself incur a non-negligible cost. Exact computation is therefore only feasible for small games with cheap value functions, and SVs are typically estimated (Chen et al., 2023). Accurate estimates, however, are essential, in particular in the critical domains mentioned above, and numerous estimators have been proposed, the vast majority of which follow the same two-stage principle: 1) evaluate the value function on a subset of coalitions and 2) estimate the SVs from these evaluations, either via Monte Carlo methods (Castro et al., 2009; Kolpaczki et al., 2024) or via a surrogate model of the value function (Lundberg & Lee, 2017; Fumagalli et al., 2026b). While both stages have seen substantial progress in recent years, the choice of coalitions is, with few exceptions (Slack et al., 2021; Burgess & Chapman, 2021; Mitchell et al., 2022; Nguyen et al., 2025; Rundel et al., 2026), *non-adaptive*: it follows a scheme that is fixed a priori and does not adapt to the observed values. Moreover, it is *symmetric* in the players, so that coalitions are distinguished at most by their size and never by the specific players involved (Covert

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

& Lee, 2021; Musco & Witter, 2025). Recently, `ShaplEIG` (Rundel et al., 2026) demonstrated that selecting coalitions *adaptively*, i.e., based on previous evaluations, improves the sample efficiency of SV estimation substantially. `ShaplEIG` is a Bayesian experimental design (BED; Chaloner & Verdinelli, 1995; Rainforth et al., 2024) approach: it models the value function with a Gaussian process (GP; Williams & Rasmussen, 1995; Rasmussen & Williams, 2006) surrogate and, in each iteration, evaluates the candidate coalition with the largest expected information gain (EIG; Lindley, 1956) about the SVs, i.e., the coalition expected to reduce the posterior uncertainty about the SVs the most. This adaptivity comes at a price: in every iteration, `ShaplEIG` refits the GP hyperparameters and evaluates the EIG over a candidate set, an overhead that grows disproportionately with the number of players and the evaluation budget. As a result, `ShaplEIG` is only applicable to settings with *costly* value functions, in which the number of evaluations is severely limited in the first place. Two recent developments point towards a substantially faster adaptive approach. First, SVs depend only on the *odd* component of the value function, which OddSHAP (Fumagalli et al., 2026a) exploits in polynomial regression by fitting only odd basis functions to odd observations, affording a richer surrogate while discarding no information about the SVs. Second, the integral representation of the Shapley weights (Owen, 1972) enables the exact computation of SVs for product-structured games, such as those arising from the leaves of tree ensembles, via quadrature (Li et al., 2026; Wettenstein et al., 2026; Mohammadi et al., 2026), which, unlike the coefficient-based schemes of `ShaplEIG`, is embarrassingly parallel and can trade accuracy for cost by using fewer nodes. In this work, we combine both ideas into fast adaptive Shapley value estimation with Odd Gaussian Processes (`ShapleyOG`), which retains the sample efficiency of `ShaplEIG` while reducing its computational overhead by a factor of 388 to 1250 in our experiments, and thus substantially extends the applicability of adaptive estimation to larger evaluation budgets and less costly value functions. Our contributions are as follows:

- We exploit that the Hamming kernel GP decomposes into independent even and odd components (Klus et al., 2021; Ginsbourger et al., 2016b) and show that, under paired sampling (Fumagalli et al., 2026a; Covert & Lee, 2021), the SV posterior of the resulting *odd GP* coincides with that of the full GP, and that under *paired acquisition* the EIG about the SVs equals the EIG of the odd target under the odd GP. Both are thus obtained from training and candidate sets of half the size.
- We extend quadrature-based computation schemes to the terms required for SV posterior inference and coalition selection; in particular, we express the prior covariance of the SVs under the Hamming GP as a bivariate integral, which Gauss–Legendre quadrature (Golub & Welsch, 1969) evaluates exactly, or approximately with fewer nodes, enabling even larger games.
- `ShapleyOG` refits the GP only on a sparse schedule and, in between, recycles or efficiently updates essentially all quantities required for the EIG maximization instead of recomputing them.
- On a diverse benchmark, `ShapleyOG` attains, on average, a lower estimation error than state-of-the-art non-adaptive estimators (Musco & Witter, 2025; Witter et al., 2025; Fumagalli et al., 2026a) at equal total computation time, i.e., including value function evaluations and estimation overhead, already when an evaluation takes a tenth of a second, e.g., one forward pass of a model, and a budget of several thousand evaluations suffices.

2 Background

2.1 Shapley Values

A cooperative game on a set $P := \{1, \dots, p\}$ of p players is defined by a value function $\nu : 2^P \to \mathbb{R}$ that assigns a real-valued payoff to each coalition $S \subseteq P$. SVs aggregate the value function into a single *attribution* score for each player $i \in P$,

$$\phi_i(\nu) := \sum_{S \subseteq P \setminus \{i\}} \frac{1}{p \cdot \binom{p-1}{|S|}} \Big(\nu(S \cup \{i\}) - \nu(S) \Big) = \mathbf{A}_i \, \boldsymbol{\nu}, \tag{1}$$

i.e., a weighted average of the marginal contributions $\nu(S \cup \{i\}) - \nu(S)$ of player i over all coalitions S not containing i, in which every coalition size receives the same total weight. Here, $\boldsymbol{\nu} := (\nu(S))_{S \subseteq P} \in \mathbb{R}^{2^p}$ collects the values of all coalitions, and $\mathbf{A} \in \mathbb{R}^{p \times 2^p}$ with rows $\mathbf{A}_i$ is the linear *Shapley transformation*, so that $\boldsymbol{\phi}(\nu) := (\phi_1(\nu), \dots, \phi_p(\nu))^\top = \mathbf{A}\boldsymbol{\nu}$. Explicitly, $\mathbf{A}_{i,S} = 1/\big(p\binom{p-1}{|S|-1}\big)$ if $i \in S$ and $\mathbf{A}_{i,S} = -1/\big(p\binom{p-1}{|S|}\big)$ otherwise. By slight abuse of notation, we

define ν equivalently on binary indicator vectors $\mathbf{z} \in \{0,1\}^p$ via their bijective correspondence with coalitions $S \subseteq P$, denote the indicator vector of a coalition S by $\mathbf{1}_S \in \{0,1\}^p$, and write $\bar{\mathbf{z}} := \mathbf{1} - \mathbf{z}$ for the complement coalition. We collect all 2^p coalitions as rows of a matrix $\mathbf{Z} \in \{0,1\}^{2^p \times p}$, index its rows and the entries of $\boldsymbol{\nu} = \nu(\mathbf{Z})$ by the coalitions themselves, and let $\mathbf{e}_{\mathbf{z}} \in \mathbb{R}^{2^p}$ denote the standard basis vector of $\mathbf{z}$, so that $\nu(\mathbf{z}) = \mathbf{e}_{\mathbf{z}}^\top \boldsymbol{\nu}$. Finally, the t coalition-value pairs observed up to iteration t are denoted by $\mathcal{D}_t := \{(\mathbf{z}^{(i)}, \nu(\mathbf{z}^{(i)}))\}_{i=1}^t$, with the coalitions collected in a matrix $\mathbf{X}_t \in \{0,1\}^{t \times p}$ and the values stacked in a vector $\mathbf{Y}_t \in \mathbb{R}^t$.

2.2 Surrogate-Based Shapley Value Estimation

Given an evaluation budget T, *surrogate-based estimators* fit a surrogate $\hat{\nu}$ to the T observed evaluations and return its SVs, $\hat{\boldsymbol{\phi}} := \boldsymbol{\phi}(\hat{\nu})$, which must be computable efficiently: this is immediate for the linear surrogates of Kernel SHAP (Lundberg & Lee, 2017) and Leverage SHAP (Musco & Witter, 2025), whose SVs are their coefficients, tractable for polynomial surrogates (Fumagalli et al., 2026b;a), and requires dedicated schemes for more expressive surrogates such as the tree-ensemble proxy of Regression MSR (Witter et al., 2025) or the GP of `ShaplEIG`. Which coalitions are evaluated is often tied to the estimator, e.g., *leverage score sampling* (Musco & Witter, 2025) draws coalitions according to their leverage in the regression problem of Leverage SHAP; *paired sampling* (Covert & Lee, 2021), which evaluates every sampled coalition $\mathbf{z}$ together with its complement $\bar{\mathbf{z}}$, and the *border trick* (Lundberg & Lee, 2017; Fumagalli et al., 2023; Musco & Witter, 2025), which favours very small and very large coalitions with the largest Shapley weights, are shared across estimators. All of these schemes are fixed before any value is observed and, although individual draws may depend on previous ones, symmetric in the players, i.e., invariant under relabeling them, so that coalitions are distinguished at most by their size.

2.3 Adaptive Estimation with ShaplEIG

We now review `ShaplEIG`, the adaptive estimator that `ShapleyOG` builds on.

Hamming Kernel GP Surrogate. `ShaplEIG` places a GP prior with a Hamming kernel (Platt et al., 2001; Qian et al., 2008; Hutter, 2009) $k_\xi : \{0,1\}^p \times \{0,1\}^p \to \mathbb{R}_{>0}$ as covariance function on the value function. This product kernel quantifies the similarity of two coalitions as

$$k_\xi(\mathbf{z}, \mathbf{z}') = \sigma_\nu^2 \prod_{j:\, z_j \neq z'_j} \beta_j, \qquad \beta_j := \exp(-1/\ell_j), \tag{2}$$

so that every player on whose membership the coalitions disagree reduces the covariance by a factor $\beta_j < 1$ (see Appendix A.1). The player-specific *lengthscales* $\ell_j > 0$ and the output scale σ_ν^2 form the hyperparameters ξ, which are learned from the observed data via maximum a posteriori (MAP) estimation (Rasmussen & Williams, 2006). In contrast to the point-estimate surrogates of Section 2.2, the GP is a *probabilistic* surrogate: updating the prior with the data $\mathcal{D}_t$ yields the full posterior over the value function vector, $\boldsymbol{\nu} \mid \mathcal{D}_t \sim \mathcal{N}_{2^p}(\boldsymbol{\mu}_{\boldsymbol{\nu}|\mathcal{D}_t}, \boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t})$, whose mean $\boldsymbol{\mu}_{\boldsymbol{\nu}|\mathcal{D}_t}$ takes the role of the surrogate $\hat{\nu}$ and whose covariance quantifies the remaining uncertainty. Notably, since the SVs are a linear transformation of $\boldsymbol{\nu}$ (Equation (1)), their posterior is available in closed form as a multivariate Gaussian as well (Chau et al., 2023),

$$\boldsymbol{\phi} \mid \mathcal{D}_t \sim \mathcal{N}_p(\mathbf{A}\boldsymbol{\mu}_{\boldsymbol{\nu}|\mathcal{D}_t}, \mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{A}^\top) = \mathcal{N}_p(\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_t}, \boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t}). \tag{3}$$

Denoting by $\mathbf{K}_\xi(\cdot,\cdot)$ the kernel matrix between two sets of coalitions, e.g., $\mathbf{K}_\xi(\mathbf{Z}, \mathbf{X}_t) \in \mathbb{R}^{2^p \times t}$ with entries $k_\xi(\mathbf{z}, \mathbf{z}')$ for all coalitions $\mathbf{z}$ in $\mathbf{Z}$ and $\mathbf{z}'$ in $\mathbf{X}_t$, the posterior mean and covariance for a zero-mean prior are given by

$$\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_t} = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{X}_t)\big[\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I}\big]^{-1}\mathbf{Y}_t, \tag{4}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t} = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top - \mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{X}_t)\big[\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I}\big]^{-1}\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{Z})\mathbf{A}^\top. \tag{5}$$

The posterior mean serves as the SV estimate, $\hat{\boldsymbol{\phi}} := \boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_t} = \boldsymbol{\phi}(\boldsymbol{\mu}_{\boldsymbol{\nu}|\mathcal{D}_t})$, which is consistent for a noiseless GP (Rundel et al., 2026). Its computation involves the prior cross-covariance $\mathrm{Cov}(\boldsymbol{\phi}, \nu(\mathbf{X}_t)) = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{X}_t) \in \mathbb{R}^{p \times t}$, whose i-th column holds the SVs of the *kernel game* $k_\xi(\cdot, \mathbf{z}^{(i)})$, and the training kernel matrix $\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t) \in \mathbb{R}^{t \times t}$. Instead of enumerating all 2^p coalitions, the former can be computed for the Hamming kernel in $\mathcal{O}(p^2)$ per column via elementary

symmetric polynomials (ESPs; van Es & Helmers, 1988; Macdonald, 1998; Charalambides, 2018), as shown by Mohammadi et al. (2025), and the prior covariance $\mathrm{Cov}(\boldsymbol{\phi}, \boldsymbol{\phi}) = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$ in Equation (5) in $\mathcal{O}(p^4)$ (Rundel et al., 2026); with the $\mathcal{O}(t^3)$ Cholesky decomposition of the training kernel matrix and the triangular solves, the SV posterior costs $\mathcal{O}(t^3 + t^2p + p^4)$ for $t \geq p$ (Appendix A.2). In Equations (4) and (5), σ_ϵ^2 is a small constant added for numerical stability, equivalent to observation noise (Appendix A.1).

EIG-based Coalition Selection. Given an initial design of coalition-value pairs, `ShaplEIG` at each iteration t selects the next coalition $\mathbf{z}^{(t+1)}$ to evaluate by maximizing the EIG about the SVs over a candidate set $\mathcal{C} \subseteq \{0,1\}^p$. This is the expected reduction in the posterior entropy H of the SVs due to evaluating the candidate $\mathbf{z}$, or equivalently the mutual information (MI) between the SVs and the candidate value (Lindley, 1956; Cover & Thomas, 2006). Under the GP surrogate, it admits a closed form (Rundel et al., 2026):

$$\mathrm{EIG}_{\boldsymbol{\phi}}^{(t)}(\mathbf{z}) := H\big(\boldsymbol{\phi} \mid \mathcal{D}_t\big) - \mathbb{E}_{p(\nu(\mathbf{z})\mid\mathcal{D}_t)}\Big[H\big(\boldsymbol{\phi} \mid \nu(\mathbf{z}), \mathcal{D}_t\big)\Big] = I\big(\boldsymbol{\phi}; \nu(\mathbf{z}) \mid \mathcal{D}_t\big) \tag{6}$$

$$\propto \log \mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}_t) - \log\big(\mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}_t) - Q_t(\mathbf{z})\big), \tag{7}$$

where $\mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}_t)$ is the posterior predictive variance of $\nu(\mathbf{z})$ and $Q_t(\mathbf{z})$ is defined as

$$Q_t(\mathbf{z}) := \big(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{e}_\mathbf{z}\big)^\top \big(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{A}^\top\big)^{-1}\big(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{e}_\mathbf{z}\big). \tag{8}$$

Evaluating the EIG over a candidate set $\mathcal{C}$ additionally requires $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathcal{C})$ in $\mathcal{O}(|\mathcal{C}|p^2)$ and triangular solves for all candidates in $\mathcal{O}(|\mathcal{C}|t^2)$, i.e., $\mathcal{O}(t^3 + p^4 + |\mathcal{C}|t^2)$ in total for $t \geq p$ (Rundel et al., 2026). The EIG depends on the observed values only through the hyperparameters ξ. To remain adaptive to previous evaluations, `ShaplEIG` therefore refits ξ in every iteration, which requires repeated $\mathcal{O}(t^3)$ decompositions of the training kernel matrix and solutions of the associated triangular systems. Together with the EIG evaluation over potentially large candidate sets, this overhead accumulates over the iterations and, being polynomial in t and p, grows disproportionately for large evaluation budgets and numbers of players, so that the estimation can take hours or even days (Rundel et al., 2026).

3 SHAPLEYOG

We transfer two recent developments to the Hamming kernel GP surrogate of `ShaplEIG`, the restriction to the odd component (Section 3.1) and quadrature-based computation for product games (Section 3.2), and combine them with a sparse refit schedule and efficient EIG updates into the `ShapleyOG` algorithm (Section 3.3).

3.1 RESTRICTING THE SURROGATE TO THE ODD COMPONENT

Our starting point is that every (value) function decomposes uniquely into an *odd* and an *even* component, $\nu = \nu_{\mathrm{odd}} + \nu_{\mathrm{even}}$, where for all $\mathbf{z} \in \{0,1\}^p$

$$\nu_{\mathrm{odd}}(\mathbf{z}) := \tfrac{1}{2}\big(\nu(\mathbf{z}) - \nu(\bar{\mathbf{z}})\big) = -\nu_{\mathrm{odd}}(\bar{\mathbf{z}}) \tag{9}$$

and $\nu_{\mathrm{even}} := \nu - \nu_{\mathrm{odd}}$ satisfies $\nu_{\mathrm{even}}(\mathbf{z}) = \nu_{\mathrm{even}}(\bar{\mathbf{z}})$. Moreover, the SVs depend on the odd component only, i.e., $\boldsymbol{\phi}(\nu) = \boldsymbol{\phi}(\nu_{\mathrm{odd}})$, since the SVs of any even function vanish (Fumagalli et al., 2026a). Surrogate-based estimators can exploit this by isolating the odd component and thereby avoid spending resources on the even component, which is irrelevant for the SVs. This requires a surrogate $\hat{\nu}_{\mathrm{odd}}$ that is odd by construction, i.e., $\hat{\nu}_{\mathrm{odd}}(\mathbf{z}) = -\hat{\nu}_{\mathrm{odd}}(\bar{\mathbf{z}})$ for all $\mathbf{z} \in \{0,1\}^p$, and fitting it to the odd component only. The latter is achieved by paired sampling, since a pair $(\nu(\mathbf{z}), \nu(\bar{\mathbf{z}}))$ determines the odd target $\nu_{\mathrm{odd}}(\mathbf{z})$ exactly, whereas the former is less straightforward. OddSHAP (Fumagalli et al., 2026a) obtains an odd polynomial surrogate by restricting the Fourier basis to odd basis functions, which halves both the candidate basis and, since only one representative per pair is fitted, the design matrix; for a fixed budget, this affords a more expressive surrogate and better SV estimates.

Posterior Inference. The Hamming kernel GP surrogate of Section 2.3 admits the same restriction. The Hamming kernel is *complement-invariant*, i.e., argumentwise invariant in the sense of Ginsbourger et al. (2012) under complementation, $k_\xi(\bar{\mathbf{z}}, \bar{\mathbf{z}}') = k_\xi(\mathbf{z}, \mathbf{z}')$ for all $\mathbf{z}, \mathbf{z}' \in \{0,1\}^p$ (Appendix B.1). The Hamming GP prior thus decomposes into independent even and odd components.

Lemma 1 (Even-odd decomposition of the Hamming kernel GP)**.** *Let $\nu \sim \mathcal{GP}(m, k_\xi)$ be the Hamming kernel GP with constant mean $m \in \mathbb{R}$. Then $\nu = \nu_{\mathrm{odd}} + \nu_{\mathrm{even}}$ with independent GPs ν_{odd} and ν_{even}, where the odd component follows the* odd GP

$$\nu_{\mathrm{odd}} \sim \mathcal{GP}(0, k_{\xi,\mathrm{odd}}), \qquad k_{\xi,\mathrm{odd}}(\mathbf{z}, \mathbf{z}') := \tfrac{1}{2}\big(k_\xi(\mathbf{z}, \mathbf{z}') - k_\xi(\mathbf{z}, \bar{\mathbf{z}}')\big). \tag{10}$$

The lemma is a special case of the decomposition of GPs by (anti-)symmetrization (Haasdonk & Burkhardt, 2007; Klus et al., 2021) into independent components (Ginsbourger et al., 2016b, Cor. 2(b)); we include a direct proof in our notation in Appendix B.2. Crucially, the odd GP is precisely the odd surrogate required above: its posterior mean is odd for any training data (Ginsbourger et al., 2016a) (Lemma B.1 in Appendix B.2). It remains to fit the odd GP to the odd component only. To this end, let the data $\mathcal{D}_t$ be *paired*, i.e., contain the complement $\bar{\mathbf{z}}$ of every observed coalition $\mathbf{z}$, so that it consists of $n := t/2$ complement pairs. We collect one representative of each pair together with its odd target in the *odd dataset* $\mathcal{D}_{t,\mathrm{odd}} := \{(\mathbf{z}, \nu_{\mathrm{odd}}(\mathbf{z})) : \mathbf{z} \in \mathcal{R}_t\}$, where $\mathcal{R}_t \subset \{\mathbf{z}^{(i)}\}_{i=1}^t$ contains exactly one coalition of each pair (the choice is immaterial since $\nu_{\mathrm{odd}}(\bar{\mathbf{z}}) = -\nu_{\mathrm{odd}}(\mathbf{z})$), with $\mathbf{X}_{t,\mathrm{odd}} \in \{0,1\}^{n \times p}$ and $\mathbf{Y}_{t,\mathrm{odd}} \in \mathbb{R}^n$ defined analogously to $\mathbf{X}_t$ and $\mathbf{Y}_t$.

Theorem 2 (SV posterior equivalence under paired sampling)**.** *Let $\nu \sim \mathcal{GP}(m, k_\xi)$ be the Hamming kernel GP with fixed hyperparameters ξ and constant mean $m \in \mathbb{R}$, and let $\mathcal{D}_t$ be paired with odd dataset $\mathcal{D}_{t,\mathrm{odd}}$. Then $p(\boldsymbol{\phi} \mid \mathcal{D}_t) = p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}})$, where the left-hand side denotes the SV posterior of the full GP ν conditioned on $\mathcal{D}_t$, and the right-hand side the SV posterior of the odd GP $\nu_{\mathrm{odd}} \sim \mathcal{GP}(0, k_{\xi,\mathrm{odd}})$ from Equation* (10) *conditioned on the odd dataset $\mathcal{D}_{t,\mathrm{odd}}$.*

The proof is given in Appendix B.3, where we also extend the statement to i.i.d. Gaussian observation noise. Thus, for given hyperparameters, both the posterior mean and the posterior covariance of the SVs can be obtained from the odd GP conditioned on half as many observations.

Adaptive Coalition Selection. Given the odd GP, the EIG-based coalition selection of `ShaplEIG` (Section 2.3) naturally becomes *paired acquisition*: since only pairs yield odd targets, in each iteration a pair $\{\mathbf{z}, \bar{\mathbf{z}}\}$ is selected and evaluated instead of a single coalition, and the natural criterion is the EIG of the odd target under the odd GP.

Theorem 3 (EIG equivalence under paired acquisition)**.** *Under the assumptions of Theorem 2, for every candidate coalition $\mathbf{z} \in \{0,1\}^p$, $I\big(\boldsymbol{\phi}; \nu(\mathbf{z}), \nu(\bar{\mathbf{z}}) \mid \mathcal{D}_t\big) = I\big(\boldsymbol{\phi}; \nu_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_{t,\mathrm{odd}}\big) =: \mathrm{EIG}^{(t)}_{\boldsymbol{\phi},\mathrm{odd}}(\mathbf{z})$, i.e., the joint EIG about the SVs of evaluating the pair $\{\mathbf{z}, \bar{\mathbf{z}}\}$ under the full GP equals the EIG of the odd target $\nu_{\mathrm{odd}}(\mathbf{z})$ under the odd GP conditioned on the odd dataset.*

The proof is given in Appendix B.4, again including observation noise. Hence, under paired sampling, coalition selection can be performed entirely on the odd GP as well. Moreover, the odd EIG is invariant under complements, $\mathrm{EIG}^{(t)}_{\boldsymbol{\phi},\mathrm{odd}}(\mathbf{z}) = \mathrm{EIG}^{(t)}_{\boldsymbol{\phi},\mathrm{odd}}(\bar{\mathbf{z}})$ (Appendix B.4), so that one representative per candidate pair suffices.

Computational Consequences. By Theorems 2 and 3, the SV posterior and the EIG under the odd GP take the same form as Equations (4), (5), (7) and (8), but with the odd dataset $\mathcal{D}_{t,\mathrm{odd}}$ of half as many rows in place of $\mathcal{D}_t$: the training kernel matrix is four times smaller and its Cholesky decomposition up to 8 times faster. Moreover, the EIG has to be scored for only one representative per candidate pair, which halves the candidate set, and paired acquisition halves the number of iterations for a given budget. Finally, the prior covariances involving the Shapley transformation coincide with those of the full kernel (Proposition B.2 in Appendix B.5), so they need not be computed for the odd kernel, which would double their cost. See Appendix B.6 for details.

3.2 Quadrature-Based Computation of the Prior Covariances

The SV posterior (Equations (4) and (5)) and the EIG (Equations (7) and (8)) require the prior covariances $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})$ and $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$. By Proposition B.2, the same terms based on the full kernel serve the odd GP. We now show that both terms can be computed via quadrature. To this end, we first recall *product games* (Mohammadi et al., 2026), which arise, for instance, from the leaves of tree ensembles, and whose value function factorizes as $\nu(S) = \prod_{j \in S} u_j$ with player-specific factors $u_j \in \mathbb{R}$ (and $\nu(\emptyset) = 1$). Moreover, the Shapley weights in Equation (1) admit a classical integral representation (Owen, 1972), $1/\big(p\binom{p-1}{s}\big) = \int_0^1 \tau^s (1-\tau)^{p-1-s}\, d\tau$ for $s \in \{0, \ldots, p-1\}$. Plugging this representation into Equation (1) for a product game collapses its SVs into one-

dimensional integrals over products of $p-1$ affine factors, without enumerating coalitions (Li et al., 2026; Wettenstein et al., 2026; Mohammadi et al., 2026).

