Thompson Sampling for Optimizing Stochastic Local Search

Tong Yu[1], Branislav Kveton[2], and Ole J. Mengshoel[1]

[1] Electrical and Computer Engineering, Carnegie Mellon University, USA
tongy1@andrew.cmu.edu, ole.mengshoel@sv.cmu.edu
[2] Adobe Research, USA
kveton@adobe.com

Abstract. Stochastic local search (SLS), like many other stochastic optimization algorithms, has several parameters that need to be optimized in order for the algorithm to find high quality solutions within a short amount of time. In this paper, we formulate a stochastic local search bandit (`SLSB`), which is a novel learning variant of SLS based on multi-armed bandits. `SLSB` optimizes SLS over a sequence of stochastic optimization problems and achieves high average cumulative reward. In `SLSB`, we study how SLS can be optimized via low degree polynomials in its noise and restart parameters. To determine the coefficients of the polynomials, we present polynomial Thompson Sampling (`PolyTS`). We derive a regret bound for `PolyTS` and validate its performance on synthetic problems of varying difficulty as well as on feature selection problems. Compared to bandits with no assumptions of the reward function and other parameter optimization approaches, our `PolyTS` assuming polynomial structure can provide substantially better parameter optimization for SLS. In addition, due to its simple model update, `PolyTS` has low computational cost compared to other SLS parameter optimization methods.

1 Introduction

Stochastic optimization algorithms are among the most popular approaches to solving NP-hard and black box computational problems [33, 15]. Stochastic optimization algorithms include stochastic local search (SLS). Even in their simplest forms, SLS algorithms typically have several parameters that dramatically impact performance.

The problem of choosing parameters for a stochastic optimization algorithm is essential, since the optimal values for these parameters are problem-specific and have dramatic impact on the performance of the algorithm. As an example, consider the feature selection problem for sensor-based human activity recognition (HAR). The mobile sensor data for a given activity, for example jogging, can vary substantially from person to person and depending on where the mobile devices are placed on the body. Compared to training models on a large-scale general dataset, training models on a small personal dataset can yield better accuracy [22, 39]. Thus, it is better for the HAR model to be trained periodically when new labeled personal data is available. In such a sequence of problems, using fixed parameters for feature selection methods may not always provide the optimal accuracy. In this paper, we study how to optimize the parameters for stochastic optimization algorithms in such sequences of problems, in order to achieve the highest cumulative reward over time.

Several previous works could potentially be applied to optimize the parameters of SLS. `UCB1` [6] is used in adaptive operator selection [11, 12, 21]. Reinforcement learning is used in single parameter tuning for local search [7, 30]. To optimize the parameters of a general algorithm, there also exists a wide range of methods, such as racing algorithms (*e.g.*, `irace`) [8, 23], ParamILS [18], sequential model-based optimization (`SMAC`) [16, 17], gender-based genetic algorithm (GGA) [3], and Bayesian optimization (*e.g.*, `GPUCB` and `GPEIMCMC`) [35, 36]. However, most previous works have one or more of these limitations, when being applied in our problem:

1. **Not optimizing cumulative reward for a sequence of problems:** Most parameter optimization methods [8, 23, 18, 16, 17, 3, 14, 32, 26] are for best-arm identification, where they explore aggressively in fixed steps and then exploit. However, in our problem, we aim to maximize a long-term reward over a sequence of problems.[3]
2. **No regret bound or suboptimal regret bound:** Previous bandit methods in Adaptive Operator Selection [11, 12, 21] have no regret bound analysis. In fact, we show that directly applying a bandit method without parametric approximation leads to suboptimal regret in Section 4.2.
3. **Intensive computational cost:** Most methods have high computational cost (see Table 1). For example, inference of Gaussian process is computationally expensive in Bayesian optimization [35, 36].
4. **Optimizing only a single parameter for local search:** Previous work on reinforcement learning only tunes a single parameter for local search [7, 30]. However, it has been shown that multiple parameters should be tuned to optimize SLS [25].

This paper seeks to jointly address all four limitations above through our *stochastic local search bandit (*`SLSB`*)*. In `SLSB`, each set of parameters for SLS corresponds to a bandit arm. We assume that the learning problem at each time of training is drawn i.i.d. from some distribution over learning problems, which is unknown to the learning agent. When `SLSB` is applied to feature selection, there is a bandit, SLS, and a classifier. At each time t, they interact as follows: 1) The bandit chooses SLS's parameters $A \in [0, 1]^2$. 2) SLS find κ feature subsets using the chosen parameters A. 3) With each feature subset, a classifier is trained and evaluated by the accuracy on the data available at time t. 4) The maximum among the accuracies on κ feature subsets is the bandit reward.

`SLSB` is a framework for learning from adaptively chosen parameter sets with the objective of finding the best SLS parameter set. It differs from classical approaches [8, 23, 18, 16, 17, 3, 14, 32, 26] to parameter optimization in two aspects. First, the objective is to identify the best set of parameters without perfectly fitting a model, which may not be necessary. Second, the training parameter sets are chosen adaptively. We would like to stress that our goal of maximizing cumulative reward[3] is different from that of best

[3] As a motivation for our focus on maximizing cumulative reward, consider parameter tuning for feature selection on a sequence of HAR datasets. In this application, it is important to provide usable predictions even in the initial steps of tuning feature selection, else users may stop using the HAR system.

arm identification [5], where the learning can explore aggressively in initial steps and its performance is measured only at some fixed later step.

The contributions of this paper are as follows:

- We solve the problem of optimizing SLS parameters to achieve the highest cumulative reward over a sequence of problems by means of a multi-armed bandit, which we call a stochastic local search bandit (`SLSB`).
- Inspired by Markov chain analyses of SLS, we study polynomial Thompson Sampling (`PolyTS`), which learns low-degree polynomials in the noise and restart parameters of SLS. In addition, we show that the parametric approximation setting of `PolyTS` can effectively improve the regret bound of `SLSB`, compared to the setting without parametric approximation.
- Experimentally, we compare `PolyTS` to several state-of-the-art parameter optimization methods. We demonstrate that `PolyTS` can effectively find optimal SLS parameters leading to high reward (*i.e.*, high accuracy) in a sequence of feature selection problems. We show that `PolyTS` induces substantially lower computational cost compared to other prominent parameter tunning methods.

