Figuring out the User in a Few Steps: Bayesian Multifidelity Active Search with Cokriging

Nikita Klyuchnikov
Skoltech

Davide Mottin
Aarhus University

Georgia Koutrika
Athena RIC

Emmanuel Müller
University of Bonn

Panagiotis Karras
Aarhus University

ABSTRACT

Can a system discover what a user wants without the user explicitly issuing a query? A recommender system proposes items of potential interest based on past user history. On the other hand, *active search* incites, and learns from, user feedback, in order to recommend items that meet a user's current tacit interests, hence promises to offer up-to-date recommendations going beyond those of a recommender system. Yet extant active search methods require an overwhelming amount of user input, relying solely on such input for each item they pick. In this paper, we propose MF-ASC, a novel active search mechanism that performs well with minimal user input. MF-ASC combines cheap, *low-fidelity* evaluations in the style of a recommender system with the user's *high-fidelity* input, using Gaussian process regression with multiple target variables (*cokriging*). To our knowledge, this is the first application of cokriging to active search. Our empirical study with synthetic and real-world data shows that MF-ASC outperforms the state of the art in terms of result relevance within a budget of interactions.

ACM Reference Format:
Nikita Klyuchnikov, Davide Mottin, Georgia Koutrika, Emmanuel Müller, and Panagiotis Karras. 2019. Figuring out the User in a Few Steps: Bayesian Multifidelity Active Search with Cokriging . In *KDD'19: The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 3–7, 2019, Anchorage, Alaska, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1234/56789012.345678

1 INTRODUCTION

Consider the following scenario. A user visits an online bookstore looking for a new novel to buy. Given the large collection of options that the bookstore offers, it is very difficult to identify interesting titles. Keyword search provides a starting point but still the returned results may be too many to sift through. On the other hand, faceted search creates fixed views of the book catalog that do not zoom in on the books that would be of interest to the user. Similarly, lists of popular or highly-rated books offer general recommendations. These interfaces may serve as a good starting point, but are far from helping the user quickly complete their task.

A way to meet the user's intent is to employ a recommender system [22]. However, recommender systems require a high computational cost for retraining. Ideally, a system should interact with and learn from the user, seeking the user's feedback to assess its own guesses about the user's interests. That is the idea behind *active search* [9, 15, 29, 30]: an online learning mechanism that, given a set of items, a similarity measure between them, and previous user feedback, iteratively chooses the next item to present to the user for evaluation. This mechanism balances curiosity for unexplored items (*exploration*) with the need to meet the user's needs through this interaction (*exploitation*). The goal is to maximize a cumulative relevance function over all presented items, until the user quits. Since past choices affect future ones, active search is more than a Markov Decision Process (MDP) [3]. The state-of-the-art active search method, GP-Select [29], models user utility as a sample from a Gaussian Process and applies Gaussian Process Regression (GPR) to address the exploration–exploitation dilemma.

Still, conventional active search methods expect that user provides *all* the input they receive. This requirement imposes an overwhelming burden on the user. The question arises: Could we *combine* the user's feedback with insights extracted from other information sources in an online manner, so as to alleviate the burden on the user and deliver highly relevant results within a few interactions?

In this paper, we propose an active search method that merges user inputs with inputs derived from other sources, as from a recommender system, so as to *learn* a continuous *relevance score* function that captures the user's interests. We treat the problem as one of *regression*, and apply the toolbox of Gaussian process regression with multiple target variables, i.e., *cokriging*. Our design belongs to the class of *multifidelity* methods [1, 7, 8], in which a *low-fidelity* function simulates expensive *high-fidelity* operations (e.g., car crash tests), so as to reduce the required amount of high-fidelity evaluations. Likewise, we bolster *high-fidelity* user evaluations by integrating them with correlated *low-fidelity* system-derived evaluations. The low-fidelity function is learned contemporaneously with the high-fidelity function, also in active fashion. Our experiments with real and simulated user interactions show that this *Multifidelity Active Search with Cokriging* (MF-ASC) mechanism outperforms the state of the art under reasonably correlated fidelities.

Contributions. We summarize our contributions as follows:

- A novel active search formulation that fuses continuous user scores with correlated computationally derived scores.
- Experiments on synthetic data, in which MF-ASC outperforms multifidelity methods for function optimization.
- Two real case-studies on tabular consumer ratings and information graphs.
- Real-user experiments in which MF-ASC outperforms state-of-the-art single-fidelity active search methods.

KDD'19, August 3–7, 2019, Anchorage, Alaska, USA
2019. ACM ISBN 123-4-567-8901-2/34/56…$15.00
https://doi.org/10.1234/56789012.345678

	Active	Search	Regression	Multivalued	Bayesian	Multifidelity	Cokriging
Similarity learning	✘	✘	✔	✔	✘	✘	✘
SVM$_{act}$ [27]	✔	✘	✘	✘	✘	✘	✘
BOAS [9]	✔	✔	✘	✘	✔	✘	✘
Soft-Label [30]	✔	✔	✘	✘	✘	✘	✘
GP-SOPT [15]	✔	✔	✔	✘	✔	✘	✘
GP-Select [29]	✔	✔	✔	✔	✔	✘	✘
MF-UCB [13]	✔	✘	✔	✔	✘	✔	✘
MISO [19]	✔	✘	✔	✔	✔	✔	✘
MF-PES [34]	✔	✘	✔	✔	✔	✔	✘
MF-GP-UCB [12]	✔	✘	✔	✔	✔	✔	✘
MF-ASC	✔	✔	✔	✔	✔	✔	✔

Table 1: Related work with present (✔) and absent (✘) affordances.

2 RELATED WORK

We survey related work on active search and multifidelity optimization. Table 1 gathers together previous works' characteristics.

2.1 Active Search

Online similarity learning. *Online* methods learn by interacting with the user. MindReader [11] learns a distance function among items in a database and example items provided by the user, thereby inferring an implicit query expressed by weights over attributes. Other works follow a similar approach to *similarity learning* [4, 23]. Such *online learning* methods adapt to user feedback, yet do not adaptively determine what feedback to ask for.

Classificatory Active Search. *Active learning* incrementally *selects* data items to learn from based on previous observations [24]. *Active search* applies active learning to arrive at apt search results under a limited budget of user feedback [30], balancing *exploration* of user feedback on unknown values to improve its model and *exploitation* of that model to collect high-utility items. SVM$_{act}$ [27] applies active learning to learn a Support Vector Machine binary classifier for image retrieval. Bayesian Optimal Active Search (BOAS) [9] applies Bayesian decision theory to active search for binary classification; Wang et al. [30] extend this idea to graphs with a *soft-label* model by which labeled nodes influence a query node in a manner diminishing by distance. Yet, such approaches collect binary user feedback and predict utility by means of binary classification [29].

Regressive Active Search. GP-Select [29] models user utility as a sample from a Gaussian Process and applies Gaussian Process Regression to address the active search exploration–exploitation dilemma. GP-SOPT [15] applies similar ideas on graphs, yet uses binary user feedback values, even while predicting such values by means of regression. Thus, to our knowledge, no previous work conducts active search on graphs using regression-based prediction and multi-valued reward values at the same time. We claim that this deficiency compromises quality. Besides, these methods can only improve their predictions by collecting more user feedback.

2.2 Multifidelity Optimization

Multifidelity optimization is applied in the design of complex systems [1], where a computationally expensive *high-fidelity* objective function is approximated by a less expensive *low-fidelity* function and a few high-fidelity samples. For instance, in aeronautical design minimizing friction at supersonic speed, the high-fidelity function is a measurement on an aircraft wing, while the low-fidelity function is a computer simulation [16].