Linear Term. We now show that the Hamming kernel falls into this class: for a fixed coalition $\mathbf{z} \in \{0,1\}^p$, the kernel game $\nu_{\mathbf{z}}(S) := k_\xi(\mathbf{1}_S, \mathbf{z})$ is a product game in S up to a multiplicative constant (Appendix C.2). Consequently, the i-th entry of $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})$ is the SV of player i in $\nu_{\mathbf{z}}$, and the product game integral becomes

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})\big)_i = \phi_i(\nu_{\mathbf{z}}) = \sigma_\nu^2 \,(1-2z_i)(\beta_i - 1) \int_0^1 \prod_{j \neq i} \big(1 - (1-\beta_j)\,|\tau - z_j|\big)\, d\tau. \tag{11}$$

This merely specializes the product game result to the Hamming kernel; we include a derivation in our notation in Appendix C.3.

Quadratic Term. The prior covariance of the SVs $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$, in contrast, involves the Shapley weights twice: its (i,j)-th entry is the bilinear form induced by the kernel matrix, evaluated at the i-th and j-th row of $\mathbf{A}$, i.e., $(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top)_{i,j} = \mathbf{A}_i \mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}_j^\top$, so that both sets of Shapley weights can be represented as integrals.

Theorem 4 (Quadratic term)**.** *For $r \in P$ and $\tau, \tau' \in [0,1]$, let $h_r(\tau, \tau') := 1 - (1-\beta_r)(\tau + \tau' - 2\tau\tau')$. Then, for all players $i \neq j$,*

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top\big)_{i,j} = \sigma_\nu^2\,(1-\beta_i)(1-\beta_j) \iint_{[0,1]^2} (1-2\tau)(1-2\tau') \prod_{r \notin \{i,j\}} h_r(\tau, \tau')\, d\tau\, d\tau', \tag{12}$$

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top\big)_{i,i} = 2\,\sigma_\nu^2\,(1-\beta_i) \iint_{[0,1]^2} \prod_{r \neq i} h_r(\tau, \tau')\, d\tau\, d\tau'. \tag{13}$$

This bivariate integral representation has, to our knowledge, not been considered before; the proof is given in Appendix C.4.

Computational Cost. The integrands of Equations (11) to (13) are polynomials of degree $p-1$ in each integration variable, amenable to Gauss–Legendre (GL) quadrature (Golub & Welsch, 1969).

Corollary 5 (Cost and exactness of GL quadrature)**.** *Let $(\tau_q, \omega_q)_{q=1}^M$ be the GL nodes and weights on $[0,1]$. Replacing the integral in Equation* (11) *by $\sum_q \omega_q\,(\cdot)(\tau_q)$ and the integrals in Equations* (12) *and* (13) *by $\sum_{q,q'} \omega_q \omega_{q'}\,(\cdot)(\tau_q, \tau_{q'})$ costs $\mathcal{O}(pM)$ for all p entries of $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})$ per coalition $\mathbf{z}$ and $\mathcal{O}(p^2M^2)$ for $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$, and is exact for $M \geq \lceil p/2 \rceil$, i.e., at costs $\mathcal{O}(p^2)$ and $\mathcal{O}(p^4)$.*

The proof and implementation details are given in Appendices C.5 and C.6; for the linear term, this computation was proposed in prior work (Li et al., 2026; Wettenstein et al., 2026; Mohammadi et al., 2026). The quadrature route matches the asymptotic cost of the coefficient-based ESP schemes of `ShaplEIG` for exact quadrature (Appendix C.7), but the node evaluations are independent and thus parallelizable, whereas the ESP recursions are sequential, and M can be chosen below $\lceil p/2 \rceil$, which reduces the cost at the price of an approximation error and makes Hamming kernel GPs applicable to even larger games.

3.3 The ShapleyOG Algorithm

The preceding results culminate in `ShapleyOG`, our algorithm for fast adaptive SV estimation, as detailed in Algorithm 1. In the following, we outline its main differences to `ShaplEIG`; Appendix D details all involved quantities and their computational costs in comparison to `ShaplEIG`.

Refit Schedule and Incremental Updates. `ShaplEIG` refits the hyperparameters in every iteration or, for larger games, on a staged schedule (Rundel et al., 2026). `ShapleyOG` instead refits only when the number of observations t reaches a power of two, which reduces the number of refits for a budget T from $\mathcal{O}(T)$ to $\mathcal{O}(\log T)$. Between refits, the hyperparameters are fixed, which `ShapleyOG` exploits systematically: whereas `ShaplEIG` recomputed the Cholesky factorizations and triangular solves in every iteration even when its hyperparameters were kept fixed, we show in Appendix D.4 that these most costly operations need not be recomputed: the Cholesky factor is extended by a single row via a bordered factorization, and the half-solves against it by one forward

Algorithm 1 `ShapleyOG`

Require: Odd GP prior $\nu_{\text{odd}} \sim \mathcal{GP}(0, k_{\xi,\text{odd}})$ with hyperparameter prior $p(\xi)$, paired initial design of size T_0, evaluation budget T, number of GL nodes $M \leq \lceil p/2 \rceil$ (default: $M = \lceil p/2 \rceil$, exact)

Initialize dataset: $\mathcal{D}_{T_0} := \{(\mathbf{z}^{(i)}, \nu(\mathbf{z}^{(i)}))\}_{i=1}^{T_0}$; $t := T_0$
while $t < T$ **do**
  if $t = T_0$ or t is a power of two **then**
    Refit hyperparameters: $\xi := \arg\max_\xi \; p(\xi \mid \mathcal{D}_{t,\text{odd}})$ (Section 3.1)
    (Re-)sample paired candidate set $\mathcal{C} \subseteq \{0,1\}^p \setminus \{\mathbf{z}^{(i)}\}_{i=1}^{t}$ (Appendix D.3)
    Compute $\text{EIG}_{\boldsymbol{\phi},\text{odd}}^{(t)}(\mathbf{z})$ per pair in $\mathcal{C}$ (Sections 3.1 and 3.2)
  else
    Update $\text{EIG}_{\boldsymbol{\phi},\text{odd}}^{(t)}(\mathbf{z})$ per pair in $\mathcal{C}$ given new data (Appendix D.4)
  end if
  Select pair: $\mathbf{z}^{(t+1)} \in \arg\max_{\mathbf{z}\in\mathcal{C}} \text{EIG}_{\boldsymbol{\phi},\text{odd}}^{(t)}(\mathbf{z})$, $\mathbf{z}^{(t+2)} := \mathbf{1} - \mathbf{z}^{(t+1)}$ (Theorem 3)
  Update dataset: $\mathcal{D}_{t+2} := \mathcal{D}_t \cup \{(\mathbf{z}^{(t+1)}, \nu(\mathbf{z}^{(t+1)})), (\mathbf{z}^{(t+2)}, \nu(\mathbf{z}^{(t+2)}))\}$
  Update candidate set: $\mathcal{C} := \mathcal{C} \setminus \{\mathbf{z}^{(t+1)}, \mathbf{z}^{(t+2)}\}$; $t := t + 2$
end while
Refit ξ on $\mathcal{D}_{T,\text{odd}}$ and **return** SV posterior $\mathcal{N}_p(\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_{T,\text{odd}}}, \boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_{T,\text{odd}}})$ (Theorem 2)

substitution each, which reduces the overhead of an iteration without refit from $\mathcal{O}(n^3 + |\mathcal{C}|n^2)$ to $\mathcal{O}(n^2 + n|\mathcal{C}|)$, up to terms in p (Appendix D.2). Moreover, since all EIG quantities are recomputed at a refit anyway, `ShapleyOG` resamples a paired candidate set of not yet evaluated coalitions at every refit essentially for free, whereas `ShaplEIG` depletes a fixed set, which risks that only less informative candidates remain (Appendix D.3).

Paired Acquisition and Odd GP. Besides computing the prior covariances via GL quadrature (Section 3.2), `ShapleyOG` acquires coalitions in pairs, as required by the odd GP (Theorems 2 and 3); empirically, this restriction does not impair SV estimation (Section 4). Moreover, it learns the hyperparameters ξ under the odd GP by MAP estimation on the odd targets. While the equivalences of Theorems 2 and 3 hold for fixed ξ, fitting the odd GP to the odd data is not only cheaper by the same factors as in Section 3.1, but also more appropriate, as the odd component is all that matters for the SVs; empirically, it does not degrade adaptive SV estimation in most cases (Section 4).

4 Experiments

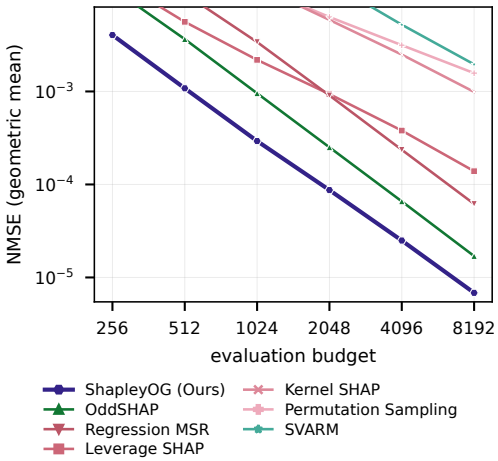
In this section, we evaluate `ShapleyOG` against state-of-the-art non-adaptive SV estimators in terms of estimation accuracy and total computation time, and we ablate its individual measures against `ShaplEIG`. We first describe the experimental setup and then present the results.

4.1 Experimental Setup

We run `ShapleyOG` as in Algorithm 1 with the following configuration. The initial design is the *border trick* design (Fumagalli et al., 2023) of size $T_0 = 2p + 2$, consisting of the empty and the grand coalition as well as all singletons and their complements. At every refit, a paired candidate set of 10240 coalitions is sampled from the not yet evaluated coalitions, and the quadrature uses the exact node count. We record the SV estimates at all budgets that are powers of two from the first above T_0 up to 8192 (capped at $2^p/2$, which only affects the 13-player Forest Fires game at 4096), where a budget counts all evaluations including the initial design; each game is run with 30 seeds (details in Appendix E.2).

Estimation Methods. We compare `ShapleyOG` against widely used and recent non-adaptive SV estimators: Kernel SHAP (Lundberg & Lee, 2017), Leverage SHAP (Musco & Witter, 2025), permutation sampling (Mitchell et al., 2022), SVARM (Kolpaczki et al., 2024), Regression MSR (Witter et al., 2025), and OddSHAP (Fumagalli et al., 2026a). All are run with the `shapiq` (Muschalik et al., 2024) defaults (Appendix E.3) and the same evaluation budgets; the ablation additionally compares against `ShaplEIG` (Rundel et al., 2026) (Appendix E.4).

Games. We consider 15 diverse explanation games with 13 to 111 players, i.e., 450 game instances over the 30 seeds. They comprise local explanations of vision and language models, of the tabular



Game (p) \ T	256	512	1024	2048	4096	8192
Language (IMDB) (14)	2.9	2.6	2.6	1.9	2.0	1.7
ResNet18-SP (14)	1.1	1.0	1.0	1.0	1.0	1.0
ViT4by4 (16)	2.0	1.2	1.2	1.2	1.4	1.6
DINOv2-16 (16)	2.5	2.1	1.8	1.8	1.9	2.3
Forest Fires (13)	3.1	3.1	3.3	3.4	3.3	–
AdultCensus (14)	1.1	1.2	1.1	1.4	2.0	2.6
Estate (15)	1.4	1.1	1.3	1.6	1.7	2.1
Crime (101)	1.0	1.0	1.0	1.0	1.0	1.0
Seismic Bumps (15)	1.0	1.3	1.1	1.4	1.5	2.0
Churn (19)	1.1	1.0	1.0	1.0	1.1	1.3
Credit-G (20)	1.4	2.0	2.2	1.6	1.8	2.1
Bankruptcy (64)	1.0	1.0	1.0	1.0	1.0	1.0
Superconductivity (81)	1.0	1.0	1.0	1.0	1.0	1.0
NATICUSdroid (86)	1.0	1.0	1.0	1.0	1.0	1.0
MIC (111)	1.0	1.0	1.0	1.0	1.0	1.0
pooled	1.4	1.3	1.3	1.3	1.4	1.5

Figure 1: Left: NMSE vs. evaluation budget, geometric mean over 30 seeds and the 14 games whose budget grid covers 256 to 8192 (Forest Fires, ending at 4096, is reported in Appendix F); the y-axis starts at the NMSE of `ShapleyOG` at 256 evaluations. Right: average NMSE rank of `ShapleyOG` among all estimators over 30 seeds per game (players in parentheses) and budget T (– beyond a game's grid; last row: 14 games pooled); gold/silver/bronze: lowest/second/third lowest average rank of all estimators.

foundation model TabPFN (Grinsztajn et al., 2026), and of LightGBM (Ke et al., 2017) gradient-boosted tree ensembles on tabular data, and are listed in Table 3 and detailed in Appendix E.1.

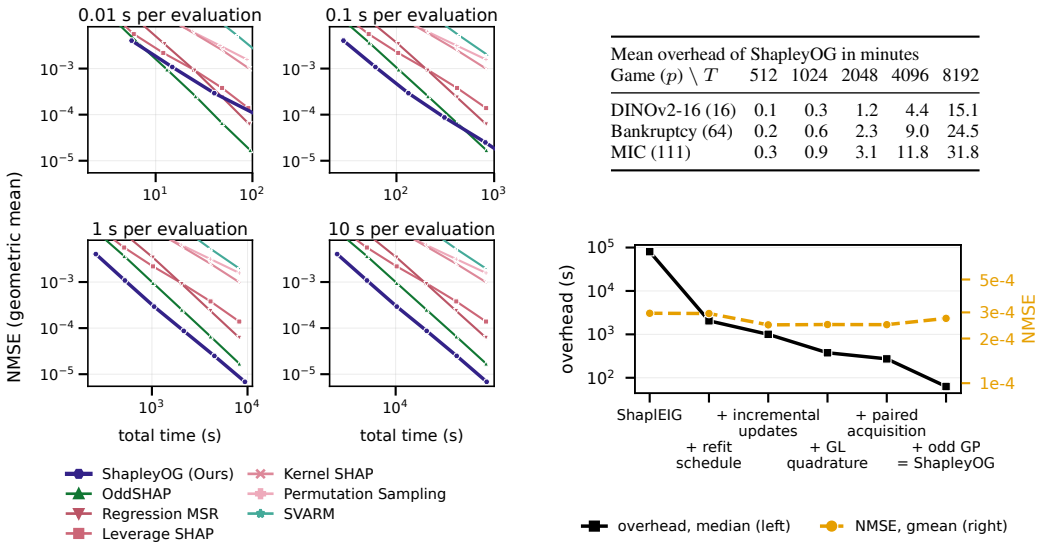
Metrics. We measure the estimation accuracy by the normalized mean squared error (NMSE) of the SV estimate, i.e., the squared error relative to the squared norm of the exact SVs, aggregated across seeds and games by the geometric mean. The overhead of an estimator is its wall-clock time excluding the value function evaluations, which for `ShapleyOG` comprises hyperparameter fitting and EIG maximization, and the total computation time is the overhead plus the number of evaluations times an assumed cost per evaluation.

4.2 Results

Estimation Accuracy. Figure 1 (left) pools the 14 games whose budget grid reaches 8192 evaluations (Forest Fires, with 13 players, ends at 4096). `ShapleyOG` attains the lowest NMSE at every budget, about a factor of three below OddSHAP, the strongest competitor, and needs about 40% fewer evaluations than OddSHAP for the same accuracy; per-game curves and arithmetic means in Appendix F.1 show the same picture. The average ranks (Figure 1, right; Appendix F.5) agree: pooled, `ShapleyOG` ranks lowest at every budget (1.3–1.5 vs. 2.0–2.6 for OddSHAP), first in every seed on the five tree games, and second or third on small games at large budgets, where OddSHAP catches up (Table 4), and on Forest Fires and IMDB, where none is consistently best.

Total Computation Time. Adaptivity costs overhead, so Figure 2 (left) plots the NMSE against the total computation time, i.e., the overhead of each estimator plus T times an assumed cost per value function evaluation: 0.01 s for high-throughput value functions, 0.1 s for a single forward pass of a model, and 1 s and 10 s for repeated forward passes or retraining. At 0.01 s the overhead of `ShapleyOG` dominates and the non-adaptive estimators reach a given accuracy faster. At 0.1 s, `ShapleyOG` is on the Pareto front except at the largest budgets, and from 1 s on it dominates every competitor at every budget. The overhead itself (Figure 2, top right) stays below a minute up to 1024 evaluations even for 111 players, in the range of minutes at 2048 to 4096, and up to half an hour at 8192 for the largest games. Appendix F.2 splits it into hyperparameter fitting and EIG maximization at refit and no-refit steps.

Ablation from `ShaplEIG` to `ShapleyOG`. To attribute the speed-up to the individual measures, we start from `ShaplEIG`, which refits the hyperparameters in every iteration, computes the prior covariances via ESPs, recomputes all factorizations in every iteration, and depletes a fixed candidate set, and add one measure at a time: the sparse refit schedule, the incremental updates with candidate resampling, the Gauss–Legendre quadrature, paired acquisition under the full GP, and finally the



Mean overhead of ShapleyOG in minutes

Game (p) \ T	512	1024	2048	4096	8192
DINOv2-16 (16)	0.1	0.3	1.2	4.4	15.1
Bankruptcy (64)	0.2	0.6	2.3	9.0	24.5
MIC (111)	0.3	0.9	3.1	11.8	31.8

Figure 2: Left: NMSE (geometric mean over the 14 games with the full budget grid and 30 seeds) vs. total computation time, i.e., overhead plus T times an assumed cost per evaluation of 0.01, 0.1, 1, and 10 seconds; markers are the budgets 256 to 8192. The overhead of a baseline is its wall-clock time minus the smallest among all baselines on the same instance and budget (Appendix E.5). Top right: mean overhead of `ShapleyOG` over 30 seeds for three games. Bottom right: ablation from `ShaplEIG` to `ShapleyOG` on Crime ($p = 101$, $T = 2048$), adding one measure at a time: overhead (median over 10 seeds, left axis) and NMSE (geometric mean, right axis).

odd GP, which yields `ShapleyOG` (Appendix E.4). Since `ShaplEIG` needs up to 22 hours for one run to 2048 evaluations, this study uses five games, 10 seeds, and budgets up to 2048. Figure 2 (bottom right) shows the Crime game: the NMSE stays within about 10% of `ShaplEIG` along the ladder, while the overhead falls from 22 hours to one minute, i.e., by a factor of 1250. Appendix F.3 reports the other four games: every measure reduces the overhead; the refit schedule (9–39×), the incremental updates (2–6×), and the odd GP (4–5× on top of paired acquisition) contribute most, the quadrature pays off with growing p (1.1–2.7×), and paired acquisition barely changes the accuracy. Relative to `ShaplEIG`, `ShapleyOG` is even more accurate on four of the five games (NMSE ratios 0.82–0.92) and less so on Credit-G (1.53), so overall the accuracy is maintained.

Quadrature at Large p. The ablation cannot separate the quadrature from the ESP route on games where both are cheap, so Appendix F.4 studies it on a further game, QSAR TID11 ($p = 1024$). There, 8 instead of the 514 Gauss–Legendre nodes leave the prior covariances exact to machine precision, and hence the EIG scores, the selected coalitions, and the NMSE unchanged, while the covariance computations drop from 16 minutes to 14 seconds and the whole routine takes 3.6 hours either way. Computing $\mathbf{AK}_\xi(\mathbf{Z}, \cdot)$ with the ESP route of `ShaplEIG` instead exceeded a 24-hour cap in every run. The quadrature thus makes games of this size feasible, and a handful of nodes suffices.

5 Conclusion

The regime in which adaptive SV estimation pays off can be broadened substantially: `ShapleyOG` keeps the accuracy of `ShaplEIG` and outperforms the non-adaptive estimators on most games and budgets, while reducing the overhead of `ShaplEIG` by orders of magnitude, so that in typical scenarios the evaluation cost outweighs the overhead already at a fraction of a second per evaluation. Two questions are open. First, on a few small games `ShapleyOG` is dominated by non-adaptive estimators, especially at large budgets, and it is unclear which structural properties of a game are responsible. Second, `ShapleyOG` scales to games with 1000 players, but its overhead still grows with the budget and the number of players, so very large budgets and games remain costly and whether the accuracy advantage persists there is open.

AI USE STATEMENT

In this work, we used generative AI tools to assist with research ideation and programming-related execution, as well as to draft and improve parts of the manuscript. All AI-assisted research contributions and text were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

To the best of our knowledge, this work does not raise any specific ethical concerns requiring further discussion.

REPRODUCIBILITY STATEMENT

The complete code of `ShapleyOG`, including the configuration files of all presented experiments, the scripts for preparing the games, and the aggregation and plotting scripts, is attached to this submission as supplementary material, so that all results are reproducible.

REFERENCES

George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*. Cambridge University Press, 1999. doi: 10.1017/CBO9781107325937.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html.

Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, Lennart Purucker, Sahithya Ravi, Jan N van Rijn, Prabhant Singh, Joaquin Vanschoren, Jos van der Velde, and Marcel Wever. OpenML: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7): 101317, 2025. doi: 10.1016/j.patter.2025.101317.

Mark A. Burgess and Archie C. Chapman. Approximating the Shapley value using stratified empirical Bernstein sampling. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 73–81. International Joint Conferences on Artificial Intelligence Organization, 2021. doi: 10.24963/ijcai.2021/11. URL https://doi.org/10.24963/ijcai.2021/11.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. doi: 10.1016/j.cor.2008.04.004. URL https://doi.org/10.1016/j.cor.2008.04.004.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pp. 273–304, 1995. doi: 10.1214/ss/1177009939.

Charalambos A Charalambides. *Enumerative combinatorics*. Chapman and Hall/CRC, 2018. doi: 10.1201/9781315273112.

Siu Lun Chau, Krikamol Muandet, and Dino Sejdinovic. Explaining the uncertain: Stochastic Shapley values for Gaussian process models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 50769–50795. Curran Associates, Inc., 2023. URL https://arxiv.org/abs/2305.15167.

Hugh Chen, Ian C Covert, Scott M Lundberg, and Su-In Lee. Algorithms to estimate Shapley value feature attributions. *Nature Machine Intelligence*, 5(6):590–601, 2023. doi: 10.1038/s42256-023-00657-x.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785–794. ACM, 2016.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2 edition, 2006. doi: 10.1002/047174882X. URL https://ci.nii.ac.jp/ncid/BA77851944.

Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3457–3465. PMLR, PMLR, 2021. URL https://proceedings.mlr.press/v130/covert21a.html.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://iclr.cc/virtual/2021/poster/3013.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 38, 2025.

Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. SHAP-IQ: Unified approximation of any-order Shapley interactions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11515–11551, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/264f2e10479c9370972847e96107db7f-Abstract-Conference.html.

Fabian Fumagalli, Landon Butler, Justin Singh Kang, Kannan Ramchandran, and R. Teal Witter. An odd estimator for Shapley values. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=xILwgiWAUk.

Fabian Fumagalli, R. Teal Witter, and Christopher Musco. PolySHAP: Extending KernelSHAP with interaction-informed polynomial regression. In *International Conference on Learning Representations*, 2026b. doi: 10.48550/arXiv.2601.18608. URL https://arxiv.org/abs/2601.18608.

Amirata Ghorbani and James Y. Zou. Data Shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, PMLR, 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

David Ginsbourger, Xavier Bay, Olivier Roustant, and Laurent Carraro. Argumentwise invariant kernels for the approximation of invariant functions. *Annales de la Faculté des Sciences de Toulouse: Mathématiques*, 21(3):501–527, 2012. doi: 10.5802/afst.1343.

David Ginsbourger, Olivier Roustant, and Nicolas Durrande. On degeneracy and invariances of random fields paths with applications in Gaussian process modelling. *Journal of Statistical Planning and Inference*, 170:117–128, 2016a. doi: 10.1016/j.jspi.2015.10.002.

David Ginsbourger, Olivier Roustant, Dominic Schuhmacher, Nicolas Durrande, and Nicolas Lenz. On ANOVA decompositions of kernels and Gaussian random field paths. In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2014*, volume 163 of *Springer Proceedings in Mathematics & Statistics*, pp. 315–330. Springer, 2016b. doi: 10.1007/978-3-319-33507-0_15.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4 edition, 2013.

Gene H. Golub and John H. Welsch. Calculation of Gauss quadrature rules. *Mathematics of Computation*, 23(106):221–230, 1969. doi: 10.1090/S0025-5718-69-99647-1.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *CoRR*, abs/2605.13986, 2026.

Bernard Haasdonk and Hans Burkhardt. Invariant kernel functions for pattern analysis and machine learning. *Machine Learning*, 68(1):35–61, 2007. doi: 10.1007/s10994-007-5009-7.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/cvpr.2016.90.

Frank Hutter. *Automated configuration of algorithms for solving hard computational problems*. PhD thesis, University of British Columbia, 2009. Chapter 12.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, volume 30, pp. 3146–3154, 2017.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository. https://archive.ics.uci.edu, 2023.

Stefan Klus, Patrick Gelß, Feliks Nüske, and Frank Noé. Symmetric and antisymmetric kernels for machine learning problems in quantum physics and chemistry. *Machine Learning: Science and Technology*, 2(4):045016, 2021. doi: 10.1088/2632-2153/ac14ad.

Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the Shapley value without marginal contributions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13246–13255, 2024. doi: 10.1609/aaai.v38i12.29225. URL https://ojs.aaai.org/index.php/AAAI/article/view/29225. Issue 12.