2 Related Work

2.1 Stochastic Local Search

Stochastic local search (SLS) algorithms are simple but broadly applicable [15], and among the best-performing algorithms for NP-hard problems including satisfiability [33, 15] as well as computing the most probable explanation [26, 27] and the maximum a posteriori hypothesis in Bayesian networks.

SLS algorithms are optimization methods that combine greedy, noisy, and restart heuristics among others. Noise and restart operations, combined with powerful initialization heuristics, have proven to be valuable additions to greedy search in many SLS algorithms [33, 32, 26, 4]. This raises the question of how to optimally balance between the greedy, noisy, and restart operations, ideally in a problem-specific manner.

One method to optimize SLS parameters is adaption. Hoos explores adaptive noise [14]. Adaptive noise and restart were recently integrated [25]. Ryvchin and Strichman advocate a related concept, namely localized restarts [32]. A second approach to optimizing SLS parameters is to perform a Markov chain analysis. Markov chain models of SLS have been used to compute expected hitting times [26, 25], which can be used for parameter optimization.

2.2 Parameter Optimization Methods

Reinforcement learning was used to optimize the performance of reactive search optimization [7] and local search [30]. Markov decision processes were used to optimize the prohibition value in reactive tabu search, the noise parameter in Adaptive Walksat, and the smoothing probability in reactive scaling and probabilistic smoothing algorithms [7]. R-learning was used to dynamically optimize noise in standard SAT local search algorithms [30]. These methods optimize algorithm parameters during execution.

Other algorithms for parameter optimization exist. Racing [8, 23] evaluates a set of candidate parameter configurations by discarding bad ones when statistically sufficient evidence is gathered against them. ParamILS [18] optimizes parameters based on Iterated Local Search. Sequential model-based optimization [16, 17] iterates between constructing a regression model and obtaining additional data for it. Previous works also model an algorithm's performance as samples from a Gaussian process [35, 36].

2.3 Bandits

A multi-armed bandit [20, 6] is a popular framework for learning to act under uncertainty. The framework has been successfully applied to many complex structured problems, ranging from stochastic combinatorial optimization to reinforcement learning. `UCB1` [6] is used in adaptive operator selection [11, 12, 21]. The proposed `SLSB` in this paper is, to our knowledge, the first work that formulates the problem of parameter learning in SLS as a multi-armed bandit problem. There are potentially other bandit algorithms suitable for `SLSB` [9, 24, 19]. Besides, Thompson Sampling [10] is applied to automatically choose the best technique and its associated hyper-parameters in a machine learning task [13]. Our novelty compared to previous bandit research is in the formulation of `SLSB` to optimize SLS, the parametric approximation adaptation of the existing bandit algorithms to our problem, the regret bound analysis, and their empirical applications in feature selection for HAR.

3 Background on Stochastic Local Search

We consider *stochastic local search (SLS)* [15] as search in the space of *bit-strings* $B = \{0, 1\}^N$. The goal is find the maximum of the *fitness function* $V : B \to \mathbb{R}$:

$$b^* = \arg\max_{b \in B} V(b).$$

The function V characterizes the problem being solved.

SLS is, in our case, parameterized by search parameters $A = (p_r, p_n)$ and the maximum number of optimization operations κ. SLS is initialized at a bit-string $b \in B$. The next, and any subsequent, SLS operation is either a restart operation, a noise operation, or a greedy operation. These operations are chosen randomly according to the following scheme [25]. SLS chooses a restart operation with probability p_r.

Definition 1. *(Restart Operation) The restart operation restarts SLS to a new random bit-string b.*

If SLS does not restart, then a noise operation is chosen with probability p_n.

Definition 2. *(Noise Operation) The noise operation randomly flips a bit in b.*

Finally, if SLS does not choose a noise operation, SLS performs a greedy operation.

Definition 3. *(Greedy Operation) The greedy operation chooses the highest-fitness neighbor of b. The neighbor of b is any bit-string that differs from b in one bit.*

The parameters $A = (p_n, p_r)$ have a dramatic impact on the performance of SLS. When $p_n = p_r = 0$, SLS behaves greedily. When $p_n = 1$ and $p_r = 0$, SLS is a random walk. The challenge is to set p_n and p_r, such that SLS finds a bit-string $\hat{b}$ such that $V(\hat{b})$ is close to $V(b^*)$. Let $b_1, \dots, b_\kappa$ be the sequence of bit-strings in the κ operations of SLS with parameters $A = (p_n, p_r)$. Then we define the *reward* of this step of the SLSB (see Section 4) as

$$f(A) = \max_{i \in [\kappa]} V(b_i) \,. \tag{1}$$

Note that f depends on both A and V, and also on the randomness in choosing the operations in SLS. To simplify notation, we do not make this dependence explicit.

We propose a variant of SLS, a learning agent, in the next section. The goal of the learning agent is to find $A = (p_n, p_r)$ that maximizes $\mathbb{E}\left[f(A)\right]$, where the expectation is taken with respect to both (i) randomly choosing V from some class of problems and (ii) the randomness of the operations in SLS.

4 Stochastic Local Search Bandit

We formulate the problem of learning optimal SLS parameters as a *stochastic local search bandit (*SLSB*)*. Formally, a stochastic local search bandit is a tuple $B = (\Theta, P)$, where $\Theta = [0,1]^2$ is the *space of arms*, all pairs of the restart and noise-operation probabilities in SLS; and P is a probability distribution over *reward functions* $f : \Theta \to \mathbb{R}$. For any arm $A \in \Theta$, we denote by $f(A)$ the *reward* of A under f, which is the same as in (1); and by $\bar{f}(A) = \mathbb{E}\left[f(A)\right]$ the *expected reward* of using SLS with parameters A.[4] The expectation is with respect to a potentially random fitness function, which represents different i.i.d. chosen problems from some set, and also the randomness in choosing the operations in SLS. We assume that $\bar{f}(A)$ is Lipschitz in A.

Let $(f_t)_{t=1}^n$ be an i.i.d. sequence of n reward functions drawn from P, where f_t is the reward function at time t. The learning agent interacts with our SLS algorithm as follows. At time t, it selects arm $A_t \in \Theta$, which depends on the observations of the agent up to time t. Then it observes the stochastic reward of arm A_t, $f_t(A_t)$, and receives it as a reward.