MF-UCB [13] first introduced an upper confidence bound for multifidelity function optimization by multi-armed bandits. MF-GP-UCB [12] improved this bound further applying predictions based on Gaussian Process Regression. Such works combine multiple inputs of diverse fidelities in order to achieve an optimization objective. However, they assume that those diverse inputs are samples from the *same* distribution, arising out of a single phenomenon. Thus, they disregard the potential different nature of such fidelities. By contrast, cokriging treats multifidelity sources properly as samples from diverse *correlated* phenomena. Besides, such function optimization methods are designed with an objective of function optimization rather than active search, i.e., they do not directly apply to a *cumulative objective* as required by active search. Similarly, MF-PES [34] performs function optimization by minimizing the predictive entropy, while posing restrictive assumptions on its objective function; thus, MF-PES cannot accommodate an active search objective either. Last, MISO [19] performs function optimization by combining multiple sources, yet it applies gradient-based optimization in order to calculate its acquisition criterion; thus, it is inappropriate for online active search applications where a gradient of user interest cannot be derived conveniently. Overall, to our knowledge, no previous work conducts active search by combining multi-valued reward values via regression.

3 DEFINITIONS AND PROBLEM SETTING

Active search is a process that progressively learns and meets the user's tacit interests. To do so, it seeks the user's feedback on its own guesses, balancing exploration of the unknown with exploitation of the known. The user provides feedback by means of an undisclosed *user evaluation* function $w : \mathcal{X} \mapsto [0, 1]$ on any item $\mathbf{x}$ in a dataset $\mathcal{X} \subseteq \mathbb{R}^n$. By our *multifidelity design*, the system also holds an internal approximation of the user relevance score, $\widetilde{w} : \mathcal{X} \mapsto [0, 1]$; we discuss $\widetilde{w}$ candidates in Section 5.

At each step the system retrieves an item $\mathbf{x} \in \mathcal{X}$ from the dataset, obtains its score either from the user or internally from $\widetilde{w}$, and pays a *fixed operation cost*, $c \in \mathbb{R}^+$ for asking the user and $\widetilde{c} \in \mathbb{R}^+$ for an internal evaluation. The system may request the evaluation of the same item $\mathbf{x} \in \mathcal{X}$ twice, once from the user and once internally; these two choices correspond to the *high-fidelity* (φ^H) and *low-fidelity* (φ^L) strategy, respectively. We call the set of items evaluated by the user so far $S^H \subseteq \mathcal{X}$, and the set of items evaluated by the system $S^L \subseteq \mathcal{X}$. The *utility* of a subset $S \subseteq \mathcal{X}$ is the total relevance of items in the S, $\mathcal{U}(S) = \sum_{\mathbf{x} \in S} w(x)$. We aim to find a policy for selecting S^H and S^L that are expected to gain the highest utility by the end of the interaction, under a given budget Λ:

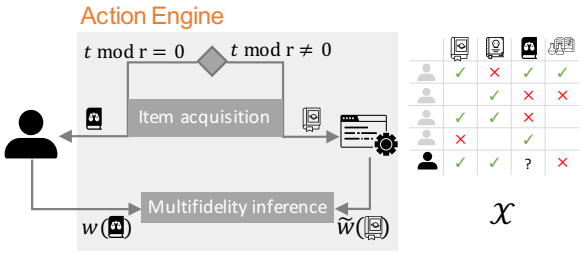
$$\arg\max_{S^H, S^L \subseteq \mathcal{X}} \quad \mathcal{U}(S^H) \quad \text{subject to} \quad c|S^H| + \widetilde{c}|S^L| \ \leq \ \Lambda$$

The utility involves only S^H, since the user sees only S^H; internal low-fidelity evaluations of S^L are undisclosed to the user, to avoid any unconscious bias. Finding the optimal policy is computationally intractable even for binary w; some works approximate the optimal plan with one-step lookahead [2] or heuristic methods [14].

4 THE MF-ASC FRAMEWORK

Our framework of Multifidelity Active Search with Cokriging (MF-ASC) seeks, and learns from, user feedback using two instruments: an *action engine* that implements a policy and an *approximate relevance function* $\widetilde{w}(\cdot)$ that estimates the user evaluation $w(\cdot)$.

Figure 1: A step of the action engine at time t used for recommendation. The data $\mathcal{X}$ is a matrix of users and item ratings. MF-ASC selects one book to be rated, and the system either shows the item to the user for evaluation (when $t \mod r = 0$) or evaluates it internally with low fidelity.



4.1 The Action Engine

The action engine selects items for evaluation; it overcomes the problem's intractability and addresses the exploration-exploitation dilemma using the acquisition criterion of [25], also used in GP-Select [29]. Our contribution with respect to [29] is that we provide the means to integrate high- and low-fidelity sources by Bayesian multifidelity inference. A parameter $r \geq 0$ determines the ratio of low-fidelity to high-fidelity calls; this parameter can be fixed in advance or decided by the user. The problem of adapting r to given needs is orthogonal to our work.

Algorithm 1 presents our action engine. For each fidelity choice, it first computes the parameters of a regression model using the current state information (Lines 2 and 6); it returns the item $\mathbf{x} \in \mathcal{X}$ that maximizes an Upper Confidence Bound (UCB)[1] acquisition criterion [25] by the regression model for either fidelity, excluding previously chosen items (Lines 3, 8, 10). The β_t parameter balances exploration and exploitation: large β_t favors exploring items having high $\sigma(\mathbf{x})$, i.e., *uncertainty* about their relevance, while small β_t favors exploiting items having high $\mu(\mathbf{x})$, i.e., *value* of relevance.

We emphasize that our system design is independent of the choice of acquisition criterion. A choice better than UCB would improve performance, while leaving the multifidelity inference mechanism intact. Yet UCB yields strong regret guarantees, as we explain in Section 4.3.

[1]For $t = 0$, when no evaluation has been already provided, it returns a random item.

Algorithm 1 Action engine

Input: Data $\mathcal{X}$, S^H, S^L, Time t
Params: Fidelity ratio r
1: **if** $t \mod r = 0$ **then**
// Where ρ, $\mu = \mu^w$; $\sigma = \sqrt{\text{diag}(\mathbf{K}^w)}$
2: $(\rho, \mu, \sigma) \leftarrow$ MF-Infer-High(S^H, S^L, κ)
// Acquisition criterion in [25]
3: $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X} \setminus S^H} \mu(\mathbf{x}) + \beta_t \sigma(\mathbf{x})$
4: $S^H \leftarrow S^H \cup \{\mathbf{x}_t\}$
5: **else**
// Where $\mu = \mu^{\widetilde{w}}$; $\sigma = \sqrt{\text{diag}(\mathbf{K}^{\widetilde{w}})}$; Sec 4.2.2
6: $(\mu, \sigma) \leftarrow$ MF-Infer-Low(S^H, S^L, κ)
// If negative correlation return lower-confidence bound
7: **if** $\rho > 0$ **then**
8: $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X} \setminus S^L} \mu(\mathbf{x}) + \beta_t \sigma(\mathbf{x})$
9: **else**
10: $\mathbf{x}_t \leftarrow \arg\min_{\mathbf{x} \in \mathcal{X} \setminus S^L} \mu(\mathbf{x}) - \beta_t \sigma(\mathbf{x})$
11: $S^L \leftarrow S^L \cup \{\mathbf{x}_t\}$

4.2 Multifidelity Inference

We now discuss the details of our inference model. To estimate the value and uncertainty of the relevance score at each node, with either fidelity, we assume that score at time t is a random variable sampled from a Gaussian distribution, and apply *cokriging* [7] (i.e., Gaussian process regression over multiple target variables) informed by the data structure and previous scores.