Weida Li, Yaoliang Yu, and Bryan Kian Hsiang Low. TreeGrad-Ranker: Feature ranking via $O(L)$-time gradients for decision trees. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2602.11623. URL https://openreview.net/forum?id=OcMeNbkN13.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956. doi: 10.1214/aoms/1177728069.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4765–4774, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1): 56–67, 2020. doi: 10.1038/s42256-019-0138-9. URL https://doi.org/10.1038/s42256-019-0138-9.

Ian Grant Macdonald. *Symmetric functions and Hall polynomials*. Oxford University Press, 2 edition, 1998. ISBN 9780198504504. URL https://ci.nii.ac.jp/ncid/BA3747188X.json.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for Shapley value estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022. URL https://jmlr.org/papers/v23/21-0439.html.

Majid Mohammadi, Siu Lun Chau, and Krikamol Muandet. Amortized linear-time exact Shapley value for product-kernel methods. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2505.16516.

Majid Mohammadi, Grigory Reznikov, Pavel Sinitcyn, Krikamol Muandet, and Siu Lun Chau. QuadraSHAP: Stable and scalable Shapley values for product games via Gauss–Legendre quadrature. *arXiv preprint*, 2026. doi: 10.48550/arXiv.2605.05870. URL https://arxiv.org/abs/2605.05870.

Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. shapiq: Shapley interactions for machine learning. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 37, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/eb3a9313405e2d4175a5a3cfcd49999b-Abstract-Datasets_and_Benchmarks_Track.html.

Christopher Musco and R. Teal Witter. Provably accurate Shapley value estimation via leverage score sampling. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a224ff18cc99a71751aa2b79118604da-Abstract-Conference.html.

Kieu Thao Nguyen Nguyen, Rachael Hwee Ling Sim, Quoc Phong Nguyen, See-Kiong Ng, and Bryan Kian Hsiang Low. Dupre: Data utility prediction for efficient data valuation. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1557–1565, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024, 2024.

Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5-part-2):64–79, 1972. doi: 10.1287/mnsc.18.5.64.

John Platt, Christopher J Burges, Steven Swenson, Christopher Weare, and Alice Zheng. Learning a Gaussian process prior for automatically generating music playlists. In *Advances in Neural Information Processing Systems*, volume 14, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/hash/6351bf9dce654515bf1ddbd6426dfa97-Abstract.html.

Peter Z. G. Qian, Huaiqing Wu, and C. F. Jeff Wu. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396, 2008. doi: 10.1198/004017008000000262.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern Bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024. doi: 10.1214/23-sts915.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006. doi: 10.7551/mitpress/3206.001.0001. URL https://cir.nii.ac.jp/crid/1971430859824246683.

David Rundel, Julius Kobialka, Constantin von Crailsheim, Matthias Feurer, Thomas Nagler, and David Rügamer. Interpretable machine learning for TabPFN. In *World Conference on Explainable Artificial Intelligence*, pp. 465–476. Springer, 2024. doi: 10.1007/978-3-031-63797-1_23.

David Rundel, Fabian Fumagalli, Maximilian Muschalik, Bernd Bischl, and Matthias Feurer. ShaplEIG: Bayesian experimental design for Shapley value estimation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.02247.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games (AM-28), Volume II*, pp. 307–317. Princeton University Press, 1953. doi: 10.1515/9781400881970-018. URL https://doi.org/10.1515/9781400881970-018.

Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950. doi: 10.1214/aoms/1177729893.

Dylan Slack, Sophie Hilgard, Sameer Singh, and Himabindu Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9391–9404, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4e246a381baf2ce038b3b0f82c7d6fb4-Abstract.html.

Michael L Stein. *Interpolation of Spatial Data*. Springer Science & Business Media, 1999. doi: 10.1007/978-1-4612-1494-6.

Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(1):1–18, 2010. doi: 10.5555/1756006.1756007. URL https://jmlr.org/papers/v11/strumbelj10a.html.

Lloyd N. Trefethen. Is Gauss quadrature better than Clenshaw–Curtis? *SIAM Review*, 50(1):67–87, 2008. doi: 10.1137/060659831.

Albertus J van Es and Roelof Helmers. Elementary symmetric polynomials of increasing order. *Probability Theory and Related Fields*, 80(1):21–35, 1988. doi: 10.1007/bf00348750.

Ron Wettenstein, Rory Mitchell, and Peng Yu. Quadrature-TreeSHAP: Depth-independent TreeSHAP and Shapley interactions. *arXiv preprint*, 2026. doi: 10.48550/arXiv.2605.04497. URL https://arxiv.org/abs/2605.04497.

Marcel Wever, Maximilian Muschalik, Fabian Fumagalli, and Marius Lindauer. HyperSHAP: Shapley values and interactions for explaining hyperparameter optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 26867–26875, 2026. doi: 10.1609/aaai.v40i32.39898.

Christopher Williams and Carl Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, volume 8, 1995. URL https://proceedings.neurips.cc/paper_files/paper/1995/hash/7cce53cf90577442771720a370c3c723-Abstract.html.

R. Teal Witter, Yurong Liu, and Christopher Musco. Regression-adjusted Monte Carlo estimators for Shapley values and probabilistic values. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0715. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1e77af93008ee6cd248a31723ce357d8-Abstract-Conference.html.

Artjom Zern, Klaus Broelemann, and Gjergji Kasneci. Interventional SHAP values and interaction values for piecewise linear regression trees. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, volume 37, pp. 11164–11173, 2023. doi: 10.1609/aaai.v37i9.26322.

A Background

A.1 Gaussian Processes

A Gaussian process (GP; Williams & Rasmussen, 1995; Rasmussen & Williams, 2006) over functions on $\{0,1\}^p$ is fully characterized by a mean function $m : \{0,1\}^p \to \mathbb{R}$ and a positive definite kernel $k_\xi : \{0,1\}^p \times \{0,1\}^p \to \mathbb{R}$ with hyperparameters ξ, and we write $\nu \sim \mathcal{GP}(m, k_\xi)$. Given data $\mathcal{D}_t$ with coalitions $\mathbf{X}_t \in \{0,1\}^{t\times p}$ and targets $\mathbf{Y}_t \in \mathbb{R}^t$, the latter corrupted by additive Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ that is i.i.d. across observations, the posterior is again a GP, and the posterior predictive distribution at any s coalitions $\mathbf{X}_\star \in \{0,1\}^{s\times p}$ is a multivariate Gaussian, $\nu(\mathbf{X}_\star) \mid \mathcal{D}_t \sim \mathcal{N}_s(\boldsymbol{\mu}_{\nu(\mathbf{X}_\star)|\mathcal{D}_t}, \boldsymbol{\Sigma}_{\nu(\mathbf{X}_\star)|\mathcal{D}_t})$. For a zero mean function, $m = 0$,

$$\boldsymbol{\mu}_{\nu(\mathbf{X}_\star)|\mathcal{D}_t} = \mathbf{K}_\xi(\mathbf{X}_\star, \mathbf{X}_t)\left[\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2\mathbf{I}\right]^{-1}\mathbf{Y}_t, \tag{14}$$

$$\boldsymbol{\Sigma}_{\nu(\mathbf{X}_\star)|\mathcal{D}_t} = \mathbf{K}_\xi(\mathbf{X}_\star, \mathbf{X}_\star) - \mathbf{K}_\xi(\mathbf{X}_\star, \mathbf{X}_t)\left[\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2\mathbf{I}\right]^{-1}\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_\star), \tag{15}$$

where $\mathbf{K}_\xi(\mathbf{U}, \mathbf{V})$ denotes the kernel matrix with entries $k_\xi(\mathbf{U}_i, \mathbf{V}_j)$ for the rows of $\mathbf{U}$ and $\mathbf{V}$.

Computational Cost. The training kernel matrix $\left[\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2\mathbf{I}\right] \in \mathbb{R}^{t\times t}$ is typically decomposed via a Cholesky factorization in $\mathcal{O}(t^3)$. The second term of Equation (15) then requires triangular solves with s right-hand sides in $\mathcal{O}(st^2)$ and a matrix product in $\mathcal{O}(s^2t)$, i.e., $\mathcal{O}(t^3 + st^2 + s^2t)$ in total. If only the predictive variances, i.e., the diagonal of Equation (15), are required, the cost reduces to $\mathcal{O}(t^3 + st^2)$.

Hamming Kernel. The Hamming kernel of Equation (2) can be written as

$$k_\xi(\mathbf{z}, \mathbf{z}') = \sigma_\nu^2 \prod_{j\in P} \exp\big(-\mathbb{1}[z_j \neq z'_j]/\ell_j\big) = \sigma_\nu^2 \prod_{j\in P} \big(\alpha_j \mathbb{1}[z_j = z'_j] + \beta_j \mathbb{1}[z_j \neq z'_j]\big) \tag{16}$$

with $\alpha_j := 1$ and $\beta_j := \exp(-1/\ell_j)$, i.e., it quantifies the similarity of two coalitions via a weighted Hamming distance $\sum_j \mathbb{1}[z_j \neq z'_j]/\ell_j$. Player j contributes the factor α_j if both coalitions agree on its membership and β_j if they differ, so that agreement leaves the covariance unchanged, whereas each disagreement reduces it by the player-specific factor $\beta_j < 1$. Small lengthscales thus encode players whose membership changes the value function strongly. The general form with agreement factors α_j and disagreement factors β_j is used in Appendix C.

Hyperparameter Fitting. The kernel hyperparameters ξ are learned by maximizing the log marginal likelihood $\log p(\mathbf{Y}_t \mid \mathbf{X}_t, \xi)$ of the data or, given a prior $p(\xi)$, the log posterior $\log p(\mathbf{Y}_t \mid \mathbf{X}_t, \xi) + \log p(\xi)$, i.e., by MAP estimation. Both objectives and their gradients with respect to ξ are available in closed form (Rasmussen & Williams, 2006), so that gradient-based optimizers apply, where each objective evaluation requires a Cholesky factorization of the training kernel matrix in $\mathcal{O}(t^3)$.

Quasi-Noiseless GPs. A noiseless GP is the limiting case $\sigma_\epsilon^2 = 0$, in which the posterior mean interpolates the training data exactly if the training kernel matrix is non-singular (Stein, 1999; Rasmussen & Williams, 2006). In practice, σ_ϵ^2 is instead fixed to a small positive constant, which keeps the Cholesky factorization numerically stable and is equivalent to assuming a small observation noise.

A.2 Adaptive SV Estimation with ShaplEIG

In each iteration, `ShaplEIG` evaluates the EIG of Equations (7) and (8) for all candidates $\mathbf{z} \in \mathcal{C}$ under the full GP with the current hyperparameters ξ,

$$\mathrm{EIG}_{\boldsymbol{\phi}}^{(t)}(\mathbf{z}) \propto \log \mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}_t) - \log\big(\mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}_t) - Q_t(\mathbf{z})\big), \tag{17}$$

$$Q_t(\mathbf{z}) = \big(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{e}_\mathbf{z}\big)^\top \big(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{A}^\top\big)^{-1}\big(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{e}_\mathbf{z}\big).$$

We describe its vectorized computation over $\mathcal{C}$, collecting the basis vectors $\mathbf{e}_\mathbf{z}$ of the candidates as columns of $\mathbf{E}_\mathcal{C} \in \mathbb{R}^{2^p\times|\mathcal{C}|}$, and state for every step the size of the involved quantities, their cost, and what is recycled from previous steps. The three posterior quantities in Equation (17) follow from Equation (15) and are assembled in Step 4.

Step 1: Prior Covariances. The prior covariances $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top \in \mathbb{R}^{p\times p}$, $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}_t) \in \mathbb{R}^{p\times t}$, and $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathcal{C}) \in \mathbb{R}^{p\times|\mathcal{C}|}$ are computed via ESPs in $\mathcal{O}(p^4)$, $\mathcal{O}(tp^2)$, and $\mathcal{O}(|\mathcal{C}|p^2)$, respectively (Section 2.3).

Step 2: Training Terms. The training kernel matrix $\left[\mathbf{K}_\xi(\mathbf{X}_t,\mathbf{X}_t)+\sigma_\epsilon^2\mathbf{I}\right] \in \mathbb{R}^{t\times t}$ is formed in $\mathcal{O}(t^2p)$ and factorized via Cholesky as $\left[\mathbf{K}_\xi(\mathbf{X}_t,\mathbf{X}_t)+\sigma_\epsilon^2\mathbf{I}\right] = \mathbf{L}\mathbf{L}^\top$ in $\mathcal{O}(t^3)$. The half-solve $\mathbf{Y}_\mathbf{X} := \mathbf{L}^{-1}(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}_t))^\top \in \mathbb{R}^{t\times p}$ then costs $\mathcal{O}(t^2p)$ and reuses $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}_t)$ from Step 1.

Step 3: Candidate Terms. The candidate kernel matrix $\mathbf{K}_\xi(\mathbf{X}_t,\mathcal{C}) \in \mathbb{R}^{t\times|\mathcal{C}|}$ is formed in $\mathcal{O}(t|\mathcal{C}|p)$, and the half-solve $\mathbf{Y}_\mathcal{C} := \mathbf{L}^{-1}\mathbf{K}_\xi(\mathbf{X}_t,\mathcal{C}) \in \mathbb{R}^{t\times|\mathcal{C}|}$ costs $\mathcal{O}(t^2|\mathcal{C}|)$, reusing $\mathbf{L}$ from Step 2.

Step 4: Posterior Covariances. From Steps 1 to 3, the three posterior quantities of Equation (17) are obtained for all candidates at once, since every product of the form $\mathbf{K}_\xi(\cdot,\mathbf{X}_t)\left[\mathbf{K}_\xi(\mathbf{X}_t,\mathbf{X}_t)+\sigma_\epsilon^2\mathbf{I}\right]^{-1}\mathbf{K}_\xi(\mathbf{X}_t,\cdot)$ in Equation (15) is a product of two half-solves, $(\mathbf{L}^{-1}\mathbf{K}_\xi(\mathbf{X}_t,\cdot))^\top(\mathbf{L}^{-1}\mathbf{K}_\xi(\mathbf{X}_t,\cdot))$:

$$\mathbf{v} := \left(\operatorname{Var}(\nu(\mathbf{z}) \mid \mathcal{D}_t)\right)_{\mathbf{z}\in\mathcal{C}} = \sigma_\nu^2\mathbf{1} - \operatorname{diag}\left(\mathbf{Y}_\mathcal{C}^\top\mathbf{Y}_\mathcal{C}\right) \in \mathbb{R}^{|\mathcal{C}|}, \qquad \mathcal{O}(t|\mathcal{C}|), \tag{18}$$

$$\mathbf{B} := \mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{E}_\mathcal{C} = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathcal{C}) - \mathbf{Y}_\mathbf{X}^\top\mathbf{Y}_\mathcal{C} \in \mathbb{R}^{p\times|\mathcal{C}|}, \qquad \mathcal{O}(tp|\mathcal{C}|), \tag{19}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t} = \mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{A}^\top = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top - \mathbf{Y}_\mathbf{X}^\top\mathbf{Y}_\mathbf{X} \in \mathbb{R}^{p\times p}, \qquad \mathcal{O}(tp^2), \tag{20}$$

where $\operatorname{diag}(\cdot)$ extracts the diagonal, i.e., the column sums of squares, and $k_\xi(\mathbf{z},\mathbf{z}) = \sigma_\nu^2$ for the Hamming kernel. The columns of $\mathbf{B}$ are the cross-covariances $\operatorname{Cov}(\boldsymbol{\phi},\nu(\mathbf{z}) \mid \mathcal{D}_t) = \mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}|\mathcal{D}_t}\mathbf{e}_\mathbf{z}$, and Equation (20) is the SV posterior covariance of Equation (5).

Step 5: EIG. With the Cholesky factor $\mathbf{L}_{\boldsymbol{\phi}}$ of $\boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t}$, computed in $\mathcal{O}(p^3)$, one triangular solve with $|\mathcal{C}|$ right-hand sides in $\mathcal{O}(p^2|\mathcal{C}|)$ yields

$$\mathbf{q} := \left(Q_t(\mathbf{z})\right)_{\mathbf{z}\in\mathcal{C}} = \operatorname{diag}\left((\mathbf{L}_{\boldsymbol{\phi}}^{-1}\mathbf{B})^\top(\mathbf{L}_{\boldsymbol{\phi}}^{-1}\mathbf{B})\right) \in \mathbb{R}^{|\mathcal{C}|}, \tag{21}$$

$$\left(\mathrm{EIG}_{\boldsymbol{\phi}}^{(t)}(\mathbf{z})\right)_{\mathbf{z}\in\mathcal{C}} \propto \log\mathbf{v} - \log(\mathbf{v}-\mathbf{q}),$$

with elementwise logarithms. In total, the EIG over $\mathcal{C}$ costs $\mathcal{O}(t^3 + t^2|\mathcal{C}| + p^4)$ for $t \geq p$, as stated in Section 2.3.

SV Posterior. The SV posterior of Equations (4) and (5) consists of the same components: its covariance is Equation (20), and its mean $\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_t} = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}_t)\left[\mathbf{K}_\xi(\mathbf{X}_t,\mathbf{X}_t)+\sigma_\epsilon^2\mathbf{I}\right]^{-1}\mathbf{Y}_t = \mathbf{Y}_\mathbf{X}^\top\mathbf{L}^{-1}\mathbf{Y}_t$ additionally requires one triangular solve with the targets in $\mathcal{O}(t^2)$ and a product in $\mathcal{O}(tp)$. Given Steps 1 and 2, it is thus available in $\mathcal{O}(t^3 + t^2p + p^4)$, as stated in Section 2.3.

Recycling between Iterations. Since `ShaplEIG` refits ξ in every iteration, it generally recomputes all steps. Whenever ξ is kept fixed, Step 1 is recycled entirely, as $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top$ remains valid and $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}_t)$ and $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathcal{C})$ change by a single column, and so do the kernel matrices $\left[\mathbf{K}_\xi(\mathbf{X}_t,\mathbf{X}_t)+\sigma_\epsilon^2\mathbf{I}\right]$ and $\mathbf{K}_\xi(\mathbf{X}_t,\mathcal{C})$ by a single row and column; `ShaplEIG` exploits this but recomputes the factorization, both half-solves, and Steps 4 and 5.

B Proofs and Details for Section 3.1

Throughout this section, the hyperparameters ξ are fixed, and we use the notation of Section 3.1. For the convenience of the reader, we restate the results of the main text before their proofs.

B.1 Complement Invariance of the Hamming Kernel

The Hamming kernel depends on its arguments only through the mismatch indicators $\mathbb{1}[z_j \neq z'_j]$, $j \in P$ (Appendix A.1), which complementing both arguments leaves unchanged. Hence it is *complement-invariant*, $k_\xi(\bar{\mathbf{z}}, \bar{\mathbf{z}}') = k_\xi(\mathbf{z}, \mathbf{z}')$, and consequently $k_\xi(\bar{\mathbf{z}}, \mathbf{z}') = k_\xi(\mathbf{z}, \bar{\mathbf{z}}')$ for all $\mathbf{z}, \mathbf{z}' \in \{0,1\}^p$. Both identities are used throughout this appendix.

B.2 Even-Odd Decomposition of the GP Prior

Lemma 1 (Even-odd decomposition of the Hamming kernel GP). *Let $\nu \sim \mathcal{GP}(m, k_\xi)$ be the Hamming kernel GP with constant mean $m \in \mathbb{R}$. Then $\nu = \nu_{\text{odd}} + \nu_{\text{even}}$ with independent GPs ν_{odd} and ν_{even}, where the odd component follows the odd GP $\nu_{\text{odd}} \sim \mathcal{GP}(0, k_{\xi,\text{odd}})$ with $k_{\xi,\text{odd}}$ as in Equation* (10)*.*

The lemma follows from the complement invariance of the Hamming kernel (Ginsbourger et al., 2012): by (anti-)symmetrization of the kernel (Haasdonk & Burkhardt, 2007; Klus et al., 2021), the GP decomposes into independent even and odd GPs (Ginsbourger et al., 2016b, Cor. 2(b)). We only sketch the computation that uses complement invariance.

Proof sketch. Write $\bar{\nu}(\mathbf{z}) := \nu(\bar{\mathbf{z}})$. Since $\bar{\nu}$ consists of the same random variables as ν, the pair $(\nu, \bar{\nu})$ is jointly Gaussian, and so are the linear transformations $\nu_{\text{odd}} = \frac{1}{2}(\nu - \bar{\nu})$ and $\nu_{\text{even}} = \frac{1}{2}(\nu + \bar{\nu})$, with $\mathbb{E}[\nu_{\text{odd}}(\mathbf{z})] = 0$ and $\mathbb{E}[\nu_{\text{even}}(\mathbf{z})] = m$. By complement invariance (Appendix B.1), $\text{Cov}(\bar{\nu}(\mathbf{z}), \bar{\nu}(\mathbf{z}')) = k_\xi(\mathbf{z}, \mathbf{z}')$ and $\text{Cov}(\nu(\mathbf{z}), \bar{\nu}(\mathbf{z}')) = k_\xi(\mathbf{z}, \bar{\mathbf{z}}') = \text{Cov}(\bar{\nu}(\mathbf{z}), \nu(\mathbf{z}'))$, hence by bilinearity

$$\begin{aligned}
\text{Cov}\left(\nu_{\text{odd}}(\mathbf{z}), \nu_{\text{odd}}(\mathbf{z}')\right) &= \tfrac{1}{4}\left[k_\xi(\mathbf{z}, \mathbf{z}') - 2k_\xi(\mathbf{z}, \bar{\mathbf{z}}') + k_\xi(\mathbf{z}, \mathbf{z}')\right] = k_{\xi,\text{odd}}(\mathbf{z}, \mathbf{z}'), \\
\text{Cov}\left(\nu_{\text{even}}(\mathbf{z}), \nu_{\text{even}}(\mathbf{z}')\right) &= \tfrac{1}{2}\left(k_\xi(\mathbf{z}, \mathbf{z}') + k_\xi(\mathbf{z}, \bar{\mathbf{z}}')\right) =: k_{\xi,\text{even}}(\mathbf{z}, \mathbf{z}'), \\
\text{Cov}\left(\nu_{\text{odd}}(\mathbf{z}), \nu_{\text{even}}(\mathbf{z}')\right) &= \tfrac{1}{4}\left[k_\xi(\mathbf{z}, \mathbf{z}') + k_\xi(\mathbf{z}, \bar{\mathbf{z}}') - k_\xi(\mathbf{z}, \bar{\mathbf{z}}') - k_\xi(\mathbf{z}, \mathbf{z}')\right] = 0.
\end{aligned}$$

Jointly Gaussian processes with vanishing cross-covariance are independent, which proves the claim. Here, $k_{\xi,\text{odd}}$ and $k_{\xi,\text{even}}$ are the antisymmetrization and symmetrization of k_ξ under complementation (Haasdonk & Burkhardt, 2007; Klus et al., 2021), and $k_{\xi,\text{odd}} + k_{\xi,\text{even}} = k_\xi$. □

Lemma B.1 (The posterior mean of the odd GP is odd)**.** *Let $\nu_{\text{odd}} \sim \mathcal{GP}(0, k_{\xi,\text{odd}})$ with $k_{\xi,\text{odd}}$ as in Equation* (10)*. Then, for any dataset $\mathcal{D}$ of observations of ν_{odd} at coalitions $\mathbf{X}$ with targets $\mathbf{Y}$ and noise variance $\sigma_\epsilon^2 \geq 0$, the posterior mean $\mu(\mathbf{z}) := \mathbb{E}[\nu_{\text{odd}}(\mathbf{z}) \mid \mathcal{D}]$ satisfies $\mu(\bar{\mathbf{z}}) = -\mu(\mathbf{z})$ for all $\mathbf{z} \in \{0,1\}^p$.*

The lemma is a special case of Ginsbourger et al. (2016a, Props. 3 and 8).

Proof. By complement invariance (Appendix B.1), $k_\xi(\bar{\mathbf{z}}, \mathbf{z}') = k_\xi(\mathbf{z}, \bar{\mathbf{z}}')$ and $k_\xi(\bar{\mathbf{z}}, \bar{\mathbf{z}}') = k_\xi(\mathbf{z}, \mathbf{z}')$, hence

$$k_{\xi,\text{odd}}(\bar{\mathbf{z}}, \mathbf{z}') = \tfrac{1}{2}\left(k_\xi(\bar{\mathbf{z}}, \mathbf{z}') - k_\xi(\bar{\mathbf{z}}, \bar{\mathbf{z}}')\right) = \tfrac{1}{2}\left(k_\xi(\mathbf{z}, \bar{\mathbf{z}}') - k_\xi(\mathbf{z}, \mathbf{z}')\right) = -k_{\xi,\text{odd}}(\mathbf{z}, \mathbf{z}'),$$

i.e., $k_{\xi,\text{odd}}$ is odd in its first argument and, by symmetry, in its second. Consequently, $\mathbf{K}_{\xi,\text{odd}}(\bar{\mathbf{z}}, \mathbf{X}) = -\mathbf{K}_{\xi,\text{odd}}(\mathbf{z}, \mathbf{X})$, and the posterior mean $\mu(\mathbf{z}) = \mathbf{K}_{\xi,\text{odd}}(\mathbf{z}, \mathbf{X})\left[\mathbf{K}_{\xi,\text{odd}}(\mathbf{X}, \mathbf{X}) + \sigma_\epsilon^2 \mathbf{I}\right]^{-1} \mathbf{Y}$ (Equation (14)) satisfies $\mu(\bar{\mathbf{z}}) = -\mu(\mathbf{z})$. □

B.3 Posterior Equivalence under Paired Sampling

Theorem 2 (SV posterior equivalence under paired sampling). *Let $\nu \sim \mathcal{GP}(m, k_\xi)$ be the Hamming kernel GP with fixed hyperparameters ξ and constant mean $m \in \mathbb{R}$, and let $\mathcal{D}_t$ be paired with odd dataset $\mathcal{D}_{t,\text{odd}}$. Then $p(\boldsymbol{\phi} \mid \mathcal{D}_t) = p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\text{odd}})$, where the left-hand side denotes the SV*

posterior of the full GP ν conditioned on $\mathcal{D}_t$, and the right-hand side the SV posterior of the odd GP $\nu_{\mathrm{odd}} \sim \mathcal{GP}(0, k_{\xi,\mathrm{odd}})$ from Equation (10) *conditioned on the odd dataset $\mathcal{D}_{t,\mathrm{odd}}$.*

The proof uses the Hamming kernel only through its complement invariance (Appendix B.1).