The goal of the learning agent is to maximize its cumulative reward over time. The quality of the agent's policy is measured by its *expected cumulative regret*. Let

$$A^* = \arg\max_{A \in \Theta} \bar{f}(A) \tag{2}$$

be the *optimal solution* in hindsight. Then the expected cumulative regret is

$$R(n) = \mathbb{E}\left[\sum_{t=1}^n R(A_t, f_t)\right] \,, \tag{3}$$

where

$$R(A_t, f_t) = f(A^*, f_t) - f(A_t, f_t) \tag{4}$$

is the *instantaneous stochastic regret* of the agent at time t. For simplicity of exposition, we assume that A^* is unique.

[4] For example, in HAR, $f(A)$ is the activity recognition accuracy of a particular machine learning model trained on a HAR dataset, when using SLS with parameters A for feature selection.

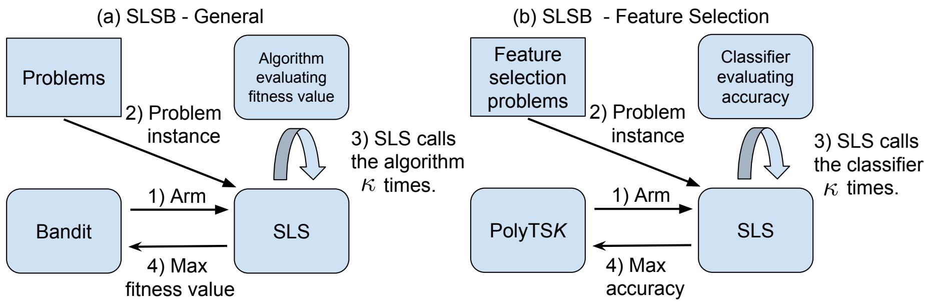
(a) SLSB - General
Problems
Algorithm evaluating fitness value
2) Problem instance
3) SLS calls the algorithm κ times.
Bandit
1) Arm
SLS
4) Max fitness value
(b) SLSB - Feature Selection
Feature selection problems
Classifier evaluating accuracy
2) Problem instance
3) SLS calls the classifier κ times.
PolyTSK
1) Arm
SLS
4) Max accuracy

Fig. 1: In (a), `SLSB` solves a sequence of stochastic optimization problems. At each step t, there are four actions: 1) Based on the fitness values up to steps $t-1$, the bandit algorithm chooses SLS's parameters, $A = (p_r, p_n)$. 2) We receive a problem instance. 3) Run SLS for κ operations on the problem with the chosen parameters. 4) Feed the maximum of the fitness value in the κ operations as the reward. The goal of `SLSB` is to maximize the cumulative accuracy over T steps, where T is the total number of steps. In (b), `SLSB` solves a sequence of feature selection problems. At each step t, 1) `PolyTS`K suggests the parameters to SLS. 2) We receive a dataset. 3) SLS searches κ feature subsets using the suggested parameters. Under the evaluation of a classifier on this dataset, the feature subset with the highest validation accuracy is selected. 4) This feature subset's corresponding validation accuracy is the reward $f(A)$ returned to `PolyTS`K.

4.1 `SLSB` in General and `SLSB`'s Application in Feature Selection

The different algorithms of `SLSB` and how those algorithms interact are illustrated in Figure 1. Figure 1(a) shows a general example of `SLSB` solving a sequence of stochastic optimization problems. One key research problem that we discuss in Section 4.2, Section 4.3, and Section 4.4 is how to design an efficient algorithm for the bandit in `SLSB` in Figure 1(a), such that the cumulative reward is maximized.

The `SLSB` can be applied in feature selection, as shown in Figure 1(b). We reduce the feature selection problems to optimization problems solved by SLS [25]. For training data with N dimension, if the ith bit is 1, we select the ith feature, else we ignore it. In total we have 2^N possible feature subsets (*i.e.*, bit-strings). At time t, we select arm A_t, the parameters of the SLS; and randomly divide the dataset at time t into the training, validation, and test sets. Then we run SLS for κ operations. The i-th operation of SLS is associated with bit-string b_i, an indicator vector of features. At operation i, we learn a classifier on the training set with features b_i and evaluate it by its accuracy $V(b_i)$ on the validation set. The SLS solution is the best performing b_i, as in (1). Our method can also be applied to other problems, not just feature selection. SLS is used for sparse signal recovery [29] and finding the most probable explanation in Bayesian networks [26, 27], where the SLS can also be optimized by `SLSB`.

4.2 Thompson Sampling (`TS`)

The key idea in our first design is to discretize the space of arms Θ into a set of points $\mathcal{A}$ and estimate $\bar{f}(A)$ separately in each $A \in \mathcal{A}$. We refer to each A as an *arm* and denote the number of arms by $|\mathcal{A}|$. We assume that $\mathcal{A}$ is an ε-grid with granularity ε, which satisfies $\|A - A'\|_\infty \leq \varepsilon$ for any $A, A' \in \mathcal{A}$. Since $\mathcal{A}$ is an ε grid on a two-dimensional space, $|\mathcal{A}| = 1/\varepsilon^2$.

Our first learning algorithm is a variant of *Thompson sampling (*`TS`*)* [37] for normally distributed rewards with σ^2 variance. We assume that the conjugate prior on the expected reward of each arm is $\mathcal{N}(0, \sigma_0^2)$, a normal distribution with a zero mean and σ_0^2 variance. At each step t, our `SLSB` algorithm operates in three stages. First, we sample the expected reward $\bar{f}_t(A)$ of each arm $A \in \mathcal{A}$ from its posterior. By our assumptions and Bayes rule, the expected reward of arm A at time t is distributed as

$$\bar{f}_t(A) \sim \mathcal{N}\left(\frac{\sigma^2_{T_{t-1}(A)}}{\sigma^2} T_{t-1}(A) \hat{f}_{T_{t-1}(A)}(A),\ \sigma^2_{T_{t-1}(A)}\right),$$

where $T_t(A)$ is the number of times arm A is chosen in t steps, $\hat{f}_s(A)$ is the average of s observed rewards of arm A, and $\sigma_s^2 = (\sigma_0^{-2} + s\sigma^{-2})^{-1}$ is the variance of the posterior after s observations. Second, we choose the arm with the highest posterior reward:

$$A_t = \arg\max{}_{A \in \mathcal{A}}\, \bar{f}_t(A)\,.$$

Finally, we observe $f_t(A_t)$ and receive it as a reward.