4.2.1 Prior Distributions. We assume that *similar items* are more likely to get the same score. Then, the *prior* joint probability distribution of the relevance score function w is a Gaussian $\mathcal{N}(0, \mathbf{K})$ with mean 0 and a covariance matrix $\mathbf{K}$ reflecting item similarity. To capture this similarity, we face a choice between (i) *implicit* parameterization, which constructs a covariance function *directly* from the item matrix; and (ii) *explicit* parameterization $p : \mathcal{X} \to \mathbb{R}^m$, which first embeds items in a low-dimensional vector space and then applies a parametric covariance function (kernel) κ on each pair of items $\mathbf{x}, \mathbf{z} \in S^H \cup S^L$ — typically a squared exponential kernel $\kappa(v, u) = \eta e^{-\theta |p(v) - p(u)|^2}$ — to compute a covariance matrix $\mathbf{K}$, estimating kernel parameters $\{\eta, \theta\}$ by maximum likelihood estimation [7]. Explicit parameterization is flexible due to its tunable parameters. On the other hand, it is complicated and incurs information loss as a result of embedding.

We opt for a compromise between the explicit way (which considers relations between all item pairs) and the implicit way (which considers only adjacency connections): first, we compute a similarity matrix $\mathcal{S}$ for the dataset (details discussed in Section 5). Then we extract the Laplacian of $\mathcal{S}$, $\mathcal{L} = \mathbf{D} - \mathcal{S}$, where $\mathbf{D}$ is a diagonal matrix obtained by summing the values in each row of $\mathcal{S}$, $\mathbf{D}_{ii} = \sum_{j=1}^{|\mathcal{X}|} \mathcal{S}_{ij}$. Last, we compute a covariance matrix from the Laplacian of $\mathcal{S}$, $\mathbf{K}_0 = (\mathcal{L} + \lambda I)^{-1}$, where λ is a regularization parameter that determines how much credit is to be given to $\mathcal{S}$.

4.2.2 Posterior Distributions. We now obtain the means μ and standard deviations σ of the *posterior* probability distributions for relevance score random variables at time t, based on Gaussian priors and the sets of items S^H and S^L evaluated at time t. We represent the user and system scores for all items S^H and S^L evaluated by

time t as a vector $\mathbf{y} = (\mathbf{y}^L, \mathbf{y}^H)^\top$, where vector $\mathbf{y}^L$ holds low-fidelity evaluations and vector $\mathbf{y}^H$ holds high-fidelity ones.

Low-Fidelity Inference (MF-Infer-Low). Let t^L be the number of items evaluated by low fidelity at time t; then we denote:

$$\mathbf{k}^{\widetilde{w}}(\mathbf{x}) = \left[\mathbf{K}_0(\mathbf{x}_1, \mathbf{x}), ..., \mathbf{K}_0(\mathbf{x}_{t^L}, \mathbf{x})\right]^\top \quad \text{and}$$
$$\mathbf{K}_0^{\widetilde{w}} = \left[\mathbf{K}_0(\mathbf{x}_i^L, \mathbf{x}_j^L)\right]_{i,j=1}^{t^L}$$

$\mathbf{K}_0^{\widetilde{w}}$ is a square matrix made out of the contents of the covariance matrix $\mathbf{K}_0$ for items already evaluated by low fidelity at time t. By Bayesian inference calculations [21], the parameters of the *low-fidelity* posterior distribution $\mathcal{N} \sim (\mu^{\widetilde{w}}, \mathbf{K}^{\widetilde{w}})$, are

$$\mu^{\widetilde{w}}(\mathbf{x}) = (\mathbf{k}^{\widetilde{w}}(\mathbf{x}))^\top (\mathbf{K}_0^{\widetilde{w}} + \sigma_n^2 I)^{-1} \mathbf{y}^L \tag{1}$$
$$\mathbf{K}^{\widetilde{w}}(\mathbf{x}, \mathbf{z}) = \mathbf{K}_0(\mathbf{x}, \mathbf{z}) - \mathbf{k}^{\widetilde{w}}(\mathbf{x})^\top (\mathbf{K}_0^{\widetilde{w}} + \sigma_n^2 I)^{-1} \mathbf{k}^{\widetilde{w}}(\mathbf{z}) \tag{2}$$

where σ_n^2 denotes the variance of noise associated with the model. We give Equation 2 in its general form, yet calculate only the diagonal entries $\mathbf{K}^{\widetilde{w}}(\mathbf{x}, \mathbf{x})$ of this covariance matrix, as we only need variance values on single items (Algorithm 1, Lines 2 and 6).

High-Fidelity Inference (MF-Infer-High). Next, we assume that the high-fidelity (user) score function $w(\cdot)$ is correlated with the low-fidelity (system) score function $\widetilde{w}(\cdot)$, and exploit this correlation for inferring high-fidelity. The user score should then be a linear combination of two independent stationary Gaussian processes over the items $\mathcal{X}$, namely $\widetilde{w}(\cdot)$ scaled by a factor ρ and a Gaussian process δ capturing the said correlation; thus, for an item $\mathbf{x} \in \mathcal{X}$:

$$w(\mathbf{x}) = \rho \cdot \widetilde{w}(\mathbf{x}) + \delta(\mathbf{x}). \tag{3}$$

As noted in Section 4.2.1, the priors of $\widetilde{w}$ and δ when the interaction starts, based on item similarities, are Gaussians. Let f denote either $\widetilde{w}$ or δ. Then $\boldsymbol{f} = (f(v_1), ..., f(v_{|\mathcal{X}|})) \sim \mathcal{N}(0, \mathbf{K}_0)$, i.e., $\Pr\{\boldsymbol{f}=\boldsymbol{x}\} \propto \exp\left(-\frac{1}{2}\left(\sum_{i,j=1}^{|\mathcal{X}|} \mathcal{S}_{ij}(x_i^2 - x_i x_j) + \lambda \sum_{j=1}^{|\mathcal{X}|} x_j^2\right)\right)$, where λ is the regularization parameter in the computation of $\mathbf{K}_0$ from the Laplacian of $\mathcal{S}$. The cokriging correlation (Eq. 3) allows us to build the inference of high-fidelity user relevance on top of the low-fidelity one. We calculate ρ as:

$$\rho = \frac{(\mathbf{y}^H)^\top \mathbf{K}_0^\delta \, \mu^{\widetilde{w}}(S^H)}{(\mu^{\widetilde{w}}(S^H))^\top \mathbf{K}_0^\delta \, \mu^{\widetilde{w}}(S^H)},$$

where $\mu^{\widetilde{w}}(S^H) = [\mu^{\widetilde{w}}(\mathbf{x}_1^H), ..., \mu^{\widetilde{w}}(\mathbf{x}_{t^H}^H)]^\top$ and $\mathbf{K}_0^\delta$ is made out of the contents of the covariance matrix for items already evaluated by high fidelity at time t, $[\mathbf{K}_0(\mathbf{x}_i^H, \mathbf{x}_j^H)]_{i,j=1}^{t^H}$. Eventually, we obtain high-fidelity posteriors, $\mu^w(\mathbf{x})$ and $\mathbf{K}^w(\mathbf{x}, \mathbf{z})$, as in Equations (1) and (2), using the vector $[\mathbf{y}^L, \mathbf{y}^H]$ obtained concatenating vector $\mathbf{y}^L$ and $\mathbf{y}^H$ in place of $\mathbf{y}^L$ and the following combined covariances in place of $\mathbf{k}^{\widetilde{w}}$ and $\mathbf{K}_0^{\widetilde{w}}$, respectively:

$$\mathbf{k}^w(\mathbf{x}) = \begin{pmatrix} \rho \mathbf{K}_0^{\widetilde{w}}(S^L, \{\mathbf{x}\}) \\ \rho^2 \mathbf{K}_0^{\widetilde{w}}(S^L, \{\mathbf{x}\}) + \mathbf{K}_0^\delta(S^H, \{\mathbf{x}\}) \end{pmatrix} \quad \text{and}$$
$$\mathbf{K}_0^w = \begin{pmatrix} \mathbf{K}_0^{\widetilde{w}}(S^L, S^L) & \rho \mathbf{K}_0^{\widetilde{w}}(S^L, S^H) \\ \rho \mathbf{K}_0^{\widetilde{w}}(S^H, S^L) & \rho^2 \mathbf{K}_0^{\widetilde{w}}(S^H, S^H) + \mathbf{K}_0^\delta(S^H, S^H) \end{pmatrix}$$

where $\mathbf{K}(A, B)$ denotes the matrix $[\mathbf{K}(\mathbf{a}_i, \mathbf{b}_j)]_{i,j}$ of pairwise correlations between items in sets A and B. These results conclude our discussion of multifidelity inference (i.e., Lines 2 and 6 of Algorithm 1). We reiterate the fact that, aptly, low-fidelity calculations are also used in high-fidelity inference.