Proof. We prove the statement for observations corrupted by i.i.d. Gaussian noise with variance $\sigma_\epsilon^2 \geq 0$; the theorem as stated in the main text is the case $\sigma_\epsilon^2 = 0$. Let $\mathcal{R}_t$ be the set of representatives from Section 3.1, i.e., the rows of $\mathbf{X}_{t,\mathrm{odd}}$. The paired dataset consists of the observations

$$y(\mathbf{z}) = \nu(\mathbf{z}) + \epsilon_{\mathbf{z}}, \qquad y(\bar{\mathbf{z}}) = \nu(\bar{\mathbf{z}}) + \epsilon_{\bar{\mathbf{z}}}, \qquad \mathbf{z} \in \mathcal{R}_t,$$

with all $\epsilon_{\mathbf{z}}, \epsilon_{\bar{\mathbf{z}}} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2)$ independent of ν (for $\sigma_\epsilon^2 = 0$ the observations are exact), i.e., $\mathcal{D}_t = \{(\mathbf{z}, y(\mathbf{z})), (\bar{\mathbf{z}}, y(\bar{\mathbf{z}})) : \mathbf{z} \in \mathcal{R}_t\}$. For each pair, define the odd and even targets

$$y_{\mathrm{odd}}(\mathbf{z}) := \tfrac{1}{2}\big(y(\mathbf{z}) - y(\bar{\mathbf{z}})\big) = \nu_{\mathrm{odd}}(\mathbf{z}) + \epsilon_{\mathrm{odd}}(\mathbf{z}),$$
$$y_{\mathrm{even}}(\mathbf{z}) := \tfrac{1}{2}\big(y(\mathbf{z}) + y(\bar{\mathbf{z}})\big) = \nu_{\mathrm{even}}(\mathbf{z}) + \epsilon_{\mathrm{even}}(\mathbf{z}),$$

with $\epsilon_{\mathrm{odd}}(\mathbf{z}) := \frac{1}{2}(\epsilon_{\mathbf{z}} - \epsilon_{\bar{\mathbf{z}}})$ and $\epsilon_{\mathrm{even}}(\mathbf{z}) := \frac{1}{2}(\epsilon_{\mathbf{z}} + \epsilon_{\bar{\mathbf{z}}})$, and collect them, together with the representatives, in the odd and even datasets

$$\mathcal{D}_{t,\mathrm{odd}} := \{(\mathbf{z}, y_{\mathrm{odd}}(\mathbf{z})) : \mathbf{z} \in \mathcal{R}_t\} \qquad \text{and} \qquad \mathcal{D}_{t,\mathrm{even}} := \{(\mathbf{z}, y_{\mathrm{even}}(\mathbf{z})) : \mathbf{z} \in \mathcal{R}_t\},$$

with targets stacked in $\mathbf{Y}_{t,\mathrm{odd}}, \mathbf{Y}_{t,\mathrm{even}} \in \mathbb{R}^n$ in the row order of $\mathbf{X}_{t,\mathrm{odd}}$. For $\sigma_\epsilon^2 = 0$, $\mathcal{D}_{t,\mathrm{odd}}$ coincides with the odd dataset of the main text.

Step 1: Conditioning on $\mathcal{D}_t$ is conditioning on $(\mathcal{D}_{t,\mathrm{odd}}, \mathcal{D}_{t,\mathrm{even}})$. For each $\mathbf{z} \in \mathcal{R}_t$, the observations and the odd and even targets are related by the linear map

$$\begin{pmatrix} y_{\mathrm{odd}}(\mathbf{z}) \\ y_{\mathrm{even}}(\mathbf{z}) \end{pmatrix} = \tfrac{1}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} y(\mathbf{z}) \\ y(\bar{\mathbf{z}}) \end{pmatrix}, \qquad \text{with inverse} \qquad \begin{pmatrix} y(\mathbf{z}) \\ y(\bar{\mathbf{z}}) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} y_{\mathrm{odd}}(\mathbf{z}) \\ y_{\mathrm{even}}(\mathbf{z}) \end{pmatrix}.$$

Since the representatives are the same on both sides, $\mathcal{D}_t$ and $(\mathcal{D}_{t,\mathrm{odd}}, \mathcal{D}_{t,\mathrm{even}})$ are in one-to-one correspondence and thus carry exactly the same information. Consequently, conditioning on either yields the same posterior,

$$p(\boldsymbol{\phi} \mid \mathcal{D}_t) = p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}}, \mathcal{D}_{t,\mathrm{even}}).$$

Step 2: The noise decomposes independently. The vector $(\epsilon_{\mathrm{odd}}(\mathbf{z}), \epsilon_{\mathrm{even}}(\mathbf{z}))$ is Gaussian with

$$\mathrm{Var}(\epsilon_{\mathrm{odd}}(\mathbf{z})) = \mathrm{Var}(\epsilon_{\mathrm{even}}(\mathbf{z})) = \tfrac{1}{4}(\sigma_\epsilon^2 + \sigma_\epsilon^2) = \tfrac{\sigma_\epsilon^2}{2},$$
$$\mathrm{Cov}(\epsilon_{\mathrm{odd}}(\mathbf{z}), \epsilon_{\mathrm{even}}(\mathbf{z})) = \tfrac{1}{4}(\sigma_\epsilon^2 - \sigma_\epsilon^2) = 0,$$

so $\epsilon_{\mathrm{odd}}(\mathbf{z})$ and $\epsilon_{\mathrm{even}}(\mathbf{z})$ are independent (Rasmussen & Williams, 2006), and both are independent across $\mathbf{z} \in \mathcal{R}_t$ and of ν.

Step 3: The even dataset is uninformative about the SVs. By Lemma 1 and Step 2, $(\nu_{\mathrm{odd}}, \epsilon_{\mathrm{odd}})$ and $(\nu_{\mathrm{even}}, \epsilon_{\mathrm{even}})$ are independent. Since the SVs of any even function vanish (Fumagalli et al., 2026a),

$$\boldsymbol{\phi}(\nu) = \mathbf{A}\boldsymbol{\nu} = \mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z}) + \mathbf{A}\nu_{\mathrm{even}}(\mathbf{Z}) = \mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z}) + \mathbf{0} = \boldsymbol{\phi}(\nu_{\mathrm{odd}}),$$

so $\boldsymbol{\phi}$ and $\mathcal{D}_{t,\mathrm{odd}}$ are functions of $(\nu_{\mathrm{odd}}, \epsilon_{\mathrm{odd}})$ only, whereas $\mathcal{D}_{t,\mathrm{even}}$ is a function of $(\nu_{\mathrm{even}}, \epsilon_{\mathrm{even}})$. Hence $\mathcal{D}_{t,\mathrm{even}}$ is independent of $(\boldsymbol{\phi}, \mathcal{D}_{t,\mathrm{odd}})$, and

$$p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}}, \mathcal{D}_{t,\mathrm{even}}) = p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}}).$$

Step 4: The SV posterior of the odd GP. By Lemma 1 and Step 2, $\mathcal{D}_{t,\mathrm{odd}}$ consists of observations of $\nu_{\mathrm{odd}} \sim \mathcal{GP}(0, k_{\xi,\mathrm{odd}})$ at $\mathbf{X}_{t,\mathrm{odd}}$ with i.i.d. Gaussian noise of variance $\sigma_\epsilon^2/2$. Standard GP regression (Rasmussen & Williams, 2006) and the linearity of the SVs in the value function (Section 2.1) yield $\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}} \sim \mathcal{N}_p(\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_{t,\mathrm{odd}}}, \boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_{t,\mathrm{odd}}})$ with

$$\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_{t,\mathrm{odd}}} = \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{X}_{t,\mathrm{odd}})\,\widetilde{\mathbf{K}}_{\mathrm{odd}}^{-1}\mathbf{Y}_{t,\mathrm{odd}}, \tag{22}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_{t,\mathrm{odd}}} = \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top - \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{X}_{t,\mathrm{odd}})\,\widetilde{\mathbf{K}}_{\mathrm{odd}}^{-1}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{X}_{t,\mathrm{odd}}, \mathbf{Z})\mathbf{A}^\top, \tag{23}$$

where $\widetilde{\mathbf{K}}_{\mathrm{odd}} := \mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{X}_{t,\mathrm{odd}}, \mathbf{X}_{t,\mathrm{odd}}) + \frac{\sigma_\epsilon^2}{2}\mathbf{I}$. Combining Steps 1, 3, and 4 gives $p(\boldsymbol{\phi} \mid \mathcal{D}_t) = p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}}, \mathcal{D}_{t,\mathrm{even}}) = p(\boldsymbol{\phi} \mid \mathcal{D}_{t,\mathrm{odd}})$ with the right-hand side given by Equations (22) and (23), which proves the claim. □

B.4 Odd Expected Information Gain

Theorem 3 (EIG equivalence under paired acquisition; stated with i.i.d. Gaussian observation noise as in Appendix B.3). *Under the assumptions of Theorem 2, let* $\mathbf{z} \in \{0,1\}^p$ *be a candidate coalition and let* $y_{\mathrm{odd}}(\mathbf{z}) := \frac{1}{2}(y(\mathbf{z}) - y(\bar{\mathbf{z}}))$ *denote the odd target obtained from evaluating both* $\mathbf{z}$ *and* $\bar{\mathbf{z}}$*. Then*

$$I\big(\boldsymbol{\phi}; y(\mathbf{z}), y(\bar{\mathbf{z}}) \mid \mathcal{D}_t\big) = I\big(\boldsymbol{\phi}; y_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_t\big) = I\big(\boldsymbol{\phi}; y_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_{t,\mathrm{odd}}\big) =: \mathrm{EIG}^{(t)}_{\boldsymbol{\phi},\mathrm{odd}}(\mathbf{z}),$$

where the last expression is the EIG under the odd GP conditioned on the odd dataset. Moreover, $\mathrm{EIG}^{(t)}_{\boldsymbol{\phi},\mathrm{odd}}(\mathbf{z}) = \mathrm{EIG}^{(t)}_{\boldsymbol{\phi},\mathrm{odd}}(\bar{\mathbf{z}})$.

Proof. For the first claim, the invertible map of Step 1 in Appendix B.3 gives $I(\boldsymbol{\phi}; y(\mathbf{z}), y(\bar{\mathbf{z}}) \mid \mathcal{D}_t) = I(\boldsymbol{\phi}; y_{\mathrm{odd}}(\mathbf{z}), y_{\mathrm{even}}(\mathbf{z}) \mid \mathcal{D}_t)$, since mutual information is invariant under invertible transformations of its arguments (Cover & Thomas, 2006). By the chain rule of mutual information (Cover & Thomas, 2006),

$$\begin{aligned} &I\big(\boldsymbol{\phi}; y_{\mathrm{odd}}(\mathbf{z}), y_{\mathrm{even}}(\mathbf{z}) \mid \mathcal{D}_t\big) \\ &\quad = I\big(\boldsymbol{\phi}; y_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_t\big) + I\big(\boldsymbol{\phi}; y_{\mathrm{even}}(\mathbf{z}) \mid y_{\mathrm{odd}}(\mathbf{z}), \mathcal{D}_t\big). \end{aligned}$$

By Step 3 of the proof of Theorem 2, and since Step 2 of that proof applies verbatim to the noise of the candidate pair, $y_{\mathrm{even}}(\mathbf{z})$ and $\mathcal{D}_{t,\mathrm{even}}$ are functions of $(\nu_{\mathrm{even}}, \epsilon_{\mathrm{even}})$, which is independent of $(\nu_{\mathrm{odd}}, \epsilon_{\mathrm{odd}})$ and hence of $(\boldsymbol{\phi}, y_{\mathrm{odd}}(\mathbf{z}), \mathcal{D}_{t,\mathrm{odd}})$. Thus the second term vanishes, which proves the first equality. For the second equality, note that $(\boldsymbol{\phi}, y_{\mathrm{odd}}(\mathbf{z}))$ is a function of $(\nu_{\mathrm{odd}}, \epsilon_{\mathrm{odd}})$ and the fresh noise of the candidate pair, all of which are independent of $\mathcal{D}_{t,\mathrm{even}}$. Conditioning on $\mathcal{D}_t$, i.e., on $(\mathcal{D}_{t,\mathrm{odd}}, \mathcal{D}_{t,\mathrm{even}})$ by Step 1 of that proof, therefore yields the same joint law of $(\boldsymbol{\phi}, y_{\mathrm{odd}}(\mathbf{z}))$ as conditioning on $\mathcal{D}_{t,\mathrm{odd}}$ alone, and the latter is the posterior of the odd GP by Step 4 of that proof.

For the second claim, recall from Equations (7) and (8) that the EIG depends on the candidate only through its posterior predictive variance and through $\mathrm{Cov}(\boldsymbol{\phi}, \nu_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_{t,\mathrm{odd}})$, the latter entering quadratically via $Q_t(\mathbf{z})$. Since $k_{\xi,\mathrm{odd}}$ is odd in each argument (proof of Lemma B.1), $\mathbf{K}_{\xi,\mathrm{odd}}(\bar{\mathbf{z}}, \mathbf{X}_{t,\mathrm{odd}}) = -\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{z}, \mathbf{X}_{t,\mathrm{odd}})$ and $k_{\xi,\mathrm{odd}}(\bar{\mathbf{z}}, \bar{\mathbf{z}}) = k_{\xi,\mathrm{odd}}(\mathbf{z}, \mathbf{z})$, so that by Equation (15) $\mathrm{Var}(\nu_{\mathrm{odd}}(\bar{\mathbf{z}}) \mid \mathcal{D}_{t,\mathrm{odd}}) = \mathrm{Var}(\nu_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_{t,\mathrm{odd}})$ and $\mathrm{Cov}(\boldsymbol{\phi}, \nu_{\mathrm{odd}}(\bar{\mathbf{z}}) \mid \mathcal{D}_{t,\mathrm{odd}}) = -\mathrm{Cov}(\boldsymbol{\phi}, \nu_{\mathrm{odd}}(\mathbf{z}) \mid \mathcal{D}_{t,\mathrm{odd}})$, hence $Q_t(\bar{\mathbf{z}}) = Q_t(\mathbf{z})$ and the EIG values coincide. □

B.5 Covariance Identities

Proposition B.2 (Covariance identities)**.** *Under the assumptions of Lemma 1, for all* $\mathbf{z} \in \{0,1\}^p$*,*

$$\mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{z}) = \mathbf{A}\mathbf{K}_{\xi}(\mathbf{Z}, \mathbf{z}) \qquad \textit{and} \qquad \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top = \mathbf{A}\mathbf{K}_{\xi}(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top.$$

Proof. For the first identity, write $\nu(\mathbf{z}) = \nu_{\mathrm{odd}}(\mathbf{z}) + \nu_{\mathrm{even}}(\mathbf{z})$ and use $\mathbf{A}\boldsymbol{\nu} = \mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z})$ (Step 3 in Appendix B.3) together with Lemma 1:

$$\begin{aligned} \mathbf{A}\mathbf{K}_{\xi}(\mathbf{Z}, \mathbf{z}) &= \mathrm{Cov}\left(\mathbf{A}\boldsymbol{\nu}, \nu(\mathbf{z})\right) = \mathrm{Cov}\left(\mathbf{A}\boldsymbol{\nu}, \nu_{\mathrm{odd}}(\mathbf{z}) + \nu_{\mathrm{even}}(\mathbf{z})\right) \\ &= \mathrm{Cov}\left(\mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z}), \nu_{\mathrm{odd}}(\mathbf{z})\right) + \mathrm{Cov}\left(\mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z}), \nu_{\mathrm{even}}(\mathbf{z})\right) = \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{z}) + \mathbf{0}. \end{aligned}$$

For the second, analogously,

$$\mathbf{A}\mathbf{K}_{\xi}(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top = \mathrm{Cov}\left(\mathbf{A}\boldsymbol{\nu}, \mathbf{A}\boldsymbol{\nu}\right) = \mathrm{Cov}\left(\mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z}), \mathbf{A}\nu_{\mathrm{odd}}(\mathbf{Z})\right) = \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top.$$

□

B.6 Computational Consequences of the Odd GP

By Theorem 2, the SV posterior of the odd GP has the same form as Equations (4) and (5) (see Equations (22) and (23) in Appendix B.3), but with the training kernel matrix $\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{X}_{t,\mathrm{odd}}, \mathbf{X}_{t,\mathrm{odd}}) \in \mathbb{R}^{n\times n}$, the targets $\mathbf{Y}_{t,\mathrm{odd}} \in \mathbb{R}^n$, and the cross-covariance $\mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z}, \mathbf{X}_{t,\mathrm{odd}}) \in \mathbb{R}^{p\times n}$ in place of their counterparts with t rows. Since $n = t/2$, this reduces the cost of the Cholesky decomposition, which scales cubically in the number of rows, by a factor of up to 8, that of the triangular solves, which scale quadratically, by a factor of 4, and that of the cross-covariance computation, which

scales linearly, by a factor of 2. By Theorem 3, the same reduction applies to all EIG quantities involving the training data (Equations (7) and (8)). In addition, since the odd EIG is invariant under complements, only one representative per candidate pair has to be scored, which halves the number of candidates $|\mathcal{C}|$ and hence the size of all matrices over the candidate set, so that the t-dependent terms of the per-iteration EIG cost $\mathcal{O}(t^3 + p^4 + |\mathcal{C}|t^2)$ of Section 2.3 shrink by a factor of up to 8, while each iteration acquires two evaluations, which halves the number of iterations for a given budget.

Moreover, the prior covariances involving the Shapley transformation need not be computed specifically for the odd kernel but coincide with those of the full kernel,

$$\mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z},\mathbf{z}) = \mathbf{A}\mathbf{K}_{\xi}(\mathbf{Z},\mathbf{z}) \qquad \text{and} \qquad \mathbf{A}\mathbf{K}_{\xi,\mathrm{odd}}(\mathbf{Z},\mathbf{Z})\mathbf{A}^{\top} = \mathbf{A}\mathbf{K}_{\xi}(\mathbf{Z},\mathbf{Z})\mathbf{A}^{\top} \tag{24}$$

for all $\mathbf{z} \in \{0,1\}^p$ (Proposition B.2 in Appendix B.5). Computing them specifically for the odd kernel would require expanding it as the difference of two Hamming kernels, $k_{\xi,\mathrm{odd}}(\mathbf{z},\mathbf{z}') = \frac{1}{2}(k_{\xi}(\mathbf{z},\mathbf{z}') - k_{\xi}(\mathbf{z},\bar{\mathbf{z}}'))$, and thus double their cost.

C Proofs and Details for Section 3.2

Throughout this section, the hyperparameters ξ are fixed. We write $\alpha_j = 1$ and $\beta_j = \exp(-1/\ell_j)$ for the match and mismatch factors of the Hamming kernel (Equation (16)); all statements hold verbatim for any product kernel with positive factors $\alpha_j, \beta_j > 0$. The quadrature route was introduced for explaining tree-based models by TreeGrad (Li et al., 2026), extended to Shapley interactions by Quadrature-TreeSHAP (Wettenstein et al., 2026), and to general product games, including product kernel methods, by QuadraSHAP (Mohammadi et al., 2026).

C.1 Shapley Weights as Beta Integrals

Lemma C.3 (Beta representation of the Shapley weights; see, e.g., Andrews et al., 1999; Owen, 1972)**.** *For $s \in \{0, \dots, p-1\}$,*

$$\frac{1}{p\binom{p-1}{s}} = \int_0^1 \tau^s (1-\tau)^{p-1-s}\, d\tau.$$

The right-hand side is the Beta function $B(s+1, p-s) = s!\,(p-1-s)!/p!$ (Andrews et al., 1999); its use for the Shapley weights goes back to the multilinear extension of Owen (1972). The entries of the Shapley transformation (Equation (1)) involve two such weights, depending on whether the player is contained in the coalition. The following lemma expresses both through a single integrand that is a product over the remaining players.

Lemma C.4 (Leave-one-out product form; Owen, 1972)**.** *For a player $\ell \in P$, a coalition $R \subseteq P$, and $\tau \in [0,1]$, define the per-player factor $w_\tau(r, R) := \tau \mathbb{1}[r \in R] + (1-\tau)\mathbb{1}[r \notin R]$. Then*

$$\int_0^1 \prod_{r \neq \ell} w_\tau(r, R)\, d\tau = \begin{cases} \dfrac{1}{p\binom{p-1}{|R|-1}}, & \ell \in R, \\ \dfrac{1}{p\binom{p-1}{|R|}}, & \ell \notin R, \end{cases}$$

and consequently the entry of $\mathbf{A}$ for player i and coalition S can be written as

$$\mathbf{A}_{i,S} = \big(\mathbb{1}[i \in S] - \mathbb{1}[i \notin S]\big) \int_0^1 \prod_{r \neq i} w_\tau(r, S)\, d\tau. \tag{25}$$

This is the multilinear extension formula of Owen (1972), $\phi_i(\nu) = \int_0^1 \partial_i f(\tau, \dots, \tau)\, d\tau$ with $f(\mathbf{x}) := \sum_{S \subseteq P} \prod_{r \in S} x_r \prod_{r \notin S} (1 - x_r)\, \nu(S)$, written entrywise for $\mathbf{A}$ via Lemma C.3.

C.2 The Hamming Kernel as a Product Game

By Equation (2), the Hamming kernel is a product kernel, $k_\xi(\mathbf{z}, \mathbf{z}') = \sigma_\nu^2 \prod_{j \in P} (\alpha_j \mathbb{1}[z_j = z'_j] + \beta_j \mathbb{1}[z_j \neq z'_j])$ with $\alpha_j = 1$ and $\beta_j = \exp(-1/\ell_j)$. Fix a coalition $\mathbf{z} \in \{0,1\}^p$ and consider the game $\nu_{\mathbf{z}}(S) = k_\xi(\mathbf{1}_S, \mathbf{z})$ from Section 3.2, i.e., the kernel value between $\mathbf{z}$ and the indicator vector $\mathbf{1}_S$ of a coalition $S \subseteq P$. Player j contributes α_j if $(\mathbf{1}_S)_j = z_j$ and β_j otherwise. Defining

$$\gamma_j(\mathbf{z}) := \begin{cases} \alpha_j, & z_j = 0, \\ \beta_j, & z_j = 1, \end{cases} \qquad \delta_j(\mathbf{z}) := \begin{cases} \beta_j, & z_j = 0, \\ \alpha_j, & z_j = 1, \end{cases}$$

i.e., the factor of player j if $j \notin S$ and if $j \in S$, respectively, we obtain

$$\nu_{\mathbf{z}}(S) = k_\xi(\mathbf{1}_S, \mathbf{z}) = \sigma_\nu^2 \prod_{j \notin S} \gamma_j(\mathbf{z}) \prod_{j \in S} \delta_j(\mathbf{z}). \tag{26}$$

Since $\alpha_j = 1$, we have $\gamma_j(\mathbf{z}) = \beta_j^{z_j}$ and $\delta_j(\mathbf{z}) = \beta_j^{1-z_j}$, so that Equation (26) reads

$$k_\xi(\mathbf{1}_S, \mathbf{z}) = \sigma_\nu^2 \prod_{j \notin S} \beta_j^{z_j} \prod_{j \in S} \beta_j^{1-z_j} = \sigma_\nu^2 \Big(\prod_{j \in P} \beta_j^{z_j} \Big) \prod_{j \in S} \beta_j^{1-2z_j}. \tag{27}$$

Factoring out the constant $\sigma_\nu^2 \prod_{j\in P} \gamma_j(\mathbf{z}) = \sigma_\nu^2 \prod_{j\in P} \beta_j^{z_j} = k_\xi(\mathbf{0}, \mathbf{z})$ (for $S = \emptyset$, every player is a non-member) leaves the product game $S \mapsto \prod_{j\in S} \delta_j(\mathbf{z})/\gamma_j(\mathbf{z}) = \prod_{j\in S} \beta_j^{1-2z_j}$; in words, adding player j to a coalition removes a mismatch with $\mathbf{z}$ if $z_j = 1$ and adds one otherwise. Hence, $\nu_\mathbf{z}$ is a product game in the sense of QuadraSHAP (Mohammadi et al., 2026) up to the constant $k_\xi(\mathbf{0}, \mathbf{z}) > 0$, which is the product-kernel observation of Mohammadi et al. (2025), and since the SVs are linear in the value function, the results for product games apply after rescaling. In the derivations below, we keep the non-member factors $\gamma_j(\mathbf{z})$ explicit, which avoids the division and keeps all factors in $(0, 1]$. Note that $\alpha_j, \beta_j > 0$ implies $\gamma_j(\mathbf{z}), \delta_j(\mathbf{z}) > 0$, so that all factors $\gamma_j(\mathbf{z})(1-\tau) + \delta_j(\mathbf{z})\,\tau$ appearing below are strictly positive for $\tau \in [0, 1]$.