Proposition 1. *Let c be the Lipschitz factor of f and $\mathcal{A}$ be an ε-grid over Θ. Then for $\varepsilon = c^{-\frac{1}{2}} n^{-\frac{1}{4}}$, the regret of* `TS` *is $O(c^{\frac{1}{2}} n^{\frac{3}{4}})$.*

Proof. Since $\mathcal{A}$ is an ε grid on a two-dimensional space, $|\mathcal{A}| = 1/\varepsilon^2$. Thus, the regret of `TS` with respect to the best arm on the ε-grid is $O(\sqrt{n|\mathcal{A}|})$ [1]. Considering $\sqrt{n|\mathcal{A}|} = \sqrt{n/\varepsilon^2} = \sqrt{n}/\varepsilon$, the regret of `TS` with respect to the best arm is $O(\sqrt{n}/\varepsilon)$. Because the Lipschitz factor of f is c and $\mathcal{A}$ is an ε-grid, the expected rewards of the best arms in Θ and $\mathcal{A}$ differ by at most $c\varepsilon$. Therefore, the regret of `PolyTS` with respect to the best arm in Θ is $O(\sqrt{n}/\varepsilon) + c\varepsilon n$. Now substitute our suggest value of ε to minimize $O(\sqrt{n}/\varepsilon) + c\varepsilon n$, and our claim follows.

The regret of `TS` is suboptimal in n and we will show `PolyTS` with an improved regret bound in Section 4.3.

4.3 Polynomial Thompson Sampling (`PolyTS`)

The shortcoming of `TS` is that its regret is suboptimal in n. We now study how to learn a parametric approximation of $\bar{f}$. Based on existing work on SLS [26, 25] and our discussion in Section 4.4, we assume that $\bar{f}$ is a low-order polynomial over Θ.

Our learning algorithm is a variant of *linear Thompson sampling (*`LinTS`*)* [1] for learning polynomials of degree K. Let i-th entry of A be $A[i]$. We assume that the reward of arm $A = (A[1], A[2]) \in \Theta$ at time t is normally distributed as

$$f_t(A) \sim \mathcal{N}(\mathbf{x}_A \theta^*, \sigma^2)\,, \tag{5}$$

where $\mathbf{x}_A \in [0,1]^{1 \times L}$ is the feature vector of all possible $L = (K+1)(K+2)/2$ terms $A[1]^i A[2]^j$, and $i \geq 0$ and $j \geq 0$ are any integers such that $i + j \leq K$. When $K = 2$, we get a quadratic polynomial and:

$$\mathbf{x}_A = (A[1]^2, A[1]A[2], A[2]^2, A[1], A[2], 1)\,.$$

Algorithm 1 SLS algorithm (see Figure 1(a) and 1(b)).

```
Function SLS ($A$)
$p_n \leftarrow A[1]$; $p_r \leftarrow A[2]$;
Randomly initialize $b \in B = \{0, 1\}^N$   { See Section 3.}
$V_{\max} \leftarrow V(b)$
for SLS operation $i \leftarrow 2$ to $\kappa$ do
    if rand(0,1) $< p_r$ then  $b \leftarrow$ RestartOperation()   { See Definition 1. The rand(0,1) generates a uniformly distributed random variable in [0, 1].};
    else
        if rand(0,1) $< p_n$ then $b \leftarrow$ NoiseOperation($b$)   { See Definition 2.};
        else $b \leftarrow$ GreedyOperation($b, D$)   { See Definition 3.} ;
    end
    if $V_{max} < V(b)$ then  $V_{\max} \leftarrow V(b)$   { $V_{\max}$ is the $f(A)$ in Equation 1. } ;
end
Return $V_{\max}$
```

Algorithm 2 PolyTSK, which calls Algorithm 1 (see Figure 1(b)) .

```
Input : Parameters $\sigma$ and $\lambda^{-1}$ in (5) and (6), total number of steps $T$.
for Step $t \leftarrow 1$ to $T$ do
    $X_{t-1} \leftarrow (\mathbf{x}_{A_1}^\intercal, \dots, \mathbf{x}_{A_{t-1}}^\intercal)^\intercal$
    $F_{t-1} \leftarrow (f_1(A_1), \dots, f_{t-1}(A_{t-1}))^\intercal \in \mathbb{R}^{(t-1)\times 1}$
    $S_{t-1} \leftarrow (\sigma^{-2} X_{t-1}^\intercal X_{t-1} + \lambda I_L)^{-1}$
    $\bar{\theta}_{t-1} \leftarrow \sigma^{-2} S_{t-1} X_{t-1}^\intercal F_{t-1}$
    Sample the parameters of $\bar{f}$ from its posterior by $\theta_t \sim \mathcal{N}(\bar{\theta}_{t-1}, S_{t-1})$
    Choose the arm with the highest expected reward $A_t \leftarrow \arg\max_{A \in \mathcal{A}} \mathbf{x}_A \theta_t$
    Observe and record $f_t(A_t) \leftarrow$ SLS($A_t$)
end
```

The vector $\theta^* \in \mathbb{R}^{L \times 1}$ represents L unknown parameters of $\bar{f}$, which are shared by all arms. We assume that the conjugate prior on θ^* is

$$\mathcal{N}(\mathbf{0}, \lambda^{-1} I_L), \tag{6}$$

a normal distribution with a zero mean and isotropic λ^{-1} variance; where I_L denotes an $L \times L$ identity matrix. For degree K, we refer to our learning algorithm as PolyTSK.

At each step t, our algorithm operates in three stages. First, we sample the parameters of $\bar{f}$ from its posterior. Denote $\bar{\theta}_{t-1}$ as the mean of the parameters of $\bar{f}$ after $t-1$ steps. By our assumptions and Bayes rule, the posterior at time t is normally distributed as $\theta_t \sim \mathcal{N}(\bar{\theta}_{t-1}, S_{t-1})$, where the *covariance matrix* is $S_t = (\sigma^{-2} X_t^\intercal X_t + \lambda I_L)^{-1}$, the *mean* is $\bar{\theta}_t = \sigma^{-2} S_t X_t^\intercal F_t$, $X_t = (\mathbf{x}_{A_1}^\intercal, \dots, \mathbf{x}_{A_t}^\intercal)^\intercal$ is a $t \times L$ matrix of the features of all chosen arms in t steps, and $F_t = (f_1(A_1), \dots, f_t(A_t))^\intercal \in \mathbb{R}^{t \times 1}$ are all observed rewards in t steps. After we sample θ_t, we choose the arm with the highest expected reward with respect to θ_t:

$$A_t = \arg\max_{A \in \mathcal{A}} \mathbf{x}_A \theta_t \,. \tag{7}$$

Finally, we observe $f_t(A_t)$ as a reward. We bound the regret of PolyTS below.