4.3 Algorithm Complexity and Regret Guarantees

The complexity of MF-ASC depends on the outlined inference method, which computes the posterior at each iteration of Algorithm 1 and applies it to eligible nodes (Line 3). This computation takes $O(|\mathcal{X}_t|^3 + |\mathcal{X}| \cdot |\mathcal{X}_t|^2)$, where $\mathcal{X}_t = S^H \cup S^L$ is the set of low- and high- fidelity evaluations performed by time t, $|\mathcal{X}_t|^3$ stands for matrix inversion in Equations 1 and 2, and $|\mathcal{X}| \cdot |\mathcal{X}_t|^2$ stands for matrix-vector multiplications for each $\mathbf{x} \in \mathcal{X}$. We reiterate that, as we need to know only variances in items, we only need to calculate posterior covariances along the diagonal. Hence, the overall complexity of MF-ASC per iteration is $O(|\mathcal{X}_t|^3 + |\mathcal{X}| \cdot |\mathcal{X}_t|^2)$; that is *cubic only* in the number of evaluations, which is constrained by the budget Λ; the complexity is *linear* in the number of nodes.

Given that datasets are finite discrete structures, the regret bounds in [25] apply to our method. Thus, if we set the exploration-exploitation tradeoff parameter to $\beta_t = \sqrt{2\log(|\mathcal{X}|t^2\pi^2/6\delta)}$, then our action engine has a regret guarantee $O(\sqrt{T\gamma_T \log|\mathcal{X}|})$ with probability $1-\delta$, where T is the number of user interactions, $\gamma_T = \frac{1}{2}\sum_{t=1}^{T} \log(1 + \sigma_n^{-2}\sigma_{t-1}(\mathbf{x}_t))$, and σ_{t-1} the posterior variance at time $t-1$ (Algorithm 1, Line 2).

5 APPLICATIONS OF MF-ASC

MF-ASC facilitates effective active search on different data types. An exhaustive study of data types to use our framework on is outside the scope of this work. Here, we suggest two practical domains: *consumer recommendation* and *information graph exploration*.

The framework has two key components: a *similarity function* defined between pairs of objects and *fidelity functions*. The low-fidelity function in particular is critical for the quality of predictions. An inadequate choice would let posterior estimates degenerate to a random predictor [32], due to the correlation by cokriging (Equation (3)). An effective low-fidelity function may be based on historical user interactions, such as query logs, and domain knowledge. The design of such an elaborate fidelity function falls outside the scope of this work. We present a general framework and instantiate it with uncomplicated functions to highlight the benefits of the framework as such.

5.1 Case 1: Consumer Recommendation

We first apply our active search framework in the domain of a classical consumer-rating-based recommender system. A conventional recommender system estimates user preferences based on past interactions only. The data $\mathcal{X}$ is a matrix of users and items, where each user expresses preferences on one or more items. We show the effect of upgrading such a system with our MF-ASC mechanism: the system itself provides low-fidelity input, while it also invites the user to score selected items, so as to progressively learn the user's current preferences and improve its recommendations.

Similarity function. In general, similarity between items can be obtained from their attributes. The particular form of similarity depends on the downstream application. In collaborative filtering recommendations, one can obtain similarity from the distance between items' latent factors, constructed, e.g., by Singular Value Decomposition (SVD) of the user-item preference matrix [5].

Fidelity functions. In this case we take the predictions of a conventional recommender algorithm as the low-fidelity function and real user's input as the high-fidelity function.

5.2 Case 2: Information Graph Exploration

Next, we expand our study to the domain of information graphs. An *information graph* is a quadruple $G : \langle V, E, \phi, \psi \rangle$, where V is a set of nodes, $E \subseteq V \times V$ is a set of edges, $\phi : V \mapsto L_V$, $\psi : E \mapsto L_E$ are node and edge labeling functions, respectively. Information graphs present a particularly hard domain for learning algorithms due to their high dimensionality and non-trivial structural relationships. We show the effect of applying MF-ASC for search over such a structure; in the absence of rating data, we simulate the low-fidelity input that a recommender system would provide as a corrupted version of the user's high-fidelity input.

Similarity function. We compute a node similarity matrix $\mathcal{S}$ (Section 4.2) in a semi-supervised manner: We add an edge between nodes with probability proportional to the Jaccard similarity of their textual labels; in case of multi-attribute nodes, we take the weighted sum of the Jaccard similarities among attributes of the same type. Then we extract a node2vec [10] embedding of 50 dimensions, using 50 random walks of length 10 from each vertex, window size 5, and a skip-gram model with negative sampling 5. Last, cosine similarities of node embeddings give the entries of $\mathcal{S}$.

Fidelity functions. We represent a user's intentions via an *intended set* $\mathcal{I}$, i.e., a set of nodes that the user tacitly wants to find. We construct $\mathcal{I}$ as the result of a query, which the user is presumably unable to formulate explicitly, and simulate the *high-fidelity* score w for a vertex v as the average cosine similarity from v to nodes in $\mathcal{I}$, normalizing all scores to $[0, 1]$.

A *low-fidelity* function should approximate the preferences of a user. We derive such functions from high-fidelity ones by introducing controlled errors, using a k-nearest-neighbor (kNN) approach: for a given node v, we remove ℓ nodes from the set of its k NNs, and calculate low-fidelity values as the average of high-fidelity values for non-removed neighbors; by tuning $\ell \leq k$ and k, we obtain low-fidelity functions with an arbitrary correlation to high fidelity.

6 EXPERIMENTAL EVALUATION

We evaluate MF-ASC against the state of the art in active search and baselines. We implemented all methods in *Python* 2.7 and ran experiments on a 16G Intel i7-4790 machine running Ubuntu 14.04LTS.

Experimental methodology. We experiment with real and simulated users, on both consumer recommendation and information graph exploration. We measure the **quality** of obtained output in terms of *relative regret*. The *regret* is the deviation of utility from the optimal utility. We compute the optimal utility $\mathcal{U}^*_\Lambda$ for a given budget Λ by summing the relevance values w of the ground-truth top-Λ nodes; then, the regret is $\mathcal{U}^*_\Lambda - \mathcal{U}_\Lambda$, where $\mathcal{U}_\Lambda$ is the sum of utilities attained by the evaluated method by the end of the interaction. Then the *relative regret* is the ratio between the obtained regret and the average regret of the random method Rand over 50 runs: $\frac{\mathcal{U}^*_\Lambda - \mathcal{U}_\Lambda}{\mathcal{U}^*_\Lambda - \mathcal{U}^{\text{Rand}}_\Lambda}$. In each experiment, we report the average over examined instances; occasionally, we refer to this *average relative regret* simply as *Regret*; dashed curves show the 0.2- and 0.8-quantiles for corresponding curves of the same color. Relative regret measures performance with respect to a random acquisition criterion; the line Regret=1 represents the performance of Rand. In Section 6.4 we also report Recall@Λ, i.e., the ratio of relevant items (i.e., items in the intended-set) in the top-Λ elements returned by each method. We also report the **time** needed to select the next item to evaluate and analyze the scalability of the methods in Section 6.7.