C.3 The Linear Term

The linear term. Equation (11) is the product-game integral of TreeGrad, Quadrature-TreeSHAP, and QuadraSHAP (Li et al., 2026; Wettenstein et al., 2026; Mohammadi et al., 2026) applied to the kernel game of Appendix C.2; we restate it in two equivalent forms and, for completeness, re-derive it via Lemma C.4. For every coalition $\mathbf{z} \in \{0, 1\}^p$ and player $i \in P$,

$$\begin{aligned}\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})\big)_i &= \sigma_\nu^2\,(1 - 2z_i)(\beta_i - 1) \int_0^1 \prod_{j\neq i} \big(1 - (1-\beta_j)\,|\tau - z_j|\big)\,d\tau \\ &= \sigma_\nu^2\,\big(\delta_i - \gamma_i\big) \int_0^1 \prod_{j\neq i} \big(\gamma_j(1-\tau) + \delta_j\,\tau\big)\,d\tau, \end{aligned} \tag{28}$$

where $\gamma_j = \gamma_j(\mathbf{z}) = \beta_j^{z_j}$ and $\delta_j = \delta_j(\mathbf{z}) = \beta_j^{1-z_j}$ are the factors from Appendix C.2. The first line is the form stated in the main text; we prove the second, equivalent form, which keeps the non-member factors γ_j explicit, and convert it at the end.

Proof. The i-th entry of $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})$ is the SV of player i for the game $\nu_\mathbf{z}(S) = k_\xi(\mathbf{1}_S, \mathbf{z})$, i.e., by Equations (25) and (26),

$$\begin{aligned}\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{z})\big)_i &= \sum_{S\subseteq P} \mathbf{A}_{i,S}\, k_\xi(\mathbf{1}_S, \mathbf{z}) \\ &= \sigma_\nu^2 \sum_{S\subseteq P} \big(\mathbb{1}[i \in S] - \mathbb{1}[i \notin S]\big) \int_0^1 \prod_{r\neq i} w_\tau(r, S)\,d\tau \prod_{j\notin S} \gamma_j \prod_{j\in S} \delta_j.\end{aligned}$$

Since the sum is finite, we may exchange it with the integral. For fixed τ, we sum over the membership of each player separately. Player i contributes δ_i if $i \in S$ and γ_i if $i \notin S$, weighted by the signs $+1$ and -1, and does not appear in $\prod_{r\neq i} w_\tau(r, S)$; every other player $j \neq i$ contributes $\tau\,\delta_j$ if $j \in S$ and $(1-\tau)\,\gamma_j$ if $j \notin S$. Hence the sum over all S factorizes into

$$\sum_{S\subseteq P} \big(\mathbb{1}[i \in S] - \mathbb{1}[i \notin S]\big) \prod_{r\neq i} w_\tau(r, S) \prod_{j\notin S} \gamma_j \prod_{j\in S} \delta_j = \big(\delta_i - \gamma_i\big) \prod_{j\neq i} \big(\gamma_j(1-\tau) + \delta_j\,\tau\big),$$

and integrating over τ gives the claim. □

Finally, $\gamma_j(1-\tau) + \delta_j\,\tau = \beta_j^{z_j}(1-\tau) + \beta_j^{1-z_j}\,\tau = 1 - (1-\beta_j)\,|\tau - z_j|$ for $z_j \in \{0, 1\}$ and $\tau \in [0, 1]$, and $\delta_i - \gamma_i = \beta_i^{1-z_i} - \beta_i^{z_i} = (1 - 2z_i)(\beta_i - 1)$, which turns Equation (28) into Equation (11). Dividing every factor by $\gamma_j > 0$ instead recovers the product-game form of QuadraSHAP (Mohammadi et al., 2026) with factors $\beta_j^{1-2z_j}$, scaled by $k_\xi(\mathbf{0}, \mathbf{z})$.

C.4 The Quadratic Term

Theorem 4 (Quadratic term). *Let $h_r(\tau, \tau') := 1 - (1-\beta_r)(\tau + \tau' - 2\tau\tau')$. Then, for $i \neq j$, $(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top)_{i,j} = \sigma_\nu^2(1-\beta_i)(1-\beta_j) \iint (1-2\tau)(1-2\tau') \prod_{r\notin\{i,j\}} h_r(\tau, \tau')\,d\tau\,d\tau'$, and $(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top)_{i,i} = 2\sigma_\nu^2(1-\beta_i) \iint \prod_{r\neq i} h_r(\tau, \tau')\,d\tau\,d\tau'$.* We prove the statement for a general product kernel with factors α_r for agreement and β_r for disagreement of player r, i.e., with $h_r(\tau, \tau') := \alpha_r[(1-\tau)(1-\tau') + \tau\,\tau'] + \beta_r[\tau(1-\tau') + (1-\tau)\tau']$ and prefactors $(\beta_i - \alpha_i)(\beta_j - \alpha_j)$ and $2(\alpha_i - \beta_i)$; the Hamming kernel has $\alpha_r = 1$ (Equation (2)), which gives the form above.

Proof. We proceed in five steps: we write the entry as a double sum over coalition pairs, represent both Shapley weights as integrals (Lemma C.4), exchange summation and integration, collapse the sum over the players other than i and j into a product, and finally evaluate the contributions of i and j.

Step 1: The entry as a double sum. The (i, j) entry of $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$ is the prior covariance between the SVs of players i and j,

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top\big)_{i,j} = \sum_{S,T\subseteq P} \mathbf{A}_{i,S}\, \mathbf{A}_{j,T}\, k_\xi(\mathbf{1}_S, \mathbf{1}_T),$$

where, by the product form of the kernel, player r contributes α_r to $k_\xi(\mathbf{1}_S, \mathbf{1}_T)$ if r is in both or in neither of S and T, and β_r if r is in exactly one of them.

Step 2: Both Shapley weights as integrals. By Equation (25), applied to (i, S) with integration variable τ and to (j, T) with integration variable τ',

$$\begin{aligned}\mathbf{A}_{i,S}\, \mathbf{A}_{j,T} = \big(\mathbb{1}[i \in S] - \mathbb{1}[i \notin S]\big)\big(\mathbb{1}[j \in T] - \mathbb{1}[j \notin T]\big) \\ \times \iint_{[0,1]^2} \prod_{r\neq i} w_\tau(r, S) \prod_{r\neq j} w_{\tau'}(r, T)\, d\tau\, d\tau'.\end{aligned}$$

Step 3: Exchanging summation and integration. Substituting into Step 1 and exchanging the finite sum with the integrals,

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top\big)_{i,j} = \sigma_\nu^2 \iint_{[0,1]^2} \sum_{S,T\subseteq P} \sigma_i(S)\, \sigma_j(T) \prod_{r\neq i} w_\tau(r, S) \prod_{r\neq j} w_{\tau'}(r, T)\, \tilde{k}(S, T)\, d\tau\, d\tau', \tag{29}$$

where $\sigma_i(S) := \mathbb{1}[i \in S] - \mathbb{1}[i \notin S]$ and $\tilde{k}(S, T) := k_\xi(\mathbf{1}_S, \mathbf{1}_T)/\sigma_\nu^2$. For fixed (τ, τ'), every factor in the summand is a product over players, so the sum over all coalition pairs (S, T) factorizes into a product over players of sums over the four membership states $(\mathbb{1}[r \in S], \mathbb{1}[r \in T]) \in \{0, 1\}^2$.

Step 4: The free players. Consider $r \notin \{i, j\}$. Its membership in S and T is unconstrained, and it contributes $w_\tau(r, S)$, $w_{\tau'}(r, T)$, and its kernel factor. Summing over the four states,

$$\underbrace{\tau\, \tau'\, \alpha_r}_{r\in S,\, r\in T} + \underbrace{\tau(1-\tau')\, \beta_r}_{r\in S,\, r\notin T} + \underbrace{(1-\tau)\, \tau'\, \beta_r}_{r\notin S,\, r\in T} + \underbrace{(1-\tau)(1-\tau')\, \alpha_r}_{r\notin S,\, r\notin T} = h_r(\tau, \tau').$$

This factor is the same for all four cases of the memberships of i and j, so every summand of Equation (29) contains the common product $\prod_{r\notin\{i,j\}} h_r(\tau, \tau')$.

Step 5a: The players i and j for $i \neq j$. Player i is excluded from $\prod_{r\neq i} w_\tau(r, S)$ but appears in $\prod_{r\neq j} w_{\tau'}(r, T)$ (as $i \neq j$) and in the kernel. For fixed membership of i in S, summing over its membership in T gives $\tau'\, \alpha_i + (1 - \tau')\, \beta_i$ if $i \in S$ (the two states $i \in T$ and $i \notin T$ contribute α_i and β_i) and $\tau'\, \beta_i + (1 - \tau')\, \alpha_i$ if $i \notin S$. Weighting by $\sigma_i(S) = \pm 1$ and adding,

$$\big(\tau'\, \alpha_i + (1 - \tau')\, \beta_i\big) - \big(\tau'\, \beta_i + (1 - \tau')\, \alpha_i\big) = (\beta_i - \alpha_i)(1 - 2\tau').$$

By symmetry, player j contributes $(\beta_j - \alpha_j)(1 - 2\tau)$. Multiplying with the common product of Step 4 and integrating gives Equation (12).

Step 5b: The diagonal $i = j$. Now the free players are all $r \neq i$ and contribute $\prod_{r\neq i} h_r(\tau, \tau')$ as in Step 4. Player i is excluded from both weight products and contributes only its kernel factor, weighted by $\sigma_i(S)\sigma_i(T)$:

$$\underbrace{\alpha_i}_{i\in S,\, i\in T}\ \underbrace{-\beta_i}_{i\in S,\, i\notin T}\ \underbrace{-\beta_i}_{i\notin S,\, i\in T}\ \underbrace{+\alpha_i}_{i\notin S,\, i\notin T} = 2(\alpha_i - \beta_i).$$

Multiplying with the common product and integrating gives Equation (13). □

C.5 Exact Gauss–Legendre Quadrature

Corollary 5 (Cost and exactness of GL quadrature). *For $M \geq \lceil p/2 \rceil$ GL nodes, the quadrature rules for Equations* (11) *to* (13) *are exact; with $M = \lceil p/2 \rceil$, the computation costs $\mathcal{O}(pM) = \mathcal{O}(p^2)$ per coalition and $\mathcal{O}(p^2M^2) = \mathcal{O}(p^4)$ for the quadratic term, i.e., the same asymptotic costs as the ESP route of* `ShaplEIG`.

Proof. The integrands of Equations (11) to (13) are polynomials of degree at most $p-1$ in each integration variable (products of $p-1$ affine factors, or of $p-2$ bilinear factors and $(1-2\tau)(1-2\tau')$), and the GL rule with M nodes integrates polynomials of degree up to $2M-1$ exactly, in each variable of a tensor-product rule separately (Golub & Welsch, 1969); hence exactness holds if and only if $2M-1 \geq p-1$, i.e., $M \geq \lceil p/2 \rceil$. The costs follow from the leave-one-out and leave-two-out products of Appendix C.6: $\mathcal{O}(pM)$ per coalition and $\mathcal{O}(p^2M^2)$ for the quadratic term. □

For $M < \lceil p/2 \rceil$, the rules are approximations. For the one-dimensional integral in Equation (11), the GL error decays geometrically in M (Mohammadi et al., 2026; Trefethen, 2008) (Section 3.2); for the bivariate integrals in Equations (12) and (13), we do not analyze the error and examine the accuracy attained with a fixed M empirically in Section 4.

C.6 Implementation

Leave-one-out products. For a batch of coalitions $\mathbf{z}$ (training coalitions or candidates), the GL nodes τ_q and weights ω_q are computed once (Golub & Welsch, 1969). For each $\mathbf{z}$, we form the $p \times M$ table $H_{j,q} := \gamma_j(\mathbf{z})(1-\tau_q) + \delta_j(\mathbf{z})\tau_q > 0$ and its column sums in log space, $L_q := \sum_j \log H_{j,q}$. By the standard log-space trick, the leave-one-out product is $\prod_{j \neq i} H_{j,q} = \exp(L_q - \log H_{i,q})$, so that by Corollary 5

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{z})\big)_i = \sigma_\nu^2\,(\delta_i - \gamma_i)\sum_{q=1}^{M} \omega_q \exp\big(L_q - \log H_{i,q}\big),$$

i.e., one $p \times M$ matrix times the weight vector per coalition, which is batched over coalitions into a single matrix product. To avoid overflow and underflow for large p, the exponent is shifted by its maximum over i per node, and the shift is folded into the weights. No division by a potentially small factor and no cancellation occurs, since all $H_{j,q}$ are positive.

Leave-two-out products as a single matrix product. For the quadratic term, we flatten the node grid into $G := M^2$ node pairs $g = (q, q')$ and form the $p \times G$ table $H_{r,g} := h_r(\tau_q, \tau_{q'}) > 0$ with log column sums $L_g := \sum_r \log H_{r,g}$. The key observation is that the leave-two-out product factorizes into two leave-one-out terms,

$$\prod_{r \notin \{i,j\}} H_{r,g} = \exp\big(L_g - \log H_{i,g} - \log H_{j,g}\big) = V_{i,g}\,V_{j,g}, \qquad V_{r,g} := \exp\big(\tfrac{1}{2}L_g - \log H_{r,g}\big).$$

With $d_i := \beta_i - \alpha_i$ and the node-pair weights $\tilde{\omega}_g := \omega_q\omega_{q'}(1-2\tau_q)(1-2\tau_{q'})$, the off-diagonal part of Equation (12) therefore reads

$$\big(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top\big)_{i,j} = \sigma_\nu^2\,d_i\,d_j\sum_{g=1}^{G}\tilde{\omega}_g\,V_{i,g}\,V_{j,g},$$

$$\text{i.e.,}\qquad \mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top\big|_{i\neq j} = \sigma_\nu^2\,\big(\mathbf{d}\mathbf{d}^\top\big) \odot \big(V\,\mathrm{diag}(\tilde{\boldsymbol{\omega}})V^\top\big),$$

i.e., a single matrix product of a $p \times G$ with a $G \times p$ matrix, where $\odot$ denotes the elementwise product. The diagonal is computed separately from Equation (13) with weights $\omega_q\omega_{q'}$ and the leave-one-out products $\exp(L_g - \log H_{i,g})$, in $\mathcal{O}(pM^2)$. Per node pair, the exponents are again shifted by their maximum over r; the shifts are folded into $\tilde{\omega}_g$. Since the node pairs are independent, the table can be processed in chunks, which bounds the memory by the chunk size instead of pM^2. The overall cost is $\mathcal{O}(p^2M^2)$ for the matrix product plus $\mathcal{O}(pM^2)$ for the tables, and the computation is dominated by a single dense matrix product, which is highly parallel on standard hardware.

C.7 Comparison with the ESP Route of ShaplEIG

The quadrature route folds the value function into the integral representation of the Shapley weights and evaluates the resulting integrand at the quadrature nodes. Coefficient-based schemes, in contrast, do not involve any integral: they compute the sums of value function evaluations over all coalitions of each size as the coefficients of a generating polynomial, using ESPs, and subsequently contract these coefficients with the size-dependent Shapley weights. `ShaplEIG` computes the two terms of Section 3.2 in this way (Rundel et al., 2026). For the linear term, it forms the coefficients of the generating polynomial $\prod_j(\gamma_j + \delta_j\zeta)$, whose coefficient of ζ^k equals the sum of kernel values over all coalitions of size k, by a degree-by-degree dynamic program over the players in $\mathcal{O}(p^2)$, then removes the factor of player i by polynomial division and contracts the resulting coefficients with the size-dependent Shapley weights. For the quadratic term, it maintains bivariate coefficient tables via prefix and suffix recursions over the players and evaluates 16 bilinear forms per pair of players, in $\mathcal{O}(p^4)$. The two routes are dual representations of the same polynomials: the ESP route works in the coefficient representation and applies the weights afterwards, which is why it must resolve every coalition size separately; the quadrature route works in the point-value representation at the GL nodes, into which the weights are folded via Lemma C.3, so that every player contributes a single scalar per node and the product over players collapses to one scalar. This has three consequences. First, the recursions of the ESP route are inherently sequential over players (and, for the prefix and suffix tables, over the leave-out index), whereas the node evaluations of the quadrature route are independent and reduce to matrix products (Appendix C.6). Second, at the exact node count $M = \lceil p/2 \rceil$ both routes have the same asymptotic cost, $\mathcal{O}(p^2)$ per coalition and $\mathcal{O}(p^4)$ for the quadratic term, so the gain of the quadrature route in this regime is one of parallelism and constant factors, not of asymptotics. Third, the ESP route has no natural approximation: all coefficients are required exactly, whereas the quadrature route admits $M < \lceil p/2 \rceil$ at the price of an approximation error (geometrically decaying for the one-dimensional integral in Equation (11); see Appendix C.5), which decouples the cost from the number of players.

D Details for Algorithm 1

D.1 Comparison with ShaplEIG

Table 1 contrasts `ShapleyOG` (Algorithm 1) with `ShaplEIG` (Algorithm 1 in Rundel et al., 2026) along all aspects in which the two algorithms differ.

Table 1: Algorithmic differences between `ShaplEIG` and `ShapleyOG`.

	`ShaplEIG`	`ShapleyOG`
Surrogate	Full GP with constant mean on all t observations	Odd GP with zero mean on the $n = t/2$ odd targets (Section 3.1)
Hyperparameters	MAP estimate on all t observations; refit in every iteration (staged schedule for large p)	MAP estimate on the n odd targets; refit at $t = T_0$ and whenever t is a power of two (Appendix D.3)
Prior covariances	ESP: theoretical $\mathcal{O}(p^2)$ per coalition, implemented $\mathcal{O}(p^3)$; $\mathcal{O}(p^4)$ for the SV covariance	GL: $\mathcal{O}(pM)$ per coalition and $\mathcal{O}(p^2M^2)$; exact for $M \geq \lceil p/2 \rceil$ (Section 3.2 and Appendix D.2)
Selection	One coalition per iteration; EIG of a single evaluation	One complement pair per iteration; joint EIG of both evaluations, scored via one representative (Theorem 3)
Candidate set	Fixed set provided as input; depleted over the run	Current experiments: fixed paired pool; proposed refresh variant: $\min\{10t, 2^p - t\}$ coalitions at each refit (Appendix D.3)
Between refits	Reuse of $\mathbf{K}_\xi(\mathbf{X}_t, \mathbf{X}_t)$, $\mathbf{AK}_\xi(\mathbf{Z}, \mathbf{X}_t)$, and $\mathbf{AK}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$; factorization and all solves recomputed	Bordered Cholesky, extended half-solves, and rank-one EIG updates, with periodic and numerical rebuilds (Appendix D.4)
Iterations for budget T	$T - T_0$	$(T - T_0)/2$

D.2 Quantities, Sizes, and Costs

We write r for the number of training rows of the surrogate and c for the number of scored candidates, i.e., $r = t$ and $c = |\mathcal{C}|$ for `ShaplEIG` and $r = n = t/2$ and $c = |\mathcal{C}|/2$ for `ShapleyOG`, where $\mathcal{C}$ denotes a paired candidate set of which one representative per pair is scored. Let $\mathbf{X}$ denote the training coalitions (one representative per pair for the odd GP), $\mathbf{Y}$ the corresponding targets, $\mathbf{K}_\xi$ the (odd) kernel matrix, and $\widetilde{\mathbf{K}_\xi}(\mathbf{X}, \mathbf{X}) := \mathbf{K}_\xi(\mathbf{X}, \mathbf{X}) + \sigma_\epsilon^2\mathbf{I}$. The SV posterior and the EIG for all candidates $\mathbf{z} \in \mathcal{C}$ (Equations (4), (5), (7) and (8); see Appendix A.2 for the written-out quantities) are assembled in three groups:

1. *Prior term* (depends on ξ only): $\mathbf{AK}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$ via GL quadrature (Corollary 5).
2. *Training terms*: the Cholesky factorization $\widetilde{\mathbf{K}_\xi}(\mathbf{X}, \mathbf{X}) = \mathbf{L}\mathbf{L}^\top$, the cross-covariance $\mathbf{AK}_\xi(\mathbf{Z}, \mathbf{X})$ via GL quadrature, and the half-solve $\mathbf{Y}_\mathbf{X} := \mathbf{L}^{-1}(\mathbf{AK}_\xi(\mathbf{Z}, \mathbf{X}))^\top \in \mathbb{R}^{r \times p}$. Then $\mathbf{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t} = \mathbf{AK}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top - \mathbf{Y}_\mathbf{X}^\top\mathbf{Y}_\mathbf{X}$ and $\boldsymbol{\mu}_{\boldsymbol{\phi}|\mathcal{D}_t} = \mathbf{Y}_\mathbf{X}^\top\mathbf{L}^{-1}\mathbf{Y}$.
3. *Candidate terms*: $\mathbf{AK}_\xi(\mathbf{Z}, \mathcal{C})$ via GL quadrature, the candidate kernel matrix $\mathbf{K}_\xi(\mathbf{X}, \mathcal{C})$, and the half-solve $\mathbf{Y}_\mathcal{C} := \mathbf{L}^{-1}\mathbf{K}_\xi(\mathbf{X}, \mathcal{C}) \in \mathbb{R}^{r \times c}$. Then the cross-covariances $\mathrm{Cov}(\boldsymbol{\phi}, \nu(\mathbf{z}) \mid \mathcal{D}) = \mathbf{AK}_\xi(\mathbf{Z}, \mathbf{z}) - \mathbf{Y}_\mathbf{X}^\top\mathbf{Y}_\mathcal{C}[:, \mathbf{z}]$ for all candidates form the matrix $\mathbf{B} \in \mathbb{R}^{p \times c}$, the predictive variances are $\mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}) = k_\xi(\mathbf{z}, \mathbf{z}) - \|\mathbf{Y}_\mathcal{C}[:, \mathbf{z}]\|^2$, and $Q_t(\mathbf{z}) = \mathbf{B}[:, \mathbf{z}]^\top \mathbf{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t}^{-1} \mathbf{B}[:, \mathbf{z}]$ is obtained from a Cholesky factor of $\mathbf{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t}$ and one triangular solve with c right-hand sides.

Table 2 separates a fresh computation from an ordinary update at fixed hyperparameters. The training-row count is halved under the odd GP: $r = t/2$. For a complement-closed pool of C coalitions, only $c = C/2$ representatives are scored. Thus the cubic training factorization and the

candidate half-solve have respectively $1/8$ of their full-GP operation counts; the training half-solve has $1/4$. These are savings in individual terms, not a uniform eightfold runtime guarantee. For the odd GP, the effective observation-noise variance in the training kernel is half the variance of an individual game observation.

The Implemented Coefficient Baseline. The theoretical $\mathcal{O}(p^2)$ per-coalition ESP bound stated in Section 2.3 must be distinguished from the coefficient routine used in the code comparison. For each player j, that routine explicitly convolves a prefix with $j+1$ coefficients and a suffix with $p-j$ coefficients. Across all players, its arithmetic count per coalition is:

$$\sum_{j=0}^{p-1}(j+1)(p-j) = \frac{p(p+1)(p+2)}{6} = \Theta(p^3). \tag{30}$$

This is the route in both the legacy implementation and the current coefficient baseline. The routine is cubic because it forms every leave-one-out polynomial explicitly; the $\mathcal{O}(p^2)$ bound of Rundel et al. (2026) instead removes the factor of each player by synthetic division, which Mohammadi et al. (2025) report to be numerically unstable beyond a few dozen players and which the implementation of `ShaplEIG` avoids for this reason. A stable $\mathcal{O}(p^2)$ coefficient construction, which contracts the suffix polynomials with the Shapley weights before combining them with the prefix polynomials, was proposed only recently (Mohammadi et al., 2025) and is not part of the implementation compared here. The gap between the coefficient and the quadrature arm in Appendix F.4 is thus measured against a $\Theta(p^3)$ routine; a comparison against the stable $\mathcal{O}(p^2)$ construction is left to future work.

Table 2: Quantities for EIG and final SV prediction, with r training rows and c scored candidates. The last column describes an ordinary accepted update at fixed hyperparameters; numerical rebuilds have the additional costs stated above. Bracketed costs describe the implemented coefficient baseline (Equation (30)).