Proposition 2. *Let c be the Lipschitz factor of f and $\mathcal{A}$ be an ε-grid over Θ. Then for $\varepsilon = L/(c\sqrt{n})$, the regret of* `PolyTS` *is $\tilde{O}(L\sqrt{n})$, where $\tilde{O}(\cdot)$ is the big-O notation up to logarithmic factors.*

Proof. The regret of `PolyTS` with respect to the best arm on the grid is $\tilde{O}(L\sqrt{n})$ [1]. Because the Lipschitz factor of f is c and $\mathcal{A}$ is an ε-grid, the expected rewards of the best arms in Θ and $\mathcal{A}$ differ by at most $c\varepsilon$. Therefore, the regret of `PolyTS` with respect to the best arm in Θ is $\tilde{O}(L\sqrt{n}) + c\varepsilon n$. Now substitute our suggested value of ε and our claim follows.

When the degree K is small, `PolyTS`K gives us a reasonable tradeoff between the bias and variance of our model. Asymptotically, we expect that `TS` can learn better approximations than `PolyTS` when the number of discretization points $|\mathcal{A}|$ is large, because `TS` does not make any assumptions on $\bar{f}$. However, in a finite time, we hypothesize that `PolyTS` can learn good-enough approximations at a much lower regret than `TS` because it learns fewer parameters. We study this issue empirically, using synthetic problems in Section 5 and real world problems in Section 6.

4.4 Discussion of the Polynomial Approximation

Markov chains were used to analytically optimize SLS [26, 25]. The expected hitting times for SLS is shown to be rational functions $\frac{P(p_n)}{Q(p_n)}$ of noise p_n, where $P(p_n)$ and $Q(p_n)$ are polynomials [26]. The analysis was extended to the analysis of both noise p_n and restart p_r parameters [25]. Although it was not mentioned explicitly, the hitting time can be written as a rational function $H(p_n, p_r) = \frac{P(p_n,p_r)}{Q(p_n,p_r)}$, where $P(p_n, p_r)$ and $Q(p_n, p_r)$ are polynomials of p_n and p_r. A possible way to optimize $H(p_n, p_r)$ is to compute p_n and p_r for $H'(p_n, p_r) = 0$. Unfortunately, obtaining p_n and p_r via $H'(p_n, p_r) = 0$ is difficult practically.

Minimizing the hitting time of SLS in step t of `SLSB` is closely related to minimizing the instantaneous stochastic regret in step t, $R(A_t, f_t)$, which is equivalent to maximizing reward $f(A_t, f_t)$ in step t in (4).

Based on the Stone-Weierstrass theorem, we have: if f is a continuous real-valued function defined on the range $[a, b] \times [c, d]$ and $\psi > 0$, then we have a polynomial function P in two variables x and y such that $|f(x, y) - P(x, y)| < \psi$ for all $x \in [a, b]$ and $y \in [c, d]$. So we can approximate the expected instantaneous reward $f(A_t, f_t)$ by polynomials in the noise and restart parameters, where $a = c = 0$, $b = d = 1$, $x = p_r$, and $y = p_n$. What degree polynomial P do we need to get a good approximation? We study this question experimentally, see Section 5 and Section 6.

5 Synthetic Experiments

5.1 Problems

All synthetic problems are defined on 20-bit bit-strings. Therefore, the cardinality of the search space in our synthetic problems is 2^{20}. These synthetic problems are designed to

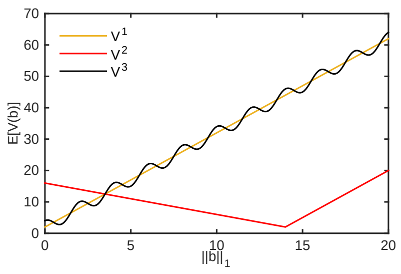

Fig. 2: 1D projections of the fitness functions in our synthetic problems, V^1, V^2 and V^3.

exercise the greedy, noise, and restart operations of SLS to varying degrees. All fitness functions are symmetric and therefore we can visualize them in a single dimension (Figure 2). The maximum of all our functions is at the all-ones bit-string, $\|b\|_1 = 20$.

The fitness function in the first problem is $V^1(b) = 3\|b\|_1 + 2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$ is a random scalar which is the same for all b, in any random realization of V^1. The maximum of $V^1(b)$ can be be found greedily from any b and for any ϵ. The fitness function in the second problem is $V^2(b) = \max\{16 - \|b\|_1, 3\|b\|_1 - 40\} + \epsilon$. The maximum of $V^2(x)$ can be found greedily if SLS starts at $\|b\|_1 \geq 14$ for any ϵ. Restarts are very beneficial in $V^2(x)$. The fitness function for the third problem is $V^3(b) = 3\|b\|_1 + 2 + 2\cos(\pi\|b\|_1) + \epsilon$. The maximum of $V^3(b)$ can be found by mixing greedy and noise operations from any b and for any ϵ.

The expected reward functions for all three problems are shown in Figure 3. Clearly, the optimal probability values p_r^* and p_n^* (in white regions) for these problems are at quite different values of the probabilities p_r and p_n.

5.2 Setting of Baseline Methods and Our Methods

For the synthetic problems, we use the following baseline methods. In `UCB1` [6], the UCB of arm A at time t is computed as $U_t(A) = \hat{f}_{T_{t-1}(A)}(A) + c_{t-1,T_{t-1}(A)}$, where $T_t(A)$ is the number of times that arm A is chosen in t steps, $\hat{f}_s(A)$ is the average of s observed rewards of arm A, $c_{t,s} = \gamma\sqrt{(1.5\log t)/s}$ is the radius of the confidence interval around $\hat{f}_s(A)$ after t steps, and γ is the difference between the maximum and minimum rewards. `irace` [23] evaluates a set of candidate parameters by discarding bad ones when statistically sufficient evidence has been gathered. `SMAC` [16] iterates between constructing a regression model and obtaining additional data based on this model. `GPEIMCMC` [35] uses Bayesian optimization to model SLS performance as samples from a Gaussian process. `GPUCB` [36] formulates parameter optimization as a bandit problem, where the reward function is sampled from a Gaussian process or has low RKHS norm. The selected baselines `irace`, `SMAC`, and `GPEIMCMC` have competitive performance [16] compared to other approaches, such as GGA [3] and ParamILS [18].