Datasets. We experiment on three datasets with real user data:

- Yahoo: The Yahoo music dataset from the KDD-Cup 2001 [6], containing song ratings on a scale from 0 to 100. We extracted 5% of the most active users and the most rated songs, obtaining 50k users and 31k songs.
- Yelp: A dataset from Yelp challenge[2] containing business ratings on a scale from 1 to 5. We selected users and businesses that have at least 100 reviews, obtaining 68k users and 12k businesses.
- ACL[3]: A graph expressing the research interests of 28 researchers across 597 papers published in the Association for Computational Linguistics conference from 2000 to 2006, presented as lists of paper IDs; we used these lists as *intended sets*. We extract similarities among papers using the full graph to which they belong, including papers from other venues, and use the papers' term-frequency feature vectors for cosine similarity calculations (see Section 6.1).

In addition, we have devised *simulated* user data with these real-world graph data:

- Freebase: We downloaded a snapshot of the Freebase [4] knowledge graph and extracted a *computer* domain, FB-Comp, containing information about computer models, manufacturers, software, and hardware, as well as a *fiction* domain, FB-Fict, containing information on novels.
- Microsoft Academic Graph: We extracted two samples from a network[5] of authors, publications, affiliations, and venues: MAG is a subgraph related to Computer Science conferences, with a taxonomy of topics related to keywords; MAG-Sm is a subset of MAG containing only papers. We calculate similarities among nodes in MAG-Sm using edges in MAG as well.

The table below lists the characteristics of graph datasets: number of edges $|E|$, vertices $|\mathcal{X}|$, node labels $|L_V|$, edge labels $|L_E|$; *avg*, *min*, and *max* node degree; modularity [18]; and density, $|E|/\binom{|\mathcal{X}|}{2}$.

	Size				*Degree*		
Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert L_V\rvert$	$\lvert L_E\rvert$	Avg/Min/Max	Mod.	Density
FB-Comp	9.7k	18k	9.7k	70	3.7/1/1082	0.70	4×10^{-4}
FB-Fict	2.5k	16k	2.5k	74	13.2/6/1081	0.63	5×10^{-3}
MAG	6k	14k	6k	5	4.8/1/391	0.69	8×10^{-4}
MAG-Sm	1.1k	3.8k	1.1k	1	-/-/-	-	-
ACL	597	614	597	1	-/-/-	-	-

User simulation. On graph data with simulated users, we construct simulated user feedback as a preference score w arising from

[2] https://www.yelp.com/dataset_challenge
[3] http://www.comp.nus.edu.sg/ sugiyama/SchPaperRecData.html, 'dataset 1'
[4] https://developers.google.com/freebase/data
[5] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

a structured query on the graph, as discussed in Section 5.2. We sample 5 nodes uniformly at random from the set of query results and use them as the initial set with all methods. We hand-crafted 25 queries for FB-Fict, 17 for FB-Comp, 19 for MAG, and 43 for MAG-Sm. All constructed queries for the first three datasets return at least 15 results with no or little overlap. The table below shows examples of such queries.

Dataset	*Query description*
FB-Comp	Programming languages influenced by Python.
	Soft with open source license.
FB-Fict	Fiction characters who have a super ability to fly.
	Fiction characters who are parents of students.
MAG	Authors that published papers alone.
	Affiliations in Germany related to Genomics field.
MAG-Sm	All papers from field Anomaly Detection.
	All papers from field User Modeling.

Implemented Algorithms. We implemented a baseline method, a degenerate (randomized) active search variant, two single-fidelity active search algorithms derived from previous works, and a multifidelity method designed for function optimization adjusted to active search purposes. In particular, we compare MF-ASC against the following contestants:

- LabelProp: a baseline method that applies label propagation [20] with a k-nearest-neighbor kernel at each iteration and returns the node having the maximum probability to belong to Class 1 for user evaluation. We embed the similarity matrix $\mathcal{S}$ in a 2-dimensional[6] space using t-SNE [28] and run the implementation of label propagation in scikit-learn[7]. As the algorithm is designed for classification, we convert user evaluations provided at previous iterations to binary, marking a node as positive (class 1) if its user relevance score deviates from the maximum by at most 0.1.
- Rand: a method with an acquisition criterion that randomly selects a node to be evaluated.
- GP-SOPT: the single-fidelity active search method in [15] that uses the Laplacian of the data set's similarity matrix $\mathcal{S}$ and a σ-optimality criterion. We include this method in our study for the purpose of establishing the superiority of the UCB acquisition criterion of the σ-optimality criterion. We do that by constructing the following single-fidelity active search method, which represents the best of previous work.
- GP_Lapl: a single-fidelity active search method that combines the best choices of previous work: applies the UCB acquisition criterion, as GP-Select [29], and represents the data set by the Laplacian of its similarity matrix $\mathcal{S}$, as GP-SOPT [15]. We set $\beta_i = 0.01$, as explained in Appendix A.1. The default form of GP_Lapl operates with high-fidelity input only, i.e., with user input as an active search method. We also create a variant, LF-only, that works with low-fidelity input only, ergo like a conventional recommender system that does not request online user input.
- MF-GP-UCB: the state-of-the-art multifidelity function optimization method [12]. We adjusted the algorithm to an active search setting by disallowing the querying of the same item and fidelity pair more than once.

Parameter settings. We set $c = 1$ and $\widetilde{c} = 0$, hence the budget Λ expresses the number of user interactions. We set β_t to 0.001 for high-fidelity and 0.01 for low-fidelity, steering low-fidelity evaluations towards exploration and high-fidelity ones towards exploitation; other values yielded worse results. We set the regularization parameter λ to 0.01, as in [15]. Unless otherwise indicated, we set the fidelity ratio in Algorithm 1 to $r = 5$, corresponding to one round of user feedback for every 5 low-fidelity evaluations. We assume a default correlation of 1 between the two fidelities. We delve into the effect of different correlation values in Section 6.5. Last, we deem the computation of the similarity matrix $\mathcal{S}$ in Section 5 as preprocessing for all algorithms.

The table below provides the default parameter values used in the experiments, unless otherwise stated.

Parameter	*Default value*	*Meaning*
c	1	User evaluation cost
$\widetilde{c}$	0	System evaluation cost
β_t	high-fid: 0.001;low-fid: 0.01	Exploration/exploitation tradeoff (Alg. 1)
r	5	Low-high fidelity ratio (Alg. 1)
λ	0.01	Regularization parameter for $\mathbf{K}_0$

Summary of results. Our results with real-user data, reported in Section 6.1, confirm the advantage gained over the single-fidelity method, GP_Lapl, as well as over the multifidelity method MF-GP-UCB by using cokriging for fidelity fusion. Section 6.2 investigates our choice of the acquisition criterion, showing that the GP_Lapl acquisition criterion we employ is preferable to that of GP-SOPT. In Section 6.3 we study the effect of the fidelity ratio r. Section 6.4 establishes the superior performance of MF-ASC over state-of-the-art methods in an ideal case, while Section 6.5 shows that MF-ASC can predict simulated users up to 3× faster if the correlation between low- and high- fidelity is at least 0.5. Section 6.6 illustrates that MF-ASC outperforms single-fidelity methods even when the simulated user input is discretized. Last, Section 6.7 shows that MF-ASC achieves real-time performance with a fast learning rate.