Quantity	Size	From scratch	Update between refits
$\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top$	$p \times p$	$\mathcal{O}(p^2M^2)$ [$\mathcal{O}(p^4)$]	reused
$\widetilde{\mathbf{K}_\xi}(\mathbf{X},\mathbf{X})$	$r \times r$	$\mathcal{O}(r^2p)$	one row and column, $\mathcal{O}(rp)$
$\mathbf{L}$ (Cholesky)	$r \times r$	$\mathcal{O}(r^3)$	one row (bordered), $\mathcal{O}(r^2)$
$\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X})$	$p \times r$	$\mathcal{O}(rpM)$ [$\mathcal{O}(rp^3)$]	one column, $\mathcal{O}(pM)$
$\mathbf{Y}_\mathbf{X} = \mathbf{L}^{-1}(\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}))^\top$	$r \times p$	$\mathcal{O}(r^2p)$	one row, $\mathcal{O}(rp)$
$\mathbf{L}^{-1}\mathbf{Y}$	r	$\mathcal{O}(r^2)$	not needed for selection
$\boldsymbol{\mu}_{\phi\mid\mathcal{D}_t} = \mathbf{Y}_\mathbf{X}^\top\mathbf{L}^{-1}\mathbf{Y}$	p	$\mathcal{O}(rp)$	final readout only
$\boldsymbol{\Sigma}_{\phi\mid\mathcal{D}_t} = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{Z})\mathbf{A}^\top - \mathbf{Y}_\mathbf{X}^\top\mathbf{Y}_\mathbf{X}$	$p \times p$	$\mathcal{O}(rp^2)$	rank one, $\mathcal{O}(p^2)$
$\boldsymbol{\Sigma}^{-1}_{\phi\mid\mathcal{D}_t}$	$p \times p$	$\mathcal{O}(p^3)$	Sherman–Morrison, $\mathcal{O}(p^2)$
$\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathcal{C})$	$p \times c$	$\mathcal{O}(cpM)$ [$\mathcal{O}(cp^3)$]	one column removed
$\mathbf{K}_\xi(\mathbf{X},\mathcal{C})$	$r \times c$	$\mathcal{O}(rcp)$	one row, $\mathcal{O}(cp)$; column removed
$\mathbf{Y}_\mathcal{C} = \mathbf{L}^{-1}\mathbf{K}_\xi(\mathbf{X},\mathcal{C})$	$r \times c$	$\mathcal{O}(r^2c)$	one row, $\mathcal{O}(rc)$
$\mathbf{B}$ (cross-covariances)	$p \times c$	$\mathcal{O}(rpc)$	rank one, $\mathcal{O}(pc)$
$\mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D})$, $\mathbf{z} \in \mathcal{C}$	c	$\mathcal{O}(rc)$	one subtraction each, $\mathcal{O}(c)$
$\mathbf{L}_\phi^{-1}\mathbf{B}$	$p \times c$	$\mathcal{O}(p^2c)$	only at score rebuilds
$Q_t(\mathbf{z})$, $\mathbf{z} \in \mathcal{C}$	c	$\mathcal{O}(pc)$ after solve	rank one, $\mathcal{O}(p^2+pc)$
$\mathrm{EIG}^{(t)}_{\phi,\mathrm{odd}}(\mathbf{z})$, $\mathbf{z} \in \mathcal{C}$	c	$\mathcal{O}(c)$	$\mathcal{O}(c)$ from updated ratios
MAP objective and gradient (per eval.)		$\mathcal{O}(r^3+r^2p)$	no refit

D.3 Refit Schedule and Candidate Refresh

Refit Schedule. The first fit uses T_0 observations, subsequent selection-time fits occur at powers of two, and a final fit uses all T observations. The first interval need not be a doubling if T_0 is not a power of two. Thereafter, a stretch from t to $2t$ observations contains $t/2$ paired selections and $t/2-1$ incremental score updates before the next fit. When T_0 and T are powers of two, the number of fits including the final one is $\log_2(T/T_0)+1$, for example 7 at $T_0 = 64$ and $T =$

4096. With J objective/gradient evaluations per fit, the sum of the cubic fit costs is proportional to $\sum_{s\in\mathcal{R}\cup\{T\}} J(s/2)^3 = \mathcal{O}(JT^3)$, whereas refitting at every paired selection gives $\mathcal{O}(JT^4)$. The quadratic-in-training-size gradient terms similarly sum to $\mathcal{O}(JpT^2)$ and $\mathcal{O}(JpT^3)$. These counts concern fitting work; they do not imply the same reduction in total runtime, which also includes every selection and game evaluation.

Candidate Refresh. At every refit, `ShapleyOG` draws a new candidate set of coalitions that have not been evaluated yet, in complement pairs, and scores one representative per pair. In the experiments, the candidate set has a fixed size of 10240 coalitions (4096 in the ablations) and is drawn with the paired coalition sampler of `shapiq` with uniform weights over coalition sizes; if fewer unevaluated coalitions remain, all of them are used (Appendix E.2). Between refits, selected pairs are removed from the set. Since a doubling stretch from t to $2t$ observations selects at most $t/2$ pairs, the set is never depleted for the budgets considered. Refreshing changes the candidate search domain and may change the selected design; it has no general multiplicative speedup guarantee.

D.4 Incremental EIG Updates

Fix the hyperparameters ξ and consider the state after a refit or a previous update: the training kernel matrix $\widetilde{\mathbf{K}_\xi} := \widetilde{\mathbf{K}_\xi}(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{r\times r}$ with Cholesky factor $\mathbf{L}$, $\widetilde{\mathbf{K}_\xi} = \mathbf{L}\mathbf{L}^\top$, and the half-solves $\mathbf{Y}_\mathbf{X} = \mathbf{L}^{-1}\mathbf{R}_\mathbf{X}$ and $\mathbf{Y}_\mathcal{C} = \mathbf{L}^{-1}\mathbf{R}_\mathcal{C}$ with right-hand sides $\mathbf{R}_\mathbf{X} := (\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X}))^\top \in \mathbb{R}^{r\times p}$ and $\mathbf{R}_\mathcal{C} := \mathbf{K}_\xi(\mathbf{X},\mathcal{C}) \in \mathbb{R}^{r\times c}$. A new observation at $\mathbf{z}^+$ (the representative of the selected pair) appends the row $\mathbf{z}^+$ to $\mathbf{X}$. Write $\mathbf{k} := \mathbf{K}_\xi(\mathbf{X},\mathbf{z}^+) \in \mathbb{R}^r$ for its kernel vector against the existing rows and $\kappa := k_\xi(\mathbf{z}^+,\mathbf{z}^+) + \sigma_\epsilon^2$ for its diagonal entry.

Bordered Cholesky Factorization. Appending one row and column to a Cholesky factorization is the last step of the column-by-column Cholesky recursion (Golub & Van Loan, 2013, Sec. 4.2.5, Eq. (4.2.9)): the extended training kernel matrix and its Cholesky factor are

$$\widetilde{\mathbf{K}_\xi}^+ = \begin{pmatrix} \widetilde{\mathbf{K}_\xi} & \mathbf{k} \\ \mathbf{k}^\top & \kappa \end{pmatrix} = \mathbf{L}^+(\mathbf{L}^+)^\top, \qquad \mathbf{L}^+ := \begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{l}^\top & d \end{pmatrix}, \quad \mathbf{l} := \mathbf{L}^{-1}\mathbf{k}, \quad d := \sqrt{\kappa - \mathbf{l}^\top\mathbf{l}}, \tag{31}$$

Here, $d^2 = \kappa - \mathbf{k}^\top\widetilde{\mathbf{K}_\xi}^{-1}\mathbf{k}$ is the Schur complement of $\widetilde{\mathbf{K}_\xi}$ in $\widetilde{\mathbf{K}_\xi}^+$, i.e., the posterior predictive variance of the new observation, which is positive for positive definite $\widetilde{\mathbf{K}_\xi}^+$ and bounded below by the noise term σ_ϵ^2 unless $\mathbf{z}^+$ duplicates a training row, which the selection from not yet evaluated candidates excludes. As $\mathbf{L}^+$ is lower triangular with positive diagonal, it is *the* Cholesky factor of $\widetilde{\mathbf{K}_\xi}^+$. Computing it requires one forward substitution for $\mathbf{l}$, i.e., $\mathcal{O}(r^2)$ instead of $\mathcal{O}(r^3)$ for a new factorization.

Extension of the Half-Solves. Let $\mathbf{R}^+ := \begin{pmatrix} \mathbf{R} \\ \boldsymbol{\rho}^\top \end{pmatrix}$ be a right-hand side extended by one row $\boldsymbol{\rho}^\top \in \mathbb{R}^{1\times m}$, i.e., $\boldsymbol{\rho} = \mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{z}^+)$ for $\mathbf{R}_\mathbf{X}$ (the new column of $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z},\mathbf{X})$, one GL evaluation) and $\boldsymbol{\rho}^\top = \mathbf{K}_\xi(\mathbf{z}^+,\mathcal{C})$ for $\mathbf{R}_\mathcal{C}$ (the new row of the candidate kernel matrix). Then

$$\mathbf{Y}^+ := (\mathbf{L}^+)^{-1}\mathbf{R}^+ = \begin{pmatrix} \mathbf{Y} \\ (\boldsymbol{\rho}^\top - \mathbf{l}^\top\mathbf{Y})/d \end{pmatrix} \qquad \text{with } \mathbf{Y} = \mathbf{L}^{-1}\mathbf{R}, \tag{32}$$

by forward substitution (Golub & Van Loan, 2013, Sec. 3.1.1). Thus, all previously computed rows of $\mathbf{Y}$ remain valid and only the new row is computed, in $\mathcal{O}(rm)$, i.e., in $\mathcal{O}(rp)$ for $\mathbf{Y}_\mathbf{X}$ and $\mathcal{O}(rc)$ for $\mathbf{Y}_\mathcal{C}$, instead of $\mathcal{O}(r^2m)$ for a new triangular solve. This is the same property that makes forward substitution row-sequential: the i-th row of $\mathbf{L}^{-1}\mathbf{R}$ depends only on the first i rows of $\mathbf{L}$ and $\mathbf{R}$.

Derived Quantities. Let $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^c$ be the transposes of the new rows of $\mathbf{Y}_\mathbf{X}$ and $\mathbf{Y}_\mathcal{C}$, after removing the selected candidate. Writing $\mathbf{b}_\mathbf{z}$ for column $\mathbf{z}$ of $\mathbf{B}$, Gaussian conditioning on the new observation (Rasmussen & Williams, 2006) gives:

$$\begin{aligned} \boldsymbol{\Sigma}^+_{\boldsymbol{\phi}|\mathcal{D}_t} &= \boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t} - \mathbf{u}\mathbf{u}^\top, & \mathbf{b}^+_\mathbf{z} &= \mathbf{b}_\mathbf{z} - \mathbf{u}v_\mathbf{z}, \\ \mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}^+) &= \mathrm{Var}(\nu(\mathbf{z}) \mid \mathcal{D}) - v_\mathbf{z}^2. \end{aligned} \tag{33}$$

An inverse update also makes it possible to update each EIG quadratic form directly. For $\mathbf{h} := \boldsymbol{\Sigma}^{-1}_{\boldsymbol{\phi}|\mathcal{D}_t}\mathbf{u}$, $a := \mathbf{u}^\top\mathbf{h}$, and $\tau := 1 - a > 0$, the Sherman–Morrison formula (Sherman & Morrison, 1950) gives

$$(\boldsymbol{\Sigma}^{+}_{\boldsymbol{\phi}|\mathcal{D}_t})^{-1} = \boldsymbol{\Sigma}^{-1}_{\boldsymbol{\phi}|\mathcal{D}_t} + \frac{\mathbf{h}\mathbf{h}^\top}{\tau}. \tag{34}$$

Substituting Equations (33) and (34) into $Q^+_t(\mathbf{z}) = (\mathbf{b}^+_{\mathbf{z}})^\top(\boldsymbol{\Sigma}^+_{\boldsymbol{\phi}|\mathcal{D}_t})^{-1}\mathbf{b}^+_{\mathbf{z}}$ yields:

$$\begin{aligned} Q^+_t(\mathbf{z}) &= (\mathbf{b}_{\mathbf{z}} - \mathbf{u}v_{\mathbf{z}})^\top \left(\boldsymbol{\Sigma}^{-1}_{\boldsymbol{\phi}|\mathcal{D}_t} + \frac{\mathbf{h}\mathbf{h}^\top}{\tau}\right)(\mathbf{b}_{\mathbf{z}} - \mathbf{u}v_{\mathbf{z}}) \\ &= Q_t(\mathbf{z}) - 2v_{\mathbf{z}}\mathbf{b}_{\mathbf{z}}^\top\mathbf{h} + av_{\mathbf{z}}^2 + \frac{(\mathbf{b}_{\mathbf{z}}^\top\mathbf{h} - av_{\mathbf{z}})^2}{\tau}. \end{aligned} \tag{35}$$

The shared vector $\mathbf{h}$ costs $\mathcal{O}(p^2)$ and all inner products $\mathbf{b}_{\mathbf{z}}^\top\mathbf{h}$ cost $\mathcal{O}(pc)$. Thus Equation (35) updates all quadratic forms in $\mathcal{O}(p^2+pc)$, without a new SV-covariance factorization or a triangular solve for every candidate. Combining them with the updated predictive variances gives the new EIG scores. The observed odd targets are needed for the next hyperparameter fit and the final posterior mean; they do not enter these fixed-hyperparameter covariance updates.

Exactness and Numerical Behavior. The preceding identities are exact in real arithmetic; floating-point updates require safeguards. The implementation rebuilds $\boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t}$, $\mathbf{B}$, candidate variances, the covariance inverse, and quadratic forms from the extended half-solves every 32 incremental steps. It also rebuilds when the SV covariance required jitter, the inverse update is poorly conditioned, a score or variance fails a consistency check, or the two largest scores are nearly tied. For example, it rejects the inverse update when $\tau \le 100\epsilon$ or $\epsilon\|\boldsymbol{\Sigma}_{\boldsymbol{\phi}|\mathcal{D}_t}\|_1\|\boldsymbol{\Sigma}^{-1}_{\boldsymbol{\phi}|\mathcal{D}_t}\|_1/\tau > 10^{-10}$, where ϵ is machine precision. A nonpositive or nonfinite Schur complement in Equation (31) triggers a fresh training factorization and fresh half-solves. The periodic score rebuild normally preserves the extended training factor and half-solves; it does not refit the hyperparameters.

E EXPERIMENTAL DETAILS

In the following, we provide further details on the experimental setup of Section 4.

E.1 GAMES

Table 3: Overview of the games considered in the experiments. p denotes the number of players, i.e., features, image patches, or tokens. Data sources are OpenML (Bischl et al., 2025) with task IDs in parentheses, the UCI repository (Kelly et al., 2023), the SHAP library (Lundberg & Lee, 2017), and the `shapiq` benchmark suite (Muschalik et al., 2024). Licenses refer to the data.

Dataset	Task	Model	p	Source	License
Vision and language datasets (Muschalik et al., 2024)					
Language (IMDB)	Classification	DistilBERT	14	shapiq / HuggingFace	Public Domain
ResNet18-SP	Image classification	ResNet18	14	shapiq	Public Domain
ViT4by4	Image classification	ViT-B	16	shapiq	Public Domain
DINOv2-16	Image classification	DINOv2-B + linear head	16	ImageNet-1k val. / HuggingFace	ImageNet Terms of Access
Tabular datasets (Štrumbelj & Kononenko, 2010; Muschalik et al., 2024)					
Forest Fires	Regression	TabPFN	13	UCI	CC-BY 4.0
AdultCensus	Classification	TabPFN	14	OpenML (1590)	CC-BY 4.0
Estate	Regression	TabPFN	15	UCI	CC-BY 4.0
Crime	Regression	LightGBM	101	SHAP	CC-BY 4.0
TabArena datasets (Erickson et al., 2025)					
Seismic Bumps	Classification	TabPFN	15	OpenML (46956)	CC-BY 4.0
Churn	Classification	TabPFN	19	OpenML (46915)	MIT
Credit-G	Classification	TabPFN	20	OpenML (46918)	CC-BY 4.0
Bankruptcy	Classification	LightGBM	64	OpenML (46950)	CC-BY 4.0
Superconductivity	Regression	LightGBM	81	OpenML (46961)	CC-BY 4.0
NATICUSdroid	Classification	LightGBM	86	OpenML (46969)	CC-BY 4.0
MIC	Classification	LightGBM	111	OpenML (46980)	CC-BY 4.0
QSAR TID11	Classification	LightGBM	1024	OpenML (46953)	Public Domain

Table 3 gives an overview of the games. They fall into two families. The first consists of precomputed local explanation games: explanations of image classifiers (ResNet18 (He et al., 2016), ViT (Dosovitskiy et al., 2021), DINOv2 (Oquab et al., 2024)) and of a DistilBERT (Sanh et al., 2019) sentiment classifier, in which the players are image patches or tokens and the value of a coalition is the model output when only the players in the coalition are retained, and explanations of predictions of the tabular foundation model TabPFN (Grinsztajn et al., 2026), in which the players are the features of the explained instance and the value of a coalition is the prediction when the features outside the coalition are set to $+\infty$ and thereby treated as missing by TabPFN, so that every coalition costs a single forward pass. The Forest Fires game deviates from this marginalization: it follows the remove-and-contextualize scheme of Rundel et al. (2024), in which the features outside the coalition are removed from both the context and the explained instance before the forward pass, instead of being marked as missing. The vision and language games are taken from the `shapiq` benchmark suite (Muschalik et al., 2024), and the remaining games of this family were precomputed with `shapiq`. The second family consists of local explanations of LightGBM tree ensembles (Ke et al., 2017) trained on tabular datasets, with the features of a single test instance as players. These games are evaluated on the fly, and their exact SVs are obtained via interventional TreeSHAP (Zern et al., 2023), which makes them suitable for benchmarking with many players. For all games, the seed selects the explained instance, while the train-test splits and the explained models are fixed. The tree game QSAR TID11 with $p = 1024$ players is used only in the quadrature study of Appendix E.4. The ImageNet-1k validation images underlying the DINOv2 game are governed by the ImageNet Terms of Access, which permit non-commercial research and educational use only, whereas the DINOv2 weights (`facebook/dinov2-base-imagenet1k-1-layer`) are released under Apache-2.0.

Since the raw payoffs of the games differ in scale by several orders of magnitude, every game is normalized such that $\nu(\emptyset) = 0$ and $\nu(P) = 1$, so that the SVs sum to one by efficiency and squared-error metrics are comparable across games. Game instances with $|\nu(P) - \nu(\emptyset)|$ below a numerical tolerance are degenerate under this normalization and are skipped; the next seed is used instead.

E.2 Configuration of ShapleyOG

`ShapleyOG` uses the odd GP of Section 3.1, i.e., a zero-mean GP with the odd Hamming kernel, fitted to the odd targets, which are rescaled to unit variance. Its kernel hyperparameters, i.e., the lengthscales and the output scale, are learned by MAP estimation with the default lengthscale prior of BoTorch (Balandat et al., 2020), $\ell_j \sim \mathrm{LogNormal}(\sqrt{2}+\frac{1}{2}\log p, \sqrt{3})$, and a minimum lengthscale of 10^{-6}, using L-BFGS-B (Byrd et al., 1995) with up to five restarts in case of failed optimization runs (see Appendix E.6 for further details). For numerical stability, we assume additive zero-mean Gaussian noise with fixed variance 10^{-6} in the standardized units, which yields quasi-noiseless GPs (Appendix A.1).

The border trick initial design of Section 4.1 is deterministic and paired by construction. The candidate set of 10240 coalitions (4096 in the ablation of Appendix E.4) is drawn from the not yet evaluated coalitions with the paired coalition sampler of `shapiq`, which samples coalition sizes uniformly and completes every draw by its complement, or, for the games with at most 16 players, all unevaluated coalitions if fewer remain; one representative per pair is scored.

E.3 Baselines

All non-adaptive baselines are run in their implementations of `shapiq` (Muschalik et al., 2024) version 1.7.0 with the default settings of this version, which we pin explicitly: Kernel SHAP with its kernel weights and without paired sampling; Leverage SHAP with paired sampling and deterministic sample counts; Regression MSR with an XGBoost (Chen & Guestrin, 2016) proxy model and paired sampling; OddSHAP, which is paired by construction, with its default interaction factor; and permutation sampling and SVARM with their own sampling schemes. Every baseline receives exactly the budget at which the estimate is recorded.

E.4 Ablations

The ablation of Section 4.1 that adds the measures of `ShapleyOG` one at a time starts from `ShaplEIG`, which refits the hyperparameters in every iteration, computes the prior covariances via ESPs, recomputes the factorizations in every iteration, and depletes a fixed candidate set. The measures are added in the order of Section 4.1, where paired acquisition is first added under the full GP with the joint EIG of the pair before the odd GP with odd hyperparameter fitting is switched on, which yields `ShapleyOG`. All variants share the implementation of Appendix E.6, so that the ablation isolates the algorithmic measures. It is run on the games ResNet18-SP, Churn, Credit-G, Bankruptcy, and Crime of Table 3 with 10 seeds, evaluation budgets up to 2048, and a candidate set of 4096 coalitions. The quadrature study runs `ShapleyOG` with $M \in \{\lceil p/2 \rceil, 256, 64, 8, 2\}$ nodes and with the ESP route on the tree game QSAR TID11 with $p = 1024$ players, five seeds, evaluation budgets of 4096 and 8192, and a candidate set of 10240 coalitions.

E.5 Metrics

Overhead. The overhead of `ShapleyOG` is the measured time of hyperparameter fitting and EIG maximization; the posterior inference reuses the quantities of the EIG maximization and is not timed separately, and the value function evaluations are not included, since the runner evaluates them outside these steps. The baselines are run in `shapiq`, whose wall-clock time includes their value function evaluations, which is not negligible for the tree games that are evaluated on the fly. We therefore report as the overhead of a baseline its wall-clock time minus the smallest wall-clock time among all baselines on the same game instance and budget. This credits the cheapest baseline with zero overhead and every other baseline with at most its true overhead, so the correction errs, if at all, in favour of the baselines.

Accuracy. For an estimate $\hat{\phi}$ and the exact SVs ϕ, the normalized mean squared error is $\mathrm{NMSE} := \|\hat{\phi} - \phi\|_2^2 / \|\phi\|_2^2$. Since the NMSE differs by orders of magnitude across games and seeds, an arithmetic mean would be dominated by the largest values; we therefore aggregate by the geometric mean, i.e., the mean in log space, across seeds and games, and additionally report the arithmetic mean and the median of the NMSE. The *average rank* of `ShapleyOG` at a fixed evaluation budget is obtained by ranking the seven estimators by NMSE on every game instance, i.e., (game, seed) combination, with rank 1 for the lowest NMSE and tied values receiving the mean of their ranks, and averaging the rank of `ShapleyOG` over the instances; the colours in Figure 1 compare the average ranks of all estimators computed in the same way. The computational overhead is the wall-clock time of an estimator excluding the value function evaluations; for `ShapleyOG`, we report hyperparameter fitting and EIG maximization separately and in total. The total computation time for an assumed cost c per value function evaluation is the overhead plus c times the budget.

E.6 Implementation Details

Beyond the measures of Section 3.3, our implementation, which builds on the publicly available implementation of `ShaplEIG` (Rundel et al., 2026), is more efficient in several further respects, which already explains part of the gap to the overheads reported there. The hyperparameters are fitted with L-BFGS-B (Byrd et al., 1995) in the log-coordinates of the lengthscales and the output scale, with closed-form gradients (Rasmussen & Williams, 2006) and warm starts at the hyperparameters of the previous refit. Hamming kernel matrices are formed via a single matrix product of the binary designs instead of a p-fold mismatch tensor.

E.7 Compute and Reproducibility

All experiments were run on CPU nodes with 32 cores and 64 GB of RAM, using Python 3.12, Torch 2.9.1, GPyTorch 1.14, BoTorch 0.14.0, and `shapiq` 1.7.0. Every run is restricted to a single CPU thread (BLAS, OpenMP, and Torch), so all reported overheads are single-thread wall-clock times. The code, including the configuration files of all experiments and the scripts for preparing the games, is attached to this submission as supplementary material, so that all results are reproducible.

F Additional Experimental Results

This appendix collects further results of the experiments of Section 4; all figures and tables are produced by one script of the code release from the recorded estimates. Appendix F.1 shows the accuracy of every estimator per game (Figures 3 and 4 and Tables 4 and 5), Appendix F.2 breaks the overhead of `ShapleyOG` down into its components (Table 6), Appendix F.3 presents the ablation from `ShaplEIG` to `ShapleyOG` on all five games (Figure 5 and Tables 7 and 8), Appendix F.4 the quadrature study, and Appendix F.5 the average ranks of the strongest competitors (Table 10).

F.1 Per-Game Accuracy

Figures 3 and 4 show the NMSE of all estimators for every game, aggregated over the 30 seeds by the geometric mean and the arithmetic mean, respectively; the y-axis of every panel starts at the NMSE of `ShapleyOG` at 256 evaluations. Tables 4 and 5 list the corresponding numbers for every method at every recorded budget.