For those methods not optimizing cumulative regret (`irace`, `SMAC`, and `GPEIMCMC`), they explore N_{budget} steps and then exploit. We fine-tune their performance on each problem using different N_{budget}. This gives them a significant competitive advantage

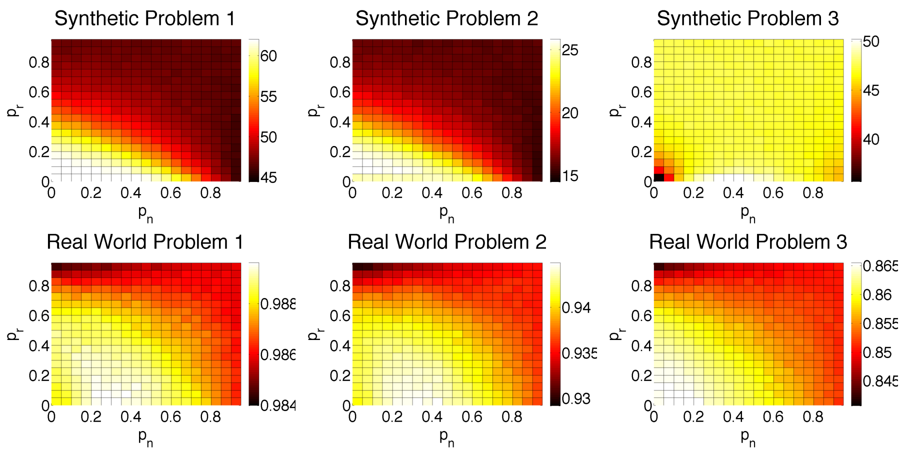

Fig. 3: The expected reward $\bar{f}$ of the three synthetic problems in Figure 2 (top row) and the three real world problems (bottom row): 1) Breast Cancer, 2) Wine and 3) Australian. The six problems are diverse, with different optimal p_r and p_n (lighter, meaning higher reward, is better). These are used as ground truth for evaluation purposes only, and not known to the optimization algorithms.

over our `SLSB` approach, which is not fine-tuned on each problem. For `irace` and `SMAC`, we try budget $N_{\text{budget}} \in \{200, 500, 1000, 2000, 5000, 10000, 20000\}$. For `GPEIMCMC`, we try $N_{\text{budget}} \in \{200, 500\}$.[5] For `GPUCB`, we set $N_{\text{budget}} = 2000$.[5] Finally, we report these methods' results at N_{budget} with the lowest cumulative regret in 2×10^4 steps. After `irace`, `SMAC`, and `GPEIMCMC` run out of budget, the slopes of the curves show the quality of the parameters they find (smaller slope is better here).

For `UCB1` and `TS`, we discretize the space of arms Θ on a 20×20 uniform grid. Let γ be the difference between the maximum and minimum rewards, which can be estimated from random samples. For `TS`, we set $\sigma_0 = \gamma$, because we expect that the learned parameters in `TS` are on the same order as the rewards. We put $\sigma = \frac{\gamma}{10}$, which means that the noise in observations of any arm $A \in \Theta$ is much lower than the range of rewards. For the same reasons, in `PolyTS`, $\sigma = \frac{\gamma}{10}$ and $\lambda^{-1} = \gamma^2$. This setting of γ is also used in `UCB1` to adjust for the range of rewards. We show results of `PolyTS2` and `PolyTS4`, which are variants of `PolyTS` for learning degree-2 (quadratic) and degree-4 polynomials respectively. We run all bandit methods for $n = 2 \times 10^4$ steps and compute their regret as in (3). We find A^* in (2) by discretizing the space of arms Θ on a 20 x 20 uniform grid. Results are averaged over 10 runs. We also show error bars indicating standard error, a measure of confidence in the estimate of the mean. For SLS in all experiments, we initialize b uniformly at random, and set $\kappa = 200$.

[5] We did not try other N_{budget} for `GPEIMCMC` and `GPUCB`, because their computation time is prohibitive.

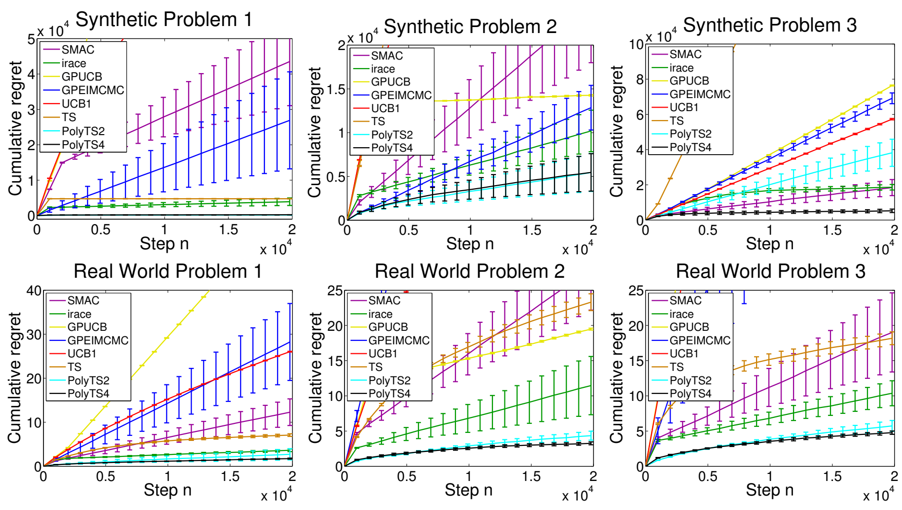

Fig. 4: The cumulative regret of different approaches in up to $n = 2 \times 10^4$ steps on the three synthetic problems V^1, V^2 and V^3 in Figure 2 and the three real world problems: 1) Breast Cancer, 2) Wine and 3) Australian.

5.3 Results

We compare the regret of all methods in Figure 4. We observe several major trends. First, in most cases our `PolyTS2` and `PolyTS4` learn as t increases in Figure 4. This is apparent from the concave shape of their regret plots. Second, we observe that the regrets of `PolyTS2` and `PolyTS4` are consistently and substantially lower than those of `UCB1` and `TS`. Third, `PolyTS4` performs better than `irace` and `SMAC`, while `PolyTS2` only outperforms `irace` and `SMAC` in synthetic problems V^1 and V^2. Fourth, `PolyTS` generally performs better than `GPEIMCMC` and `GPUCB`. `GPEIMCMC` performs better than `GPUCB`, which is consistent with previous results [35]. Finally, the error bars of `PolyTS4` are small, showing that `PolyTS4` is stable.