6.1 Real User Preferences

First, we test algorithms on preferences derived from real users on the domains discussed in Section 5: consumer recommendation and information graph exploration.

Consumer Recommendation. In this experiment we use the Yelp and Yahoo data. High-fidelity evaluations are provided by recorded user-item ratings. Since these ratings are incomplete, we restrict the search area of the high-fidelity function during tests for all algorithms, so that they would only query high-fidelity for items on which there is a recorded user score. Thus, we use exactly the score that the real user in question would provide. On the Yelp data, we define the low-fidelity function as the average score of an item (i.e., business) by all friends of a user, if any. We obtain a low-fidelity function on the Yahoo data by partitioning it into *training* (60%) and *testing* (40%) parts along the time dimension. The testing part provides high-fidelity evaluations, while the training part provides low-fidelity predictions of the test data by means of collaborative filtering with SVD.

We test on 30 randomly selected users for each dataset. Figure 2 presents our cumulative results. On both data, MF-ASC outperforms LabelProp and MF-GP-UCB, as well as the random baseline by a wide confidence margin. On the Yahoo data, the single-fidelity method, GP_Lapl, performs better than multi-fidelity methods.

[6] We tried higher dimensionality, without much improvement, hence we sticked to 2.
[7] http://scikit-learn.org/stable/modules/label_propagation.html

This is due to weak correlations between low and high fidelity: the average Pearson coefficient between fidelities on the Yahoo data is 0.13, whereas on the Yelp data it is 0.4. Therefore, on the Yahoo data, multi-fidelity methods cannot regain the budget spent on exploring the dependence between fidelities. Despite this poor performance of multi-fidelity methods on the Yahoo data, MF-ASC clearly outperforms MF-GP-UCB. This result testifies the virtues of the cokriging approach to data source fusion.

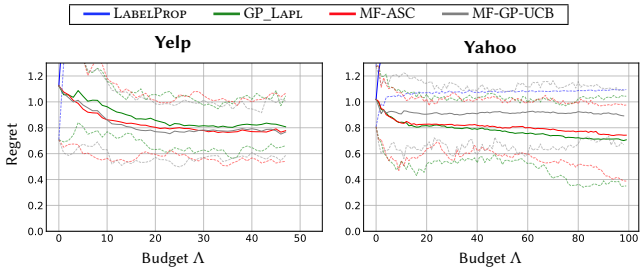


Figure 2: Relative regret vs. Λ on real datasets.

In the above, we examined a single-fidelity method that uses only the high-fidelity component of MF-ASC. In order to complete our study, we need to also examine a single-fidelity method that uses only the low-fidelity component of MF-ASC, hence behaves like a traditional recommender system. Figure 3 presents our results with such a method, LF-only. We observe that LF-only fares much worse than active search methods, reaffirming the advantage of active search over a conventional recommender system.

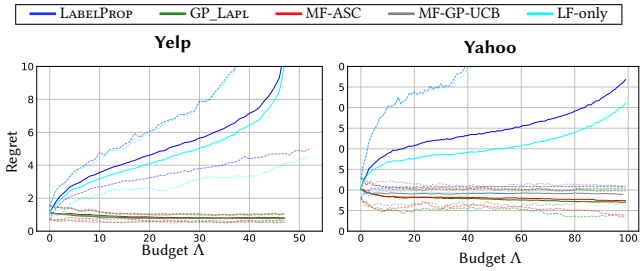
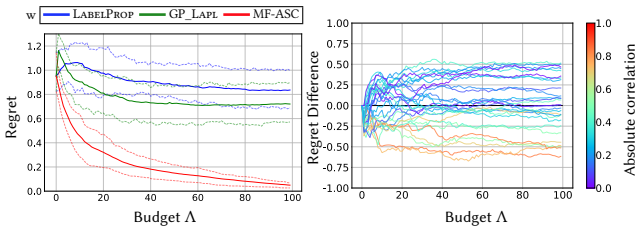


Figure 3: Relative regret vs. Λ on real data with LF-only.

Information Graph Exploration. Here, we use the ACL dataset. We obtain high-fidelity evaluations by averaging the term-frequency-vector cosine similarity between a paper and the papers in a researcher's *intended set*. Figure 4a shows results for a best-case scenario, where low-fidelity has correlation=1.0 with high-fidelity. In a more realistic scenario, *low-fidelity* evaluations are derived by cosine similarity between a paper and a researcher's most recent papers (not the full list). Figure 4b shows the regret difference between MF-ASC and GP_Lapl per researcher in that case, color-coding the measured (absolute) correlation between high- and low-fidelity. MF-ASC clearly outperforms GP_Lapl when the absolute correlation is greater than 0.75, consistently with our observations on simulated user interests. This result is due to the sensitivity of MF-ASC to noise in low-fidelity data. In a passive learning setting multifidelity models perform no worse than single-fidelity ones [31, 33], as the former are reducible to the latter. However, in active learning, low-fidelity noise affects the search on high-fidelity data.



(a) Regret vs budget (Λ) **(b) Difference between regret of MF-ASC and GP_Lapl.**

Figure 4: ACL with real-user intended sets.

6.2 Assessing Selection Strategies

We now proceed to an exhaustive experimental evaluation with simulated user data. We first assess our choice of acquisition criterion vs. the σ-optimality criterion of GP-SOPT [15]. We set GP-SOPT against GP_Lapl [25], which uses the UCB criterion that we have adopted in MF-ASC. We set the value of β_t, which controls the exploration-exploitation tradeoff in GP_Lapl, to 0.01, as we report in Appendix A.1. Figure 5 shows the results on MAG-Sm, tuning the GP-SOPT α and k parameters. Notably, GP_Lapl, represented by a black curve, outperforms all GP-SOPT variants, represented by solid-color curves. This result establishes that σ-optimality is inappropriate for regression problems. Henceforth, we use GP_Lapl as the state-of-the-art benchmark combining best practices of previous works.

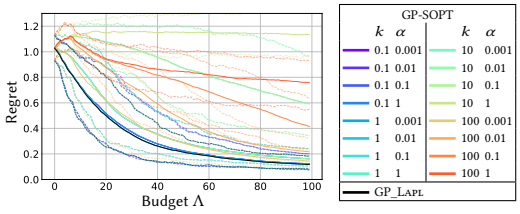


Figure 5: Relative regret of GP-SOPT compared to GP_Lapl on MAG-Sm for different k, α.

6.3 Varying the Fidelity Ratio (r)

Next, we study the effect of the budget Λ and fidelity ratio r on regret of MF-ASC. Figure 6 presents our results on MAG-Sm. Expectedly, the budget Λ, i.e., the number of high-fidelity user evaluations, affects quality, achieving a 5-fold improvement from 10 to 100 queries. The ratio of low-fidelity queries per high-fidelity evaluation also has significant, albeit gradually attenuated, effect. This result justifies our choice of $r = 5$ as a default value that achieves a reasonable quality-speed tradeoff.

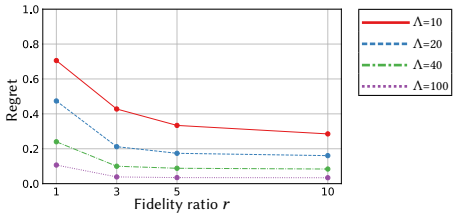


Figure 6: Relative regret on MAG-Sm vs. r.