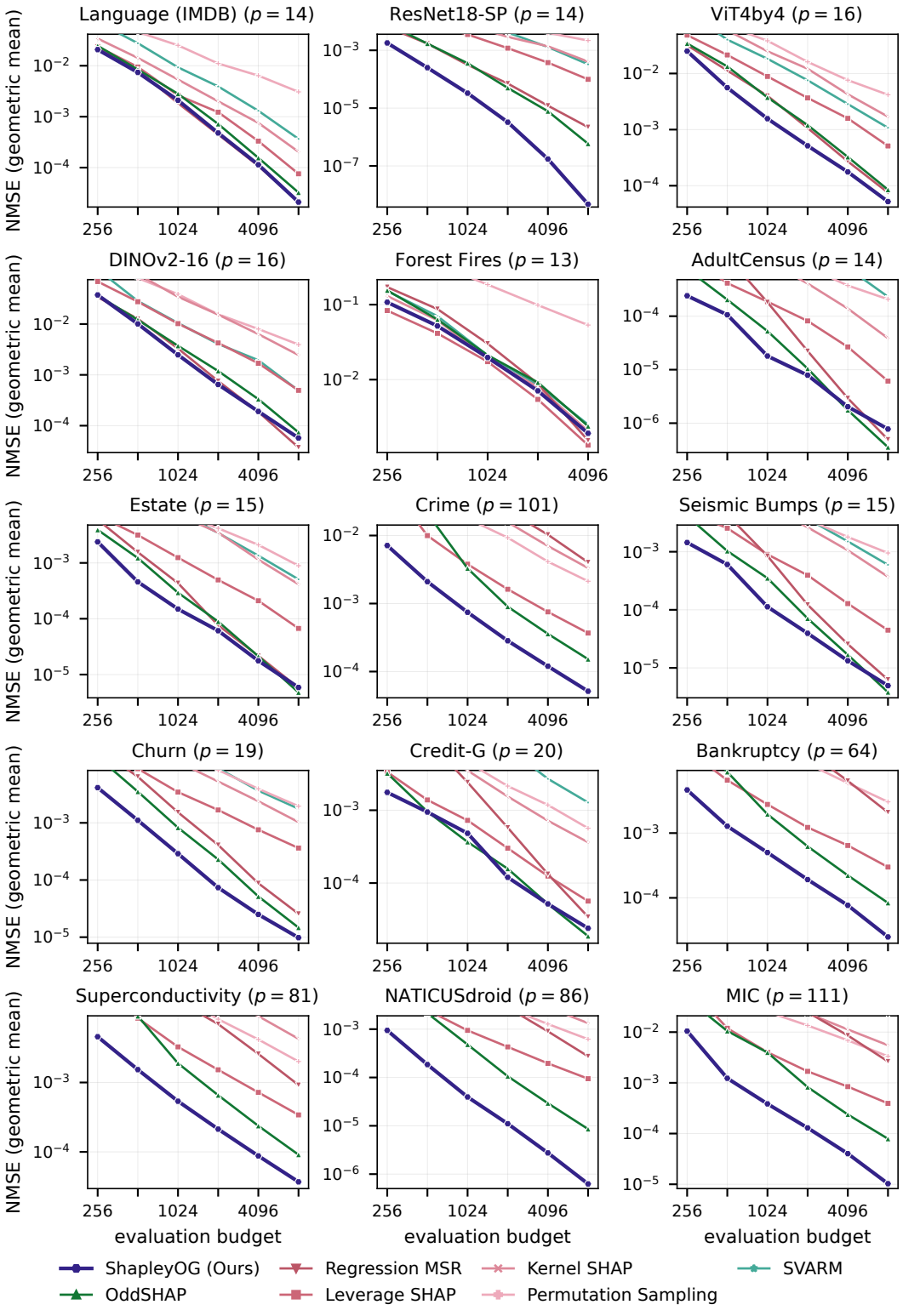


Figure 3: NMSE vs. evaluation budget per game, geometric mean over 30 seeds. The y-axis starts at the NMSE of `ShapleyOG` at 256 evaluations, so estimators starting above it enter the panel later.

Table 4: NMSE per game (geometric mean over 30 seeds) at every recorded budget T; the best method per game and budget is in bold; – marks budgets beyond a game's grid.

Game (p)	ShapleyOG	OddSHAP	Regression MSR	Leverage SHAP	Kernel SHAP	Permutation	SVARM
$T = 256$							
Language (IMDB) (14)	**2.1e-2**	2.5e-2	2.2e-2	2.4e-2	3.4e-2	9.6e-2	7.1e-2
ResNet18-SP (14)	**1.8e-3**	8.2e-3	6.4e-3	2.1e-2	4.5e-2	5.8e-2	5.5e-2
ViT4by4 (16)	**2.5e-2**	3.4e-2	3.2e-2	4.8e-2	1.3e-1	1.6e-1	1.3e-1
DINOv2-16 (16)	3.7e-2	3.6e-2	**3.4e-2**	6.9e-2	2.1e-1	1.6e-1	1.2e-1
Forest Fires (13)	1.1e-1	1.5e-1	1.7e-1	**8.3e-2**	1.3e-1	6.8e-1	1.5e-1
AdultCensus (14)	**2.4e-4**	7.5e-4	5.5e-3	1.3e-3	5.9e-3	7.3e-3	3.8e-2
Estate (15)	**2.4e-3**	3.9e-3	5.9e-3	7.1e-3	4.0e-2	2.4e-2	6.0e-2
Crime (101)	**7.1e-3**	6.6e-2	1.3e-1	6.1e-2	2.0e-1	7.8e-2	5.4e0
Seismic Bumps (15)	**1.4e-3**	4.1e-3	1.8e-2	4.9e-3	4.1e-2	2.9e-2	6.3e-2
Churn (19)	**4.1e-3**	1.4e-2	2.0e-2	1.7e-2	5.9e-2	6.9e-2	1.3e-1
Credit-G (20)	**1.8e-3**	3.2e-3	2.3e-2	3.3e-3	1.8e-2	2.3e-2	1.0e-1
Bankruptcy (64)	**4.6e-3**	4.4e-2	1.6e-1	1.9e-2	5.0e-1	9.6e-2	1.1e0
Superconductivity (81)	**4.6e-3**	2.9e-2	5.3e-2	3.1e-2	2.1e-1	6.9e-2	3.3e0
NATICUSdroid (86)	**9.4e-4**	6.3e-3	3.5e-2	9.2e-3	7.4e-2	2.8e-2	3.1e0
MIC (111)	**1.0e-2**	4.6e-2	2.1e-1	8.7e-2	3.4e-1	1.1e-1	2.1e0
$T = 512$							
Language (IMDB) (14)	**7.4e-3**	8.4e-3	8.2e-3	9.4e-3	1.4e-2	4.6e-2	2.8e-2
ResNet18-SP (14)	**2.5e-4**	1.7e-3	1.7e-3	8.1e-3	2.1e-2	2.7e-2	2.1e-2
ViT4by4 (16)	**5.6e-3**	1.3e-2	1.1e-2	2.1e-2	5.9e-2	7.0e-2	4.0e-2
DINOv2-16 (16)	**1.0e-2**	1.2e-2	1.3e-2	2.8e-2	9.6e-2	8.1e-2	2.8e-2
Forest Fires (13)	5.2e-2	6.3e-2	8.7e-2	**4.1e-2**	6.2e-2	3.4e-1	7.0e-2
AdultCensus (14)	**1.1e-4**	2.0e-4	1.3e-3	4.2e-4	2.5e-3	3.8e-3	1.6e-2
Estate (15)	**4.6e-4**	1.2e-3	1.6e-3	3.2e-3	1.6e-2	1.6e-2	1.9e-2
Crime (101)	**2.1e-3**	2.1e-2	8.2e-2	1.0e-2	7.7e-2	3.4e-2	4.2e0
Seismic Bumps (15)	**6.0e-4**	1.0e-3	4.4e-3	2.5e-3	1.6e-2	1.4e-2	2.3e-2
Churn (19)	**1.1e-3**	3.5e-3	6.4e-3	8.9e-3	2.2e-2	3.6e-2	4.8e-2
Credit-G (20)	**9.5e-4**	9.6e-4	9.3e-3	1.4e-3	8.0e-3	1.1e-2	3.8e-2
Bankruptcy (64)	**1.3e-3**	8.8e-3	9.2e-2	6.5e-3	2.2e-1	4.9e-2	5.8e-1
Superconductivity (81)	**1.5e-3**	9.0e-3	2.9e-2	8.5e-3	9.1e-2	4.0e-2	2.1e0
NATICUSdroid (86)	**1.9e-4**	2.1e-3	1.8e-2	2.5e-3	3.1e-2	1.0e-2	2.0e0
MIC (111)	**1.2e-3**	1.1e-2	1.2e-1	1.2e-2	1.2e-1	5.7e-2	1.5e0
$T = 1024$							
Language (IMDB) (14)	2.1e-3	2.8e-3	**1.8e-3**	2.6e-3	5.4e-3	2.5e-2	9.3e-3
ResNet18-SP (14)	**3.3e-5**	3.6e-4	3.1e-4	3.5e-3	7.7e-3	1.5e-2	9.6e-3
ViT4by4 (16)	**1.6e-3**	3.7e-3	4.0e-3	8.9e-3	2.5e-2	3.8e-2	1.8e-2
DINOv2-16 (16)	**2.5e-3**	3.7e-3	3.3e-3	1.0e-2	3.4e-2	3.9e-2	1.0e-2
Forest Fires (13)	2.0e-2	2.1e-2	3.0e-2	**1.7e-2**	1.9e-2	1.9e-1	2.1e-2
AdultCensus (14)	**1.8e-5**	5.3e-5	1.8e-4	1.9e-4	1.0e-3	1.8e-3	5.7e-3
Estate (15)	**1.5e-4**	2.9e-4	4.3e-4	1.2e-3	7.2e-3	7.7e-3	8.3e-3
Crime (101)	**7.4e-4**	3.3e-3	4.5e-2	3.8e-3	3.4e-2	1.8e-2	1.6e0
Seismic Bumps (15)	**1.1e-4**	3.5e-4	8.5e-4	9.0e-4	6.3e-3	6.8e-3	9.0e-3
Churn (19)	**2.9e-4**	8.2e-4	1.5e-3	3.5e-3	1.2e-2	1.9e-2	2.0e-2
Credit-G (20)	4.8e-4	**3.7e-4**	2.4e-3	7.3e-4	3.5e-3	4.6e-3	1.5e-2
Bankruptcy (64)	**5.0e-4**	2.0e-3	4.4e-2	2.8e-3	1.0e-1	2.6e-2	1.8e-1
Superconductivity (81)	**5.4e-4**	1.9e-3	1.5e-2	3.3e-3	4.4e-2	1.8e-2	6.0e-1
NATICUSdroid (86)	**3.9e-5**	4.8e-4	8.0e-3	9.4e-4	1.5e-2	6.0e-3	6.2e-1
MIC (111)	**3.8e-4**	4.0e-3	6.2e-2	4.0e-3	5.2e-2	2.5e-2	7.2e-1
$T = 2048$							
Language (IMDB) (14)	4.8e-4	7.2e-4	**4.5e-4**	1.2e-3	2.0e-3	1.1e-2	3.9e-3
ResNet18-SP (14)	**3.3e-6**	5.0e-5	7.1e-5	1.2e-3	2.9e-3	7.8e-3	4.1e-3
ViT4by4 (16)	**5.1e-4**	1.2e-3	1.1e-3	3.7e-3	1.2e-2	1.6e-2	7.6e-3
DINOv2-16 (16)	**6.5e-4**	1.2e-3	7.6e-4	4.2e-3	1.5e-2	1.6e-2	4.1e-3
Forest Fires (13)	7.1e-3	9.2e-3	8.9e-3	**5.5e-3**	8.0e-3	9.9e-2	8.5e-3
AdultCensus (14)	**7.9e-6**	1.0e-5	2.2e-5	8.1e-5	4.1e-4	8.2e-4	2.2e-3
Estate (15)	**6.1e-5**	8.9e-5	7.6e-5	5.0e-4	3.5e-3	4.2e-3	3.4e-3
Crime (101)	**2.8e-4**	9.0e-4	2.3e-2	1.6e-3	1.6e-2	9.3e-3	2.4e-1
Seismic Bumps (15)	**3.9e-5**	7.1e-5	1.2e-4	3.9e-4	2.6e-3	3.6e-3	3.7e-3
Churn (19)	**7.4e-5**	2.3e-4	4.1e-4	1.7e-3	5.3e-3	8.4e-3	9.0e-3
Credit-G (20)	**1.2e-4**	1.6e-4	5.8e-4	3.0e-4	1.5e-3	2.1e-3	7.2e-3
Bankruptcy (64)	**1.9e-4**	6.2e-4	1.8e-2	1.2e-3	4.3e-2	1.2e-2	6.0e-2
Superconductivity (81)	**2.1e-4**	6.6e-4	6.9e-3	1.5e-3	1.8e-2	8.2e-3	1.0e-1
NATICUSdroid (86)	**1.1e-5**	1.1e-4	2.8e-3	4.3e-4	6.0e-3	2.7e-3	1.1e-1
MIC (111)	**1.3e-4**	8.2e-4	2.5e-2	1.7e-3	2.4e-2	1.4e-2	1.9e-1
$T = 4096$							
Language (IMDB) (14)	1.1e-4	1.6e-4	**1.1e-4**	3.3e-4	7.6e-4	6.4e-3	1.3e-3
ResNet18-SP (14)	**1.7e-7**	7.8e-6	1.2e-5	3.8e-4	1.4e-3	4.0e-3	1.3e-3
ViT4by4 (16)	**1.8e-4**	3.3e-4	2.7e-4	1.6e-3	4.3e-3	7.6e-3	2.9e-3
DINOv2-16 (16)	1.9e-4	3.3e-4	**1.9e-4**	1.7e-3	6.3e-3	8.0e-3	1.9e-3
Forest Fires (13)	1.9e-3	2.4e-3	1.5e-3	**1.3e-3**	2.1e-3	5.3e-2	2.5e-3
AdultCensus (14)	2.0e-6	**1.7e-6**	2.9e-6	2.7e-5	1.4e-4	3.7e-4	8.0e-4
Estate (15)	**1.8e-5**	2.1e-5	2.2e-5	2.1e-4	1.1e-3	2.1e-3	1.4e-3
Crime (101)	**1.2e-4**	3.6e-4	1.0e-2	7.5e-4	7.0e-3	4.1e-3	4.9e-2
Seismic Bumps (15)	**1.3e-5**	1.7e-5	2.6e-5	1.3e-4	1.1e-3	1.8e-3	1.5e-3
Churn (19)	**2.5e-5**	5.2e-5	8.8e-5	7.6e-4	2.4e-3	3.9e-3	3.7e-3
Credit-G (20)	5.2e-5	**5.2e-5**	1.3e-4	1.2e-4	7.2e-4	1.2e-3	2.7e-3
Bankruptcy (64)	**7.7e-5**	2.2e-4	6.4e-3	6.4e-4	2.2e-2	5.9e-3	2.5e-2
Superconductivity (81)	**8.8e-5**	2.4e-4	2.6e-3	7.2e-4	9.0e-3	4.2e-3	3.1e-2
NATICUSdroid (86)	**2.7e-6**	2.9e-5	9.0e-4	2.0e-4	2.8e-3	1.3e-3	3.6e-2
MIC (111)	**4.0e-5**	2.4e-4	8.6e-3	8.4e-4	1.1e-2	6.9e-3	5.4e-2
$T = 8192$							
Language (IMDB) (14)	**2.1e-5**	3.2e-5	2.1e-5	7.6e-5	2.0e-4	3.1e-3	3.7e-4
ResNet18-SP (14)	**4.7e-9**	5.9e-7	2.2e-6	9.9e-5	4.1e-4	2.2e-3	3.3e-4
ViT4by4 (16)	**5.2e-5**	8.5e-5	7.5e-5	5.1e-4	1.7e-3	4.2e-3	1.1e-3
DINOv2-16 (16)	5.7e-5	7.4e-5	**3.7e-5**	5.0e-4	2.5e-3	3.9e-3	4.9e-4
Forest Fires (13)	–	–	–	–	–	–	–
AdultCensus (14)	7.8e-7	**3.5e-7**	4.9e-7	6.1e-6	3.9e-5	2.1e-4	2.3e-4
Estate (15)	5.9e-6	**4.8e-6**	5.8e-6	6.7e-5	4.1e-4	8.9e-4	5.1e-4
Crime (101)	**5.1e-5**	1.5e-4	4.0e-3	3.7e-4	3.3e-3	2.1e-3	2.0e-2
Seismic Bumps (15)	4.9e-6	**3.8e-6**	6.3e-6	4.5e-5	3.7e-4	9.5e-4	5.9e-4
Churn (19)	**9.8e-6**	1.5e-5	2.6e-5	3.6e-4	1.0e-3	2.0e-3	1.8e-3
Credit-G (20)	2.4e-5	**1.9e-5**	3.4e-5	5.7e-5	3.6e-4	5.7e-4	1.3e-3
Bankruptcy (64)	**2.5e-5**	8.4e-5	2.1e-3	3.0e-4	1.1e-2	3.0e-3	1.1e-2
Superconductivity (81)	**3.7e-5**	9.1e-5	9.2e-4	3.4e-4	4.3e-3	2.0e-3	1.4e-2
NATICUSdroid (86)	**6.3e-7**	8.6e-6	2.7e-4	9.4e-5	1.3e-3	6.2e-4	1.6e-2
MIC (111)	**1.0e-5**	7.9e-5	2.6e-3	3.9e-4	5.5e-3	3.3e-3	2.3e-2

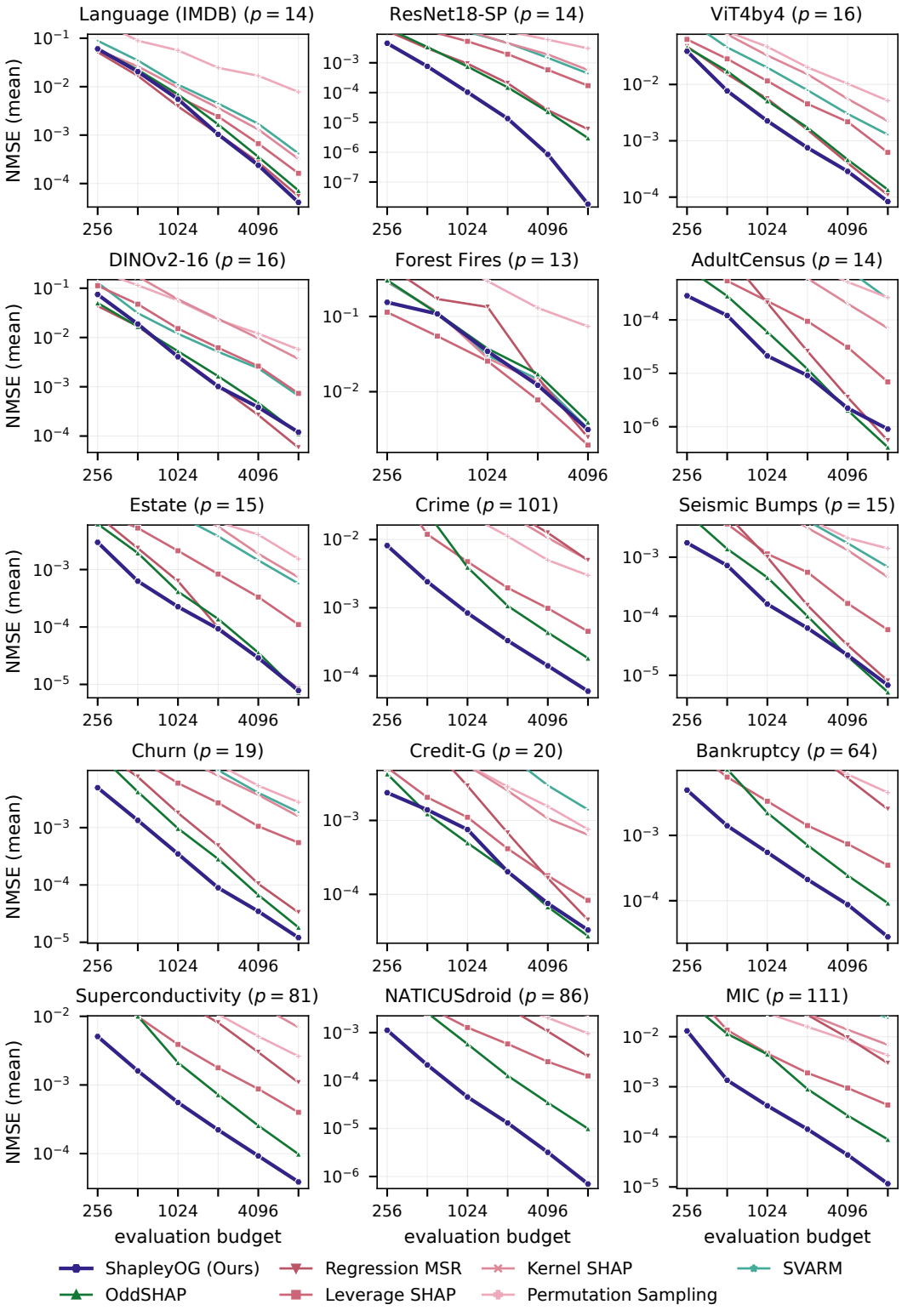


Figure 4: NMSE vs. evaluation budget per game, arithmetic mean over 30 seeds; otherwise as Figure 3.

Table 5: NMSE per game (arithmetic mean over 30 seeds) at every recorded budget T; otherwise as Table 4.