The regret of `PolyTS` is lower than the regrets of `UCB1` and `TS`, indicating that the polynomial approximation of the expected reward is reasonable. Compared to `PolyTS4`, `PolyTS2`'s higher regret in problem V^3 could be caused by the lightest and abruptly changing regions as showed in Figure 3. It may be difficult to approximate well those regions by quadratic polynomials. The reason why `GPEIMCMC` is performing worse here may be because we only try $N_{\text{budget}} \in \{200, 500\}$, due to high computational cost.

The parameter tuning times of `irace`, `SMAC`, `GPEIMCMC` , `GPUCB` and `PolyTS4` on problem V^3 are shown in Table 1. We use V^3 as an example and the tuning time comparison results are similar on V^1, V^2 and V^3. We have several observations. First, `PolyTS4` has the shortest running time. Second, the running times of `irace`, `SMAC`, and `PolyTS4` grow approximately linearly in N_{budget}. Third, the running times of methods based on Gaussian process grow in a super-linear fashion. When N_{budget} is large, these Gaussian process methods may not be suitable. These results suggest that `PolyTS4` has very low computational cost, compared to other parameter optimization methods.

Table 1: The parameter tuning time (in seconds) of different methods with different N_{budget}. We use a Linux server with two Intel Xeon E5530 2.40GHz CPUs and 24 GB memory. The running time of SLS is excluded and we only compare parameter tuning time. Results are averaged over 10 runs. Marker '-' means the experiment needs more than 40,000 seconds (about 11 hours).

	N_{budget}						
	200	500	1000	2000	5000	10000	20000
`irace`	2.07	3.32	5.00	8.14	17.95	35.62	74.86
`SMAC`	59.72	98.90	273.98	504.04	1,443.60	3,249.51	6,583.36
`GPEIMCMC`	1,975.73	10,309.59	39,716.12	-	-	-	-
`GPUCB`	14.87	84.19	577.38	4,284.76	-	-	-
`PolyTS4`	**0.06**	**0.13**	**0.21**	**0.39**	**0.90**	**1.73**	**3.51**

6 Real World Experiments

6.1 Problems: Feature Selection and Its Applications in HAR

In this section, we evaluate our methods on five real world feature selection problems: Breast Cancer (9 features, 700 samples), Wine (13 features, 178 samples), Australian (14 features, 690 samples), HAR data 1 (561 features, $3,433$ samples) [2] and HAR data 2 (416 features, $12,590$ samples) [34].[6]

For HAR, limiting the number of features reduces power consumption of mobile devices [38]. Thus, here we select $N_{\text{feature}} \leq 20$ features from engineered features.[7] We initialize SLS with $b = 0\ldots0$ and set $\kappa = 20$.[8] We simulate two scenarios when generating a sequence of feature selection problems. In each step, the training, validation and test data are sampled from (i) 10 of 30 users on HAR data 1 and (ii) data collected from one body position (left pocket, right pocket, wrist, upper arm, and belt) on HAR data 2.

6.2 Setting of Baseline Methods and Our Methods

For the small scale datasets, the settings of baselines and our methods are the same as in Section 5. For the large scale HAR experiments, we use the following baseline methods. The `irace` is used as baseline, as it outperforms `SMAC` in the small scale problems. The `GPEIMCMC` is also evaluated, as it outperforms `GPUCB` in the small scale problems. Besides, `GPEIMCMC` is more tunable with regard to different N_{budget}. The budget N_{budget} is smaller here than in the small scale problems. Two traditional feature selection methods are also compared to `PolyTS4`: a sparse learning based method `RFS` [28] and a similarity based method `reliefF` [31]. For the traditional methods, we try budget $N_{\text{budget}} \in \{150, 200, 250, 300\}$ and report the result for N_{budget} with the highest

[6] Breast Cancer, Wine and Australian can be found at `https://archive.ics.uci.edu/ml/datasets.html`.

[7] Feature descriptions for HAR data 1 are in [2]. The features for HAR data 2 include mean, variance and FFT values of sensors signals for each 1 second sliding window.

[8] If $b = 0\ldots0$ and $\kappa = 20$, we have $N_{\text{feature}} \leq 20$. The reason is that after $\kappa = 20$ SLS operations, the bit-string found by SLS can differ from the initial b in at most 20 bit.

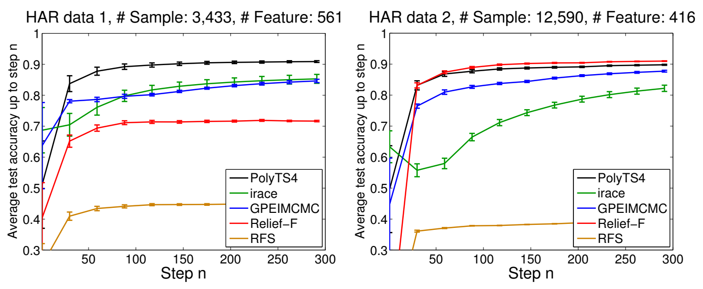

Fig. 5: The average test accuracy up to step n on HAR problems.

average cumulate reward over 300 steps. We control RFS and reliefF such that they select the same number of features as PolyTS4. We use decision tree as the classifier. We evaluate different methods based on test accuracy, since this is a better measure of generalization than the validation accuracy. In practice, A^* in (2) for larger dataset is computationally expensive to obtain, so it is difficult to calculate regret as evaluation criterion. On the HAR datasets, we thus report the average test accuracy up to step n. All results are averaged over 10 runs.

6.3 Results

Figure 4 shows the regret on the small scale datasets. We have similar observations as for the synthetic problems. The regrets of PolyTS2 and PolyTS4 are better than the regrets of irace, SMAC, GPEIMCMC, and other bandit methods.