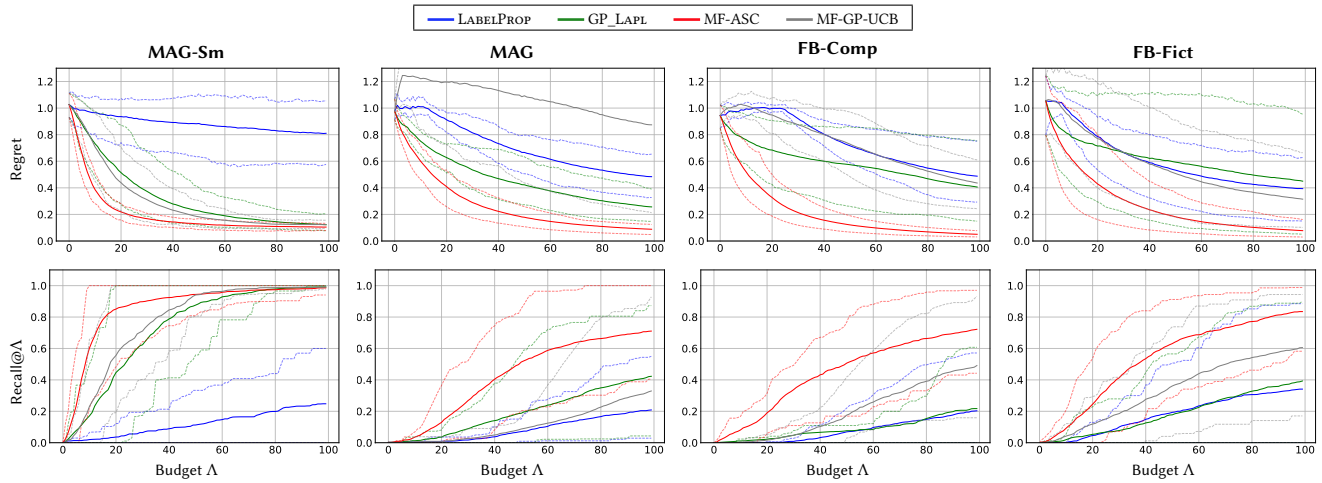
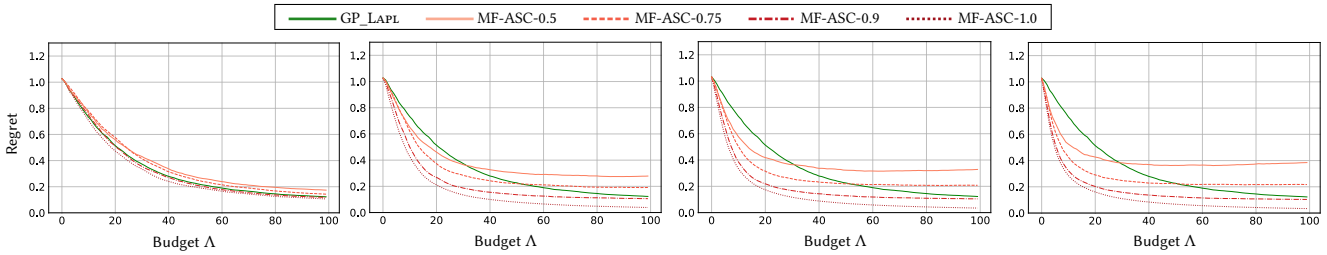
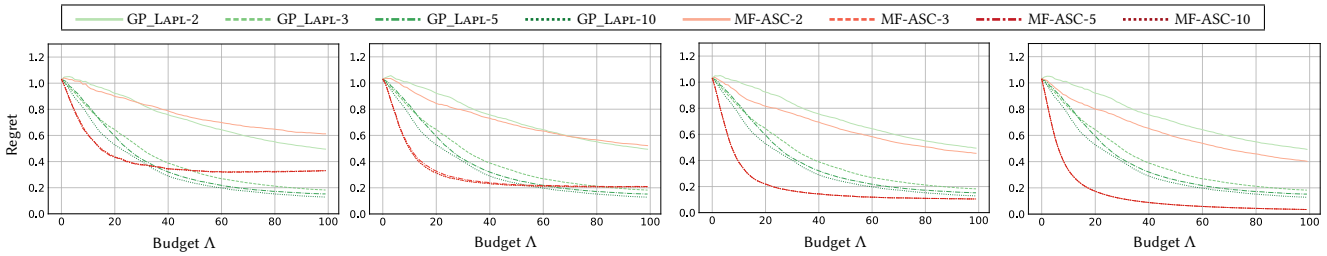


Figure 7: Relative regret and Recall@Λ vs. Λ in case low-fidelity is equal to high-fidelity.



(a) Fidelity ratio $r = 1$ **(b) Fidelity ratio $r = 3$** **(c) Fidelity ratio $r = 5$** **(d) Fidelity ratio $r = 10$**

Figure 8: Relative regret vs. Λ and fidelity correlation (shown by numbers in the legend), MAG-Sm.



(a) correlation 0.5 **(b) correlation 0.75** **(c) correlation 0.9** **(d) correlation 1.0**

Figure 9: Relative regret vs. Λ and number of discretized score values (shown by numbers in legend) on MAG-Sm.

6.4 Effect of Budget (Λ)

We now commence our comparative evaluation of different methods, studying the effect of the interaction budget Λ on performance. Figure 7 shows our results on average relative regret and Recall@Λ (the share of found elements in the intended set) vs. Λ on all datasets, setting MF-ASC against the best method derived from the state of the art, GP_Lapl, and the LabelProp baseline. We consider an ideal scenario when low-fidelity is equal to high-fidelity. MF-ASC consistently outperforms both competitors with a significant improvement over GP_Lapl on all datasets. On the other hand, LabelProp, being a passive method, shows the worst performance.

6.5 Sensitivity to Fidelities Correlation

Thus far, we have used a low-fidelity function correlated to the simulated user input. Here, we study the sensitivity to the correlation between the low- and high-fidelity functions. Figure 8 reports our results vs. different handcrafted correlation values and also different fidelity ratios r, ranging from r=1 (Figure 8a) corresponding to one low-fidelity evaluation after each high-fidelity request, to r=10 (Figure 8d). The results indicate that with a reasonably high correlation value MF-ASC outperforms GP_Lapl, while with lower values it does not fare so well. Besides, the advantage of MF-ASC becomes evident at earlier interaction steps as more low-fidelity evaluations are performed (Figure 8c).

6.6 Sensitivity to Score Granularity

Our simulated user relevance score has hitherto been continuous. Nevertheless, real-world users provide granular input of a few grades [17, 26]. We now examine the sensitivity of our results to the granularity of user input. Figure 9 reports our results with

the simulated user input discretized, with both MF-ASC and GP_Lapl, for different granularities of this discretization as well as for different correlation values between high- and low-fidelity scores. Note that regret is still calculated with respect to continuous high-fidelity values. We also use continuous values for low-fidelity scores as they are specified by an internal model of the user which is expected to work with such continuous values.

Notably, a discretization granularity of 2 levels (i.e. binary scores) gives poor quality, whereas 3 levels achieve quality indistinguishable from that of 10 levels; that is also virtually identical as that of the continuous case, which we omit from the figure as its difference from the 10-level one is imperceptible. These results confirm that a continuous user function, which we have used in the rest of our simulated-user studies, yields results that we can also achieve with the kind of discrete input that a real-world users provide.

In the complementary problem of function interpolation, the benefits of using a multifidelity approach over a single-fidelity one were studied in [32]. That study shows that, depending on the cost ratio of the high-fidelity function to the low-fidelity function, and the correlation between them, the multifidelity approach is beneficial, whereas when the cost ratio or correlation is sufficiently low, multifidelity loses its advantage.

