Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

A COMPUTATIONAL STUDY OF PROBABILISTIC BRANCH AND BOUND WITH MULTILEVEL IMPORTANCE SAMPLING

Hao Huang

Department of Industrial
Engineering and Management
Yuan Ze University
Taoyuan, Taiwan

Pariyakorn Maneekul
Danielle F. Morey
Zelda B. Zabinsky

Department of Industrial and Systems Engineering
University of Washington
Seattle WA, USA

Giulia Pedrielli

School of Computing and Augmented Intelligence
Arizona State University
Tempe AZ, USA

ABSTRACT

Probabilistic branch and bound (PBnB) is a partition-based algorithm developed for level set approximation, where investigating all subregions simultaneously is a computational costly sampling scheme. In this study, we hypothesize that focusing branching and sampling on promising subregions will improve the efficiency of the PBnB algorithm. Two variations to Original PBnB are proposed: Multilevel PBnB and Multilevel PBnB with Importance Sampling. Multilevel PBnB focuses its branching on promising subregions that are likely to be maintained or pruned, as opposed to Original PBnB that branches more subregions. Multilevel PBnB with Importance Sampling attempts to further improve this efficiently by combining focused branching with a posterior distribution that updates iteratively. We present numerical experiments using benchmark functions to compare the performance of each PBnB variation.

1 INTRODUCTION

Level set identification is an active field of research. The problem of estimating level sets of black-box objective functions arises in a wide range of applications, including monitoring environmental parameters (Rahimi et al. 2004) and aircraft configuration (Priem et al. 2020). Sequential learning is one method to solve this problem. Most sequential learning algorithms use a stochastic process model such as a Gaussian process (GP) as a surrogate model, as in Shekhar and Javidi (2019). Other methods include the use of Bayesian neural networks (Ha et al. 2020), linear bandits (Mason et al. 2021), and problem reduction (Bachoc et al. 2021). Level set approximation via Probabilistic Branch and Bound has been useful in designing a simulated water distribution network for a large city (Tsai et al. 2018), and in optimizing screening and treatment decisions for hepatitis C over a 40 year projected time horizon (Huang et al. 2016).

Probabilistic Branch and Bound (PBnB) is a partition-based method for level set approximation that uses sampling, branching and classification of subregions to provide a collection of subregions that form the level set approximation. The PBnB algorithm, found in Huang and Zabinsky (2013) and Zabinsky

and Huang (2020), seeks a target level set associated with a target quantile, and iteratively updates its confidence interval on the value of the target quantile. PBnB classifies subregions as maintained (contained in the target level set) and pruned (no intersection with the target level set) with statistical confidence, and updates its collection of current undecided subregions that do not have statistical confidence to be maintained or pruned. A finite-time analysis of PBnB provides probability bounds on incorrectly pruning and maintaining subregions on each iteration. This result quantifies the quality of the solution and helps inform a decision maker when to stop the algorithm with an acceptable quality of the solution.

As illustrated in Figure 1, the original PBnB (Algorithm A) has two primary components. The first component is to sample uniformly over the current subregions (Step 1A), and to provide an interval estimate of the objective function value for the target quantile, denoted $y(\delta)$ where δ is the user-input for the target quantile (Step 2A). The second component (Steps 3A, 4A, and 5A) focuses on classifying subregions into maintained or pruned, and branching undecided subregions for more sampling to glean more information.

The main computational challenge of Original PBnB stems from making too many function evaluations and branching too many subregions. The branching scheme in Original PBnB branches **all** of the current subregions. This scheme leads to a proliferation of smaller subregions that requires a large number of samples to confirm maintaining and pruning. This motivates the concept of Multilevel PBnB to branch **only** the promising best and worst subregions. The idea is that identifying the best subregions increases the chance of maintaining, and identifying the worst subregions increases the chance of pruning, with fewer subregions and fewer function evaluations. In this study, we hypothesize that focusing more on the promising subregions will improve the efficiency of the PBnB algorithm.

We also note that Original PBnB samples **uniformly** on the current subregions, which is a conservative sampling distribution that is useful in estimating $y(\delta)$ with confidence intervals. We consider a form of importance sampling