The results on the HAR datasets are in Figure 5. In the very early steps, PolyTS4 has slightly lower average cumulative accuracy than irace and GPEIMCMC. As learning goes on, after about 10 steps, PolyTS4 is better than irace and GPEIMCMC. PolyTS is not better initially because it explores the parameter space. Later on, PolyTS exploits more and is better. Besides, PolyTS4 outperforms RFS and reliefF in most cases, except that it has slightly lower accuracy (1.2%) than that of reliefF on HAR data 2. The results suggest that PolyTS4 can find useful feature subsets that have high cumulative accuracy on average. Further, PolyTS4 also provides reasonably good accuracy in initial steps, which is very useful in real world HAR scenarios.

7 Conclusion

This paper is motivated by the strong performance of SLS in a multitude of applications [15], along with the need to carefully optimize SLS search parameters in order to achieve such performance. To optimize SLS parameters to achieve the highest cumulative reward over a sequence of stochastic optimization problems, we propose SLSB. We study different bandit methods to implement SLSB, specifically: UCB1, TS, and PolyTS. Both UCB1 and TS work, but their regret bound is suboptimal. Motivated by Markov chain analyses of SLS and the observation that our reward function is smooth in the coordinates of arms, we approximate the reward by polynomial functions, and present

PolyTS with an improved regret bound. Compared to other parameter optimization methods, we show superior performance of PolyTS on synthetic and real-world problems. In addition, we show that the model update of PolyTS is very simple, such that PolyTS requires lower computational cost than other parameter optimization methods.

Acknowledgment

This material is based, in part, upon work supported by the National Science Foundation under Grant No. 1344768.

References

1. Agrawal, S., Goyal, N.: Thompson sampling for contextual bandits with linear payoffs. In: ICML-2013. pp. 127–135 (2013)
2. Anguita, D., Ghio, A., Oneto, L., Parra Perez, X., Reyes Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: ESANN-2013. pp. 437–442 (2013)
3. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: CP-2009. pp. 142–157 (2009)
4. Audemard, G., Simon, L.: Refining restarts strategies for SAT and UNSAT. In: Principles and Practice of Constraint Programming, pp. 118–126 (2012)
5. Audibert, J.Y., Bubeck, S.: Best arm identification in multi-armed bandits. In: COLT-2010. pp. 13–p (2010)
6. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47, 235–256 (2002)
7. Battiti, R., Campigotto, P.: An investigation of reinforcement learning for reactive search optimization. In: Autonomous Search, pp. 131–160 (2011)
8. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: GECCO-2002. pp. 11–18 (2002)
9. Cappé, O., Garivier, A., Maillard, O.A., Munos, R., Stoltz, G.: Kullback–leibler upper confidence bounds for optimal sequential allocation. The Annals of Statistics 41(3), 1516–1541 (2013)
10. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: NIPS 24, pp. 2249–2257 (2011)
11. DaCosta, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: GECCO-2008. pp. 913–920 (2008)
12. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Annals of Mathematics and Artificial Intelligence 60(1-2), 25–64 (2010)
13. Hoffman, M., Shahriari, B., Freitas, N.: On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In: Artificial Intelligence and Statistics. pp. 365–374 (2014)
14. Hoos, H.H.: An adaptive noise mechanism for WalkSAT. In: AAAI-2002. pp. 655–660 (2002)
15. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann, San Francisco (2005)
16. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proc. of LION-5 (2011)

17. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: Time-bounded sequential parameter optimization. In: Proc. of LION-4 (2010)
18. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. JAIR 36, 267–306 (2009)
19. Kleinberg, R.D.: Nearly tight bounds for the continuum-armed bandit problem. In: NIPS. vol. 17, pp. 697–704 (2004)
20. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6(1), 4–22 (1985)
21. Li, K., Fialho, A., Kwong, S., Zhang, Q.: Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 18(1), 114–130 (2014)
22. Lockhart, J.W., Weiss, G.M.: The benefits of personalized smartphone-based activity recognition models. In: SDM-2014. pp. 614–622 (2014)
23. López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration (2011)
24. Maillard, O.A., Munos, R.: Online learning in adversarial lipschitz environments. Machine Learning and Knowledge Discovery in Databases pp. 305–320 (2010)
25. Mengshoel, O.J., Ahres, Y., Yu, T.: Markov chain analysis of noise and restart in stochastic local search. In: IJCAI. pp. 639–646 (2016)
26. Mengshoel, O.J.: Understanding the role of noise in stochastic local search: Analysis and experiments. Artificial Intelligence 172(8), 955–990 (2008)
27. Mengshoel, O.J., Wilkins, D.C., Roth, D.: Initialization and restart in stochastic local search: Computing a most probable explanation in bayesian networks. IEEE Transactions on Knowledge and Data Engineering 23(2), 235–247 (2011)
28. Nie, F., Huang, H., Cai, X., Ding, C.H.: Efficient and robust feature selection via joint l 2, 1-norms minimization. In: NIPS. pp. 1813–1821 (2010)
29. Pal, D.K., Mengshoel, O.J.: Stochastic cosamp: Randomizing greedy pursuit for sparse signal recovery. In: ECML/PKDD (1). pp. 761–776 (2016)
30. Prestwich, S.: Tuning local search by average-reward reinforcement learning. In: Learning and Intelligent Optimization, pp. 192–205. Springer (2007)
31. Robnik-Šikonja, M., Kononenko, I.: Theoretical and empirical analysis of relieff and rrelieff. Machine learning 53(1-2), 23–69 (2003)
32. Ryvchin, V., Strichman, O.: Local restarts in SAT. Constraint Programming Letters (CPL) 4, 3–13 (2008)
33. Selman, B., Levesque, H., Mitchell, D.: A new method for solving hard satisfiability problems. In: AAAI. pp. 440–446 (1992)
34. Shoaib, M., Scholten, H., Havinga, P.J.: Towards physical activity recognition using smartphone sensors. In: UIC/ATC-2013. pp. 80–87. IEEE (2013)
35. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: NIPS. pp. 2951–2959 (2012)
36. Srinivas, N., Krause, A., Seeger, M., Kakade, S.M.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: ICML-10. pp. 1015–1022 (2010)
37. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25(3-4), 285–294 (1933)
38. Yu, M.C., Yu, T., Wang, S.C., Lin, C.J., Chang, E.Y.: Big data small footprint: the design of a low-power classifier for detecting transportation modes. Proceedings of the VLDB Endowment 7(13), 1429–1440 (2014)
39. Yu, T., Zhuang, Y., Mengshoel, O.J., Yagan, O.: Hybridizing personal and impersonal machine learning models for activity recognition on mobile devices. In: MobiCASE-2016. pp. 117–126 (2016)