6.7 Scalability

Last, we evaluate MF-ASC on its ability to produce real-time answers with growing budget Λ. Figure 10 shows our results on time per user interaction (i.e., high-fidelity evaluation) vs. Λ, as well as its variance, on logarithmic axes. As expected (cf. Section 4.3), the time per interaction of MF-ASC grows cubically in the number of high-fidelity evaluations. As MF-ASC evaluates r=5 times more nodes than GP_Lapl, due to low-fidelity internal evaluations, it incurs additional computational overhead. However, this time remains within real-time performance. Even with the biggest dataset, FB-Comp, time per interaction is less than 1 second.

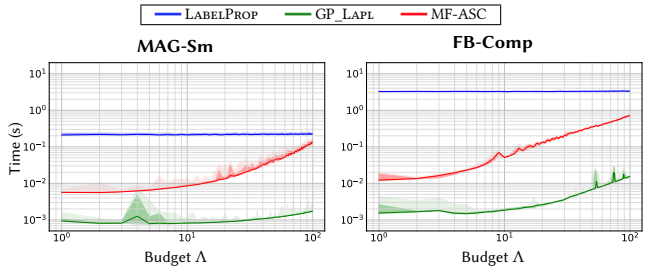


Figure 10: Time per iteration; shades denote 0.8 quantiles.

7 CONCLUSIONS

We proposed MF-ASC, an active search mechanism that integrates user feedback with information from an *internal evaluation function* via *cokriging* Gaussian interpolation, and thereby finds results the user wants with minimal user input. As MF-ASC performs similarity computations in a preprocessing step, it faces no scalability obstacle — its time complexity is cubic only in the size of the set of *scored* nodes; the core motivation of this work is to ensure that this set is kept small. Our experimental study on real and simulated user data shows that MF-ASC surpasses the state of the art, delivering highly relevant information after a few user interactions.

REFERENCES

[1] Natalia M. Alexandrov, John E Dennis Jr, Robert Michael Lewis, and Virginia Torczon. 1998. A Trust Region Framework for managing the use of approximation models in optimization. *Structural Optimization* 15, 1 (1998), 16–23.
[2] Evgeny Burnaev and Maxim Panov. 2015. Adaptive design of experiments based on Gaussian processes. In *SLDS*. 116–125.
[3] Apostolos N. Burnetas and Michael N. Katehakis. 1997. Optimal adaptive policies for Markov decision processes. *Math. Operations Research* 22, 1 (1997), 222–255.
[4] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. 2007. Information-theoretic metric learning. In *ICML*. 209–216.
[5] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.
[6] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. 2012. The Yahoo! music dataset and KDD-Cup '11. In *Proc. KDD Cup 2011*. 8–18.
[7] Alexander I. J. Forrester, András Sóbester, and Andy J. Keane. 2007. Multi-fidelity optimization via surrogate modelling. *Proc. R. Soc. A* 463, 2088 (2007), 3251–3269.
[8] Shawn E. Gano, John E. Renaud, and Brian Sanders. 2005. Hybrid variable fidelity optimization by using a kriging-based scaling function. *AIAA Journal* 43 (11 2005), 2422–2433.
[9] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff G. Schneider, and Richard P. Mann. 2012. Bayesian optimal active search and surveying. In *ICML*.
[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.
[11] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. 1998. MindReader: Querying databases through multiple examples. In *VLDB*. 218–227.
[12] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. 2016. Gaussian process bandit optimisation with multi-fidelity evaluations. In *NIPS*. 992–1000.
[13] Kirthevasan Kandasamy, Gautam Dasarathy, Barnabas Poczos, and Jeff Schneider. 2016. The multi-fidelity multi-armed bandit. In *NIPS*. 1777–1785.
[14] Remi R. Lam, Karen E. Willcox, and David H. Wolpert. 2016. Bayesian optimization with a finite budget: an approximate dynamic programming approach. In *NIPS*. 883–891.
[15] Yifei Ma, Tzu-Kuo Huang, and Jeff G. Schneider. 2015. Active search and bandits on graphs using sigma-optimality. In *UAI*. 542–551.
[16] Andrew March, Karen Willcox, and Qiqi Wang. 2011. Gradient-based multifidelity optimisation for aircraft design using bayesian model calibration. *Aeronautical Journal* 115, 1174 (2011), 729.
[17] Roderick P McDonald. 2013. *Test theory: A unified treatment.* Psychology Press.
[18] Mark E. J. Newman. 2006. Modularity and community structure in networks. *PNAS* 103, 23 (2006), 8577–8582.
[19] Matthias Poloczek, Jialei Wang, and Peter Frazier. 2017. Multi-information source optimization. In *NIPS*. 4288–4298.
[20] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.
[21] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. Vol. 1. The MIT Press.
[22] Paul Resnick and Hal R. Varian. 1997. Recommender Systems. *Commun. ACM* 40, 3 (1997), 56–58.
[23] Matthew Schultz and Thorsten Joachims. 2003. Learning a distance metric from relative comparisons. In *NIPS*. 41–48.
[24] Burr Settles. 2012. *Active Learning*. Morgan & Claypool Publishers.
[25] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. 2012. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Trans. Information Theory* 58, 5 (2012), 3250–3265.
[26] Stanley Smith Stevens. 1946. On the theory of scales of measurement. *Science* 103, 2684 (1946), 677–680.
[27] Simon Tong and Edward Chang. 2001. Support vector machine active learning for image retrieval. In *MULTIMEDIA*. 107–118.
[28] Laurens J. P. van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9 (2008), 2579–2605.
[29] Hastagiri P. Vanchinathan, Andreas Marfurt, Charles-Antoine Robelin, Donald Kossmann, and Andreas Krause. 2015. Discovering valuable items from massive data. In *KDD*. 1195–1204.
[30] Xuezhi Wang, Roman Garnett, and Jeff Schneider. 2013. Active search on graphs. In *KDD*. 731–738.
[31] Alexey Zaytsev and Evgeny Burnaev. 2017. Large scale variable fidelity surrogate modeling. *Annals of Mathematics and Artificial Intelligence* 81, 1 (2017), 167–186.
[32] Alexey Zaytsev and Evgeny Burnaev. 2017. Minimax approach to variable fidelity data interpolation. In *AISTATS*. 652–661.
[33] Hao Zhang and Wenxiang Cai. 2015. When doesn't cokriging outperform kriging? *Statist. Sci.* 30, 2 (2015), 176–180.
[34] Yehong Zhang, Trong Nghia Hoang, Bryan Kian Hsiang Low, and Mohan Kankanhalli. 2017. Information-based multi-fidelity Bayesian optimization. In *NIPS Workshop on Bayesian Optimization*.

A REPRODUCIBILITY DETAILS

Here, we report some details pertaining to our implementation and reproducibility. All our implementations and data used for experiments are available[8] online.

A.1 Tuning β_t for GP_Lapl

In order to make a fair comparison, we tune GP_Lapl on the best value of the exploration-exploitation tradeoff β_t. Recall that a large β_t favors exploration, skewing the selection towards nodes with high uncertainty. Figure 11 reports the results on the largest (FB-Comp) and the smallest (MAG-Sm) dataset; we witnessed similar performance for the other datasets. We note that $\beta_t \sim 0.01$ is the optimal choice for GP_Lapl; thus, we set $\beta_t = 0.01$.

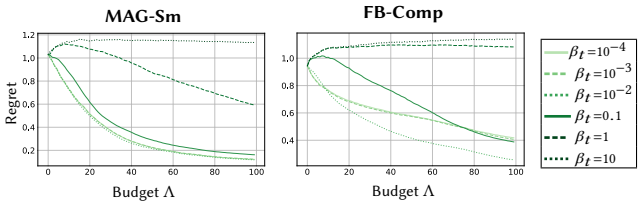


Figure 11: Regret of GP_Lapl vs. Λ and β_t.

[8] https://github.com/user526/BMFASC