Game (p)	ShapleyOG	OddSHAP	Regression MSR	Leverage SHAP	Kernel SHAP	Permutation	SVARM
$T = 256$							
Language (IMDB) (14)	6.0e-2	6.0e-2	**5.2e-2**	5.6e-2	5.7e-2	2.3e-1	8.9e-2
ResNet18-SP (14)	**4.5e-3**	1.4e-2	1.2e-2	3.2e-2	6.7e-2	1.0e-1	6.1e-2
ViT4by4 (16)	**3.9e-2**	4.6e-2	4.7e-2	6.3e-2	1.7e-1	2.0e-1	1.4e-1
DINOv2-16 (16)	7.5e-2	5.0e-2	**4.3e-2**	1.1e-1	3.4e-1	2.3e-1	1.3e-1
Forest Fires (13)	1.5e-1	3.0e-1	4.8e-1	**1.1e-1**	3.0e-1	2.2e0	2.9e-1
AdultCensus (14)	**2.8e-4**	9.4e-4	7.0e-3	1.7e-3	7.4e-3	9.9e-3	4.5e-2
Estate (15)	**3.0e-3**	6.2e-3	8.7e-3	1.0e-2	9.1e-2	3.3e-2	6.7e-2
Crime (101)	**8.2e-3**	8.1e-2	1.5e-1	8.6e-2	3.0e-1	9.7e-2	7.0e0
Seismic Bumps (15)	**1.8e-3**	5.5e-3	2.1e-2	6.2e-3	4.7e-2	3.7e-2	7.2e-2
Churn (19)	**5.0e-3**	2.0e-2	2.4e-2	3.0e-2	1.0e-1	1.0e-1	1.4e-1
Credit-G (20)	**2.4e-3**	4.3e-3	2.7e-2	5.2e-3	3.0e-2	3.1e-2	1.1e-1
Bankruptcy (64)	**4.9e-3**	5.0e-2	1.8e-1	2.4e-2	6.1e-1	1.1e-1	1.5e0
Superconductivity (81)	**5.1e-3**	3.2e-2	6.4e-2	3.9e-2	3.2e-1	8.3e-2	4.7e0
NATICUSdroid (86)	**1.1e-3**	7.6e-3	4.4e-2	1.2e-2	1.1e-1	4.3e-2	4.2e0
MIC (111)	**1.3e-2**	5.2e-2	2.4e-1	1.0e-1	4.1e-1	1.3e-1	3.1e0
$T = 512$							
Language (IMDB) (14)	2.0e-2	2.2e-2	**1.7e-2**	2.1e-2	2.6e-2	8.9e-2	3.5e-2
ResNet18-SP (14)	**7.6e-4**	3.4e-3	3.0e-3	1.2e-2	3.3e-2	4.9e-2	2.4e-2
ViT4by4 (16)	**7.7e-3**	1.7e-2	1.5e-2	2.8e-2	7.7e-2	8.7e-2	4.6e-2
DINOv2-16 (16)	1.9e-2	**1.7e-2**	1.7e-2	4.7e-2	1.7e-1	1.1e-1	3.1e-2
Forest Fires (13)	1.1e-1	1.1e-1	1.7e-1	**5.5e-2**	1.1e-1	7.1e-1	1.1e-1
AdultCensus (14)	**1.2e-4**	2.8e-4	1.7e-3	5.4e-4	3.3e-3	4.9e-3	1.7e-2
Estate (15)	**6.3e-4**	1.9e-3	2.4e-3	5.3e-3	3.0e-2	2.6e-2	2.1e-2
Crime (101)	**2.4e-3**	2.5e-2	9.9e-2	1.2e-2	1.2e-1	4.2e-2	5.5e0
Seismic Bumps (15)	**7.2e-4**	1.4e-3	5.2e-3	3.7e-3	1.9e-2	1.7e-2	2.6e-2
Churn (19)	**1.3e-3**	4.2e-3	7.7e-3	1.6e-2	3.9e-2	5.0e-2	5.0e-2
Credit-G (20)	1.4e-3	**1.2e-3**	1.2e-2	2.1e-3	1.5e-2	1.4e-2	4.0e-2
Bankruptcy (64)	**1.4e-3**	1.0e-2	1.0e-1	7.8e-3	2.6e-1	7.3e-2	7.4e-1
Superconductivity (81)	**1.6e-3**	1.0e-2	3.6e-2	9.9e-3	1.4e-1	5.0e-2	3.0e0
NATICUSdroid (86)	**2.1e-4**	2.6e-3	2.1e-2	3.4e-3	4.8e-2	1.6e-2	2.7e0
MIC (111)	**1.4e-3**	1.1e-2	1.4e-1	1.4e-2	1.5e-1	7.1e-2	2.3e0
$T = 1024$							
Language (IMDB) (14)	5.5e-3	6.9e-3	**3.9e-3**	6.2e-3	9.6e-3	5.6e-2	1.1e-2
ResNet18-SP (14)	**1.0e-4**	7.5e-4	9.4e-4	5.2e-3	1.2e-2	2.4e-2	1.1e-2
ViT4by4 (16)	**2.3e-3**	5.1e-3	5.6e-3	1.2e-2	3.3e-2	4.7e-2	2.0e-2
DINOv2-16 (16)	**4.1e-3**	5.2e-3	4.3e-3	1.5e-2	5.9e-2	5.6e-2	1.2e-2
Forest Fires (13)	3.4e-2	3.7e-2	1.3e-1	**2.5e-2**	2.8e-2	2.9e-1	3.0e-2
AdultCensus (14)	**2.1e-5**	6.0e-5	2.1e-4	2.3e-4	1.5e-3	2.5e-3	6.1e-3
Estate (15)	**2.3e-4**	4.1e-4	6.3e-4	2.1e-3	1.2e-2	1.1e-2	8.8e-3
Crime (101)	**8.4e-4**	3.9e-3	5.4e-2	4.7e-3	5.2e-2	2.3e-2	2.0e0
Seismic Bumps (15)	**1.6e-4**	4.5e-4	1.0e-3	1.1e-3	8.2e-3	9.1e-3	1.0e-2
Churn (19)	**3.5e-4**	9.7e-4	1.8e-3	6.0e-3	1.9e-2	3.0e-2	2.1e-2
Credit-G (20)	7.5e-4	**5.0e-4**	2.9e-3	1.1e-3	5.5e-3	5.7e-3	1.7e-2
Bankruptcy (64)	**5.5e-4**	2.2e-3	5.0e-2	3.3e-3	1.2e-1	3.8e-2	2.0e-1
Superconductivity (81)	**5.6e-4**	2.1e-3	1.8e-2	3.9e-3	7.5e-2	2.2e-2	7.5e-1
NATICUSdroid (86)	**4.5e-5**	5.8e-4	9.3e-3	1.3e-3	2.3e-2	9.1e-3	7.6e-1
MIC (111)	**4.2e-4**	4.5e-3	6.8e-2	4.6e-3	6.4e-2	3.0e-2	1.0e0
$T = 2048$							
Language (IMDB) (14)	**1.0e-3**	1.7e-3	1.1e-3	2.4e-3	3.6e-3	2.5e-2	4.5e-3
ResNet18-SP (14)	**1.3e-5**	1.5e-4	2.1e-4	1.9e-3	4.6e-3	1.1e-2	4.6e-3
ViT4by4 (16)	**7.5e-4**	1.7e-3	1.6e-3	4.5e-3	1.5e-2	2.0e-2	7.9e-3
DINOv2-16 (16)	**1.0e-3**	1.6e-3	1.1e-3	6.2e-3	2.4e-2	2.3e-2	5.1e-3
Forest Fires (13)	1.2e-2	1.7e-2	1.5e-2	**7.8e-3**	1.4e-2	1.3e-1	1.5e-2
AdultCensus (14)	**9.1e-6**	1.2e-5	2.6e-5	9.4e-5	6.0e-4	1.1e-3	2.4e-3
Estate (15)	**9.3e-5**	1.4e-4	9.8e-5	8.4e-4	5.6e-3	6.7e-3	3.8e-3
Crime (101)	**3.3e-4**	1.1e-3	2.8e-2	2.0e-3	2.4e-2	1.1e-2	2.7e-1
Seismic Bumps (15)	**6.3e-5**	1.0e-4	1.5e-4	5.6e-4	3.1e-3	4.5e-3	4.1e-3
Churn (19)	**8.9e-5**	2.8e-4	4.8e-4	2.7e-3	8.0e-3	1.3e-2	1.0e-2
Credit-G (20)	**2.0e-4**	2.0e-4	6.8e-4	4.1e-4	2.5e-3	2.9e-3	7.8e-3
Bankruptcy (64)	**2.1e-4**	7.1e-4	2.1e-2	1.4e-3	5.2e-2	1.8e-2	6.1e-2
Superconductivity (81)	**2.2e-4**	7.2e-4	8.1e-3	1.8e-3	3.1e-2	1.1e-2	1.0e-1
NATICUSdroid (86)	**1.3e-5**	1.3e-4	3.3e-3	5.8e-4	9.4e-3	4.0e-3	1.1e-1
MIC (111)	**1.4e-4**	9.0e-4	2.7e-2	1.9e-3	2.9e-2	1.6e-2	2.2e-1
$T = 4096$							
Language (IMDB) (14)	**2.4e-4**	3.6e-4	2.8e-4	6.7e-4	1.3e-3	1.7e-2	1.7e-3
ResNet18-SP (14)	**8.4e-7**	2.3e-5	2.6e-5	5.8e-4	1.9e-3	6.0e-3	1.5e-3
ViT4by4 (16)	**2.9e-4**	4.6e-4	4.0e-4	2.2e-3	5.5e-3	1.0e-2	3.0e-3
DINOv2-16 (16)	3.8e-4	4.7e-4	**2.7e-4**	2.6e-3	9.6e-3	1.2e-2	2.4e-3
Forest Fires (13)	3.1e-3	3.9e-3	2.5e-3	**2.0e-3**	3.0e-3	7.4e-2	3.2e-3
AdultCensus (14)	2.2e-6	**2.0e-6**	3.6e-6	3.1e-5	2.0e-4	5.1e-4	8.8e-4
Estate (15)	2.9e-5	3.6e-5	**2.9e-5**	3.3e-4	1.9e-3	4.1e-3	1.5e-3
Crime (101)	**1.4e-4**	4.3e-4	1.3e-2	9.8e-4	1.1e-2	5.0e-3	5.0e-2
Seismic Bumps (15)	2.2e-5	**2.1e-5**	3.2e-5	1.6e-4	1.3e-3	2.1e-3	1.7e-3
Churn (19)	**3.5e-5**	6.7e-5	1.0e-4	1.1e-3	3.7e-3	5.4e-3	4.0e-3
Credit-G (20)	7.5e-5	**6.8e-5**	1.7e-4	1.8e-4	1.1e-3	1.5e-3	3.0e-3
Bankruptcy (64)	**8.7e-5**	2.4e-4	7.5e-3	7.4e-4	2.6e-2	8.6e-3	2.5e-2
Superconductivity (81)	**9.3e-5**	2.6e-4	3.0e-3	8.7e-4	1.5e-2	5.1e-3	3.2e-2
NATICUSdroid (86)	**3.2e-6**	3.5e-5	1.1e-3	2.5e-4	4.7e-3	2.0e-3	3.6e-2
MIC (111)	**4.3e-5**	2.7e-4	9.5e-3	9.4e-4	1.4e-2	8.5e-3	5.5e-2
$T = 8192$							
Language (IMDB) (14)	**4.1e-5**	7.2e-5	5.5e-5	1.6e-4	3.2e-4	7.8e-3	4.2e-4
ResNet18-SP (14)	**1.8e-8**	3.0e-6	5.9e-6	1.7e-4	5.7e-4	3.0e-3	4.4e-4
ViT4by4 (16)	**8.3e-5**	1.4e-4	1.1e-4	6.2e-4	2.2e-3	5.2e-3	1.3e-3
DINOv2-16 (16)	1.2e-4	1.1e-4	**5.8e-5**	7.3e-4	3.7e-3	5.7e-3	6.6e-4
Forest Fires (13)	–	–	–	–	–	–	–
AdultCensus (14)	9.1e-7	**4.1e-7**	5.5e-7	6.9e-6	7.0e-5	2.6e-4	2.5e-4
Estate (15)	7.8e-6	**7.3e-6**	8.4e-6	1.1e-4	7.3e-4	1.5e-3	5.7e-4
Crime (101)	**6.0e-5**	1.8e-4	5.0e-3	4.5e-4	5.0e-3	3.0e-3	2.0e-2
Seismic Bumps (15)	6.8e-6	**5.2e-6**	8.1e-6	5.9e-5	4.6e-4	1.4e-3	6.8e-4
Churn (19)	**1.2e-5**	1.8e-5	3.3e-5	5.5e-4	1.6e-3	2.8e-3	1.9e-3
Credit-G (20)	3.3e-5	**2.7e-5**	4.5e-5	8.3e-5	6.4e-4	7.6e-4	1.4e-3
Bankruptcy (64)	**2.8e-5**	9.2e-5	2.5e-3	3.5e-4	1.3e-2	4.5e-3	1.2e-2
Superconductivity (81)	**3.9e-5**	9.8e-5	1.1e-3	4.0e-4	6.9e-3	2.6e-3	1.5e-2
NATICUSdroid (86)	**7.0e-7**	9.9e-6	3.2e-4	1.3e-4	2.2e-3	9.6e-4	1.6e-2
MIC (111)	**1.2e-5**	8.9e-5	3.0e-3	4.3e-4	6.9e-3	4.2e-3	2.3e-2

F.2 OVERHEAD OF SHAPLEYOG

Table 6 splits the overhead of ShapleyOG reported in Figure 2 into hyperparameter fitting, the EIG maximization at refit steps, where the prior covariances and the factorizations are recomputed for a fresh candidate set, and the incremental EIG updates at all other steps. It also reports the mean time of a single refit step and of a single no-refit step, which shows what the incremental updates save per iteration.

Table 6: Mean overhead of ShapleyOG over 30 seeds for three games of increasing size, split into its components: cumulative times up to the budget T in minutes / seconds (share of the total in parentheses), and the mean time in seconds of one refit step and of one no-refit step.

Game (p)	Component \ T	512	1024	2048	4096	8192
DINOv2-16 (16)	hyperparameter fitting (cum.)	0.0 / 1 (20%)	0.1 / 3 (15%)	0.2 / 10 (14%)	0.7 / 42 (16%)	0.7 / 42 (5%)
	EIG, refit steps (cum.)	0.0 / 0 (6%)	0.0 / 1 (4%)	0.0 / 1 (2%)	0.0 / 3 (1%)	0.1 / 7 (1%)
	EIG, no-refit steps (cum.)	0.1 / 5 (73%)	0.3 / 17 (81%)	1.0 / 59 (84%)	3.7 / 221 (83%)	14.3 / 858 (95%)
	total (cum.)	**0.1 / 7**	**0.3 / 20**	**1.2 / 70**	**4.4 / 265**	**15.1 / 907**
	EIG per refit step (s)	0.11	0.14	0.22	0.39	0.88
	EIG per no-refit step (s)	0.02	0.03	0.06	0.11	0.21
Bankruptcy (64)	hyperparameter fitting (cum.)	0.0 / 1 (12%)	0.1 / 7 (17%)	0.6 / 34 (25%)	3.1 / 186 (35%)	3.1 / 186 (13%)
	EIG, refit steps (cum.)	0.0 / 2 (14%)	0.0 / 3 (7%)	0.1 / 4 (3%)	0.1 / 6 (1%)	0.2 / 12 (1%)
	EIG, no-refit steps (cum.)	0.2 / 9 (75%)	0.5 / 30 (76%)	1.7 / 99 (72%)	5.8 / 346 (64%)	21.2 / 1272 (87%)
	total (cum.)	**0.2 / 12**	**0.6 / 39**	**2.3 / 137**	**9.0 / 539**	**24.5 / 1471**
	EIG per refit step (s)	0.82	0.88	1.00	1.28	1.98
	EIG per no-refit step (s)	0.05	0.07	0.10	0.18	0.32
MIC (111)	hyperparameter fitting (cum.)	0.0 / 2 (11%)	0.1 / 7 (13%)	0.7 / 40 (22%)	3.7 / 223 (31%)	3.7 / 223 (12%)
	EIG, refit steps (cum.)	0.1 / 4 (25%)	0.1 / 7 (13%)	0.2 / 10 (5%)	0.2 / 14 (2%)	0.4 / 21 (1%)
	EIG, no-refit steps (cum.)	0.2 / 11 (64%)	0.7 / 41 (75%)	2.3 / 137 (73%)	7.9 / 472 (67%)	27.8 / 1666 (87%)
	total (cum.)	**0.3 / 18**	**0.9 / 54**	**3.1 / 188**	**11.8 / 709**	**31.8 / 1910**
	EIG per refit step (s)	2.22	2.29	2.44	2.77	3.58
	EIG per no-refit step (s)	0.08	0.10	0.15	0.24	0.42

F.3 ABLATION FROM SHAPLEIG TO SHAPLEYOG

Figure 5 repeats the ablation of Figure 2 for the four remaining games, Table 7 quantifies, for every game at $T = 2048$ evaluations, the speed-up of each variant over ShaplEIG and its NMSE relative to ShaplEIG, and Table 8 lists the absolute overheads and NMSEs.

Table 7: Speed-up of each variant over ShaplEIG (median over 10 seeds of the paired ratio of overheads) and its NMSE relative to ShaplEIG (geometric mean over seeds of the paired ratio; values above 1 are worse), per game at $T = 2048$ evaluations.

Variant \ Game (p)	ResNet18-SP (14)	Churn (19)	Credit-G (20)	Bankruptcy (64)	Crime (101)
+ refit schedule	9× / 0.88	11× / 1.00	11× / 1.33	25× / 1.00	39× / 1.00
+ incremental updates	54× / 0.79	58× / 1.04	55× / 1.29	72× / 0.80	80× / 0.84
+ GL quadrature	61× / 0.85	70× / 1.09	66× / 1.25	134× / 0.74	214× / 0.84
+ paired acquisition	73× / 0.87	85× / 0.87	80× / 1.31	177× / 0.83	289× / 0.84
+ odd GP = ShapleyOG	388× / 0.88	431× / 0.85	412× / 1.53	819× / 0.82	1250× / 0.92

Table 8: Absolute counterpart of Table 7: overhead (median over 10 seeds) and NMSE (geometric mean over seeds) of ShaplEIG and of each variant, per game at $T = 2048$ evaluations.

Variant \ Game (p)	ResNet18-SP (14)	Churn (19)	Credit-G (20)	Bankruptcy (64)	Crime (101)
ShaplEIG	3.2 h / 5.0e-6	3.8 h / 7.6e-5	3.7 h / 1.1e-4	11.1 h / 2.5e-4	22.4 h / 3.0e-4
+ refit schedule	20.2 min / 4.4e-6	20.6 min / 7.6e-5	20.7 min / 1.4e-4	26.6 min / 2.5e-4	34.2 min / 2.9e-4
+ incremental updates	3.5 min / 4.0e-6	4.0 min / 7.9e-5	4.0 min / 1.4e-4	9.3 min / 2.0e-4	16.7 min / 2.5e-4
+ GL quadrature	3.1 min / 4.3e-6	3.3 min / 8.2e-5	3.3 min / 1.3e-4	5.0 min / 1.8e-4	6.3 min / 2.5e-4
+ paired acquisition	2.6 min / 4.4e-6	2.7 min / 6.6e-5	2.7 min / 1.4e-4	3.7 min / 2.1e-4	4.5 min / 2.5e-4
+ odd GP = ShapleyOG	30 s / 4.4e-6	32 s / 6.4e-5	32 s / 1.6e-4	49 s / 2.0e-4	1.0 min / 2.7e-4

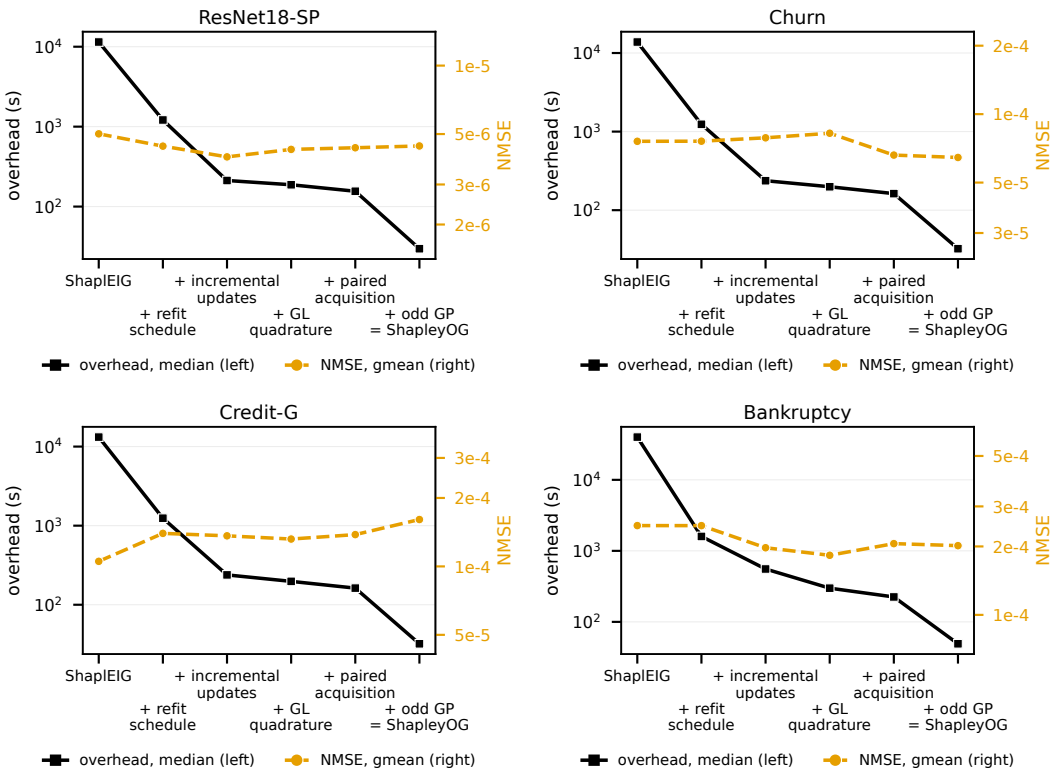


Figure 5: Ablation from `ShaplEIG` to `ShapleyOG` at $T = 2048$ evaluations on ResNet18-SP, Churn, Credit-G, and Bankruptcy (Crime is shown in Figure 2): overhead (median over 10 seeds, left axis) and NMSE (geometric mean, right axis) when adding one measure at a time.

F.4 Quadrature Study

The Gauss–Legendre quadrature of Section 3.2 is exact for $M \geq \lceil p/2 \rceil$ nodes; here we ask how many are needed in practice. On QSAR TID11 ($p = 1024$), we run `ShapleyOG` with $M \in \{514, 256, 64, 8, 2\}$ nodes, where 514 is the default of our implementation and serves as the exact arm, for both covariance terms (5 seeds, 2050 initial coalitions, up to 8192 evaluations, otherwise as in Appendix E.2). At every refit and every 256th iteration, each run also recomputes its prior covariances and EIG scores with the exact quadrature on its own state, which yields the relative errors in Table 9 and whether the exact scores select the same coalition; these checkpoints are excluded from the reported times. A sixth arm computes $\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \cdot)$ with the coefficient (ESP) route of `ShaplEIG` instead (the ESP route of the $p \times p$ term needs 34 GB at this p and is out of reach).

Down to 8 nodes, the covariances are exact to machine precision, every checkpoint selects the same coalition, and the SV estimates equal those of the exact run; only 2 nodes break the quadrature (errors up to $2 \cdot 10^{-2}$, 15% identical selections, 18–32% higher NMSE). The 8 nodes cut the covariance computations from 16 minutes to 14 seconds per run, while the total overhead stays at 3.6 hours, dominated by the incremental EIG updates and the hyperparameter fitting. The ESP arm, which differs from the exact arm only in the prior covariances, exceeded the 24-hour cap in every run, i.e., the coefficient route alone adds more than 20 hours where the quadrature needs 16 minutes.

F.5 Average Ranks

Table 10 extends the right panel of Figure 1 to the three strongest competitors: for every game and budget, it lists the average rank of `ShapleyOG`, OddSHAP, Regression MSR, and Leverage SHAP among all seven estimators (Appendix E.5). The medals mark the lowest, second lowest, and third lowest average rank of all seven estimators, so a cell such as IMDB at 256 evaluations, where

Table 9: Quadrature study on QSAR TID11 ($p = 1024$, 5 seeds) for different numbers of nodes M: median relative Frobenius error of the prior covariances and of the EIG scores against the exact quadrature at the checkpoints, share of checkpoints selecting the same coalition, NMSE (geometric mean over seeds), and median time per run of the covariance computations and of the whole method.

	relative error vs. exact				NMSE at T		time per run	
M	$\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathcal{C})$	$\mathbf{A}\mathbf{K}_\xi(\mathbf{Z}, \mathbf{Z})\mathbf{A}^\top$	EIG	same argmax	4096	8192	covariances	total
514 (exact)	–	–	–	–	7.9e-4	3.7e-4	16.4 min	3.7 h
256	4.6e-15	5.3e-15	2.2e-9	100%	7.9e-4	3.7e-4	6.2 min	3.6 h
64	1.1e-14	1.2e-14	2.0e-8	100%	7.9e-4	3.7e-4	1.3 min	3.6 h
8	8.4e-15	6.5e-14	7.5e-8	100%	7.9e-4	3.7e-4	14 s	3.6 h
2	3.9e-4	4.2e-4	2.4e-2	15%	1.0e-3	4.4e-4	7 s	3.5 h
ESP route	–	–	–	–	–	–	–	> 24 h

`ShapleyOG` is gold with an average rank of 2.9, means that every competitor is ranked worse on average (Regression MSR: 2.93 vs. 2.90); the medals absent from the four rows on Forest Fires go to KernelSHAP.

Table 10: Average rank (over 30 seeds, 1 = lowest NMSE) of `ShapleyOG` and its three strongest competitors among all seven estimators, per game and budget T; *pooled* averages over the 14 games whose grid reaches 8192 evaluations; – marks budgets beyond a game's grid. Gold, silver, bronze: lowest, second lowest, third lowest average rank of all seven estimators.

Method	Language (MDB) (14)	ResNet18_SP (14)	ViTtiny4 (16)	DINOv2-16 (16)	Forest Fires (13)	AdultCensus (14)	Estate (15)	Crime (101)	Seismic Bumps (15)	Churn (19)	Credit-G (20)	Bankruptcy (64)	Superconductivity (81)	NATICUSdroid (86)	MIC (111)	*pooled*
$T = 256$																
ShapleyOG (Ours)	2.9	1.1	2.0	2.5	3.1	1.1	1.4	1.0	1.0	1.1	1.4	1.0	1.0	1.0	1.0	1.4
OddSHAP	3.2	2.9	2.8	2.3	4.1	2.2	2.2	3.2	2.4	2.7	2.4	3.0	2.5	2.2	2.2	2.6
Regression MSR	2.9	2.4	2.6	2.2	4.2	4.7	3.1	4.9	4.3	3.6	5.2	4.9	4.2	4.7	4.9	3.9
Leverage SHAP	3.2	4.3	3.5	3.7	2.6	2.9	3.4	2.8	2.7	3.1	2.4	2.1	3.0	3.0	3.4	3.1
$T = 512$																
ShapleyOG (Ours)	2.6	1.0	1.2	2.1	3.1	1.2	1.1	1.0	1.3	1.0	2.0	1.0	1.0	1.0	1.0	1.3
OddSHAP	3.2	2.6	2.9	2.5	3.7	2.1	2.4	3.1	1.9	2.3	1.8	2.8	2.5	2.4	2.3	2.5
Regression MSR	2.6	2.5	2.5	2.6	4.4	4.3	2.8	5.4	3.9	3.4	5.0	4.9	4.4	5.0	5.4	3.9
Leverage SHAP	3.3	4.2	4.1	4.0	2.4	2.9	3.9	2.1	3.2	3.8	2.3	2.2	2.6	2.7	2.7	3.1
$T = 1024$																
ShapleyOG (Ours)	2.6	1.0	1.2	1.8	3.3	1.1	1.3	1.0	1.1	1.0	2.2	1.0	1.0	1.0	1.0	1.3
OddSHAP	3.2	2.5	2.5	2.7	3.8	2.0	2.1	2.4	2.1	2.1	1.5	2.2	2.2	2.0	2.5	2.3
Regression MSR	2.3	2.5	2.6	2.3	4.3	3.6	2.7	5.5	3.4	3.2	4.5	4.8	4.5	5.0	5.6	3.8
Leverage SHAP	3.1	4.4	4.2	4.3	3.0	3.4	3.9	2.7	3.4	3.8	2.5	2.8	2.9	3.0	2.5	3.4
$T = 2048$																
ShapleyOG (Ours)	1.9	1.0	1.2	1.8	3.4	1.4	1.6	1.0	1.4	1.0	1.6	1.0	1.0	1.0	1.0	1.3
OddSHAP	2.8	2.4	2.5	2.7	4.0	1.8	2.3	2.0	2.1	2.1	2.0	2.1	2.1	2.0	2.1	2.2
Regression MSR	1.7	2.6	2.4	1.8	3.9	2.9	2.1	5.6	2.6	2.9	3.9	4.9	4.5	4.6	5.4	3.4
Leverage SHAP	4.3	4.4	4.3	4.5	2.2	3.9	4.0	3.0	4.0	4.1	2.8	2.9	2.9	3.0	2.9	3.6
$T = 4096$																
ShapleyOG (Ours)	2.0	1.0	1.4	1.9	3.3	2.0	1.7	1.0	1.5	1.1	1.8	1.0	1.0	1.0	1.0	1.4
OddSHAP	2.5	2.4	2.4	2.5	4.3	1.7	2.1	2.0	2.0	2.0	1.7	2.0	2.0	2.0	2.0	2.1
Regression MSR	2.1	2.6	2.2	1.7	2.8	2.3	2.2	5.6	2.5	2.8	3.5	4.7	4.4	4.3	4.9	3.3
Leverage SHAP	4.0	4.3	4.4	4.6	2.3	4.0	4.1	3.0	4.0	4.0	3.1	3.0	3.0	3.0	3.0	3.7
$T = 8192$																
ShapleyOG (Ours)	1.7	1.0	1.6	2.3	–	2.6	2.1	1.0	2.0	1.3	2.1	1.0	1.0	1.0	1.0	1.5
OddSHAP	2.5	2.0	2.3	2.4	–	1.3	1.7	2.0	1.5	2.0	1.6	2.0	2.0	2.0	2.0	2.0
Regression MSR	2.0	2.9	2.2	1.6	–	2.1	2.2	5.2	2.5	2.8	2.8	4.4	4.2	4.0	4.5	3.1
Leverage SHAP	4.0	4.2	4.2	4.4	–	4.0	4.0	3.0	4.1	4.1	3.5	3.0	3.0	3.1	3.0	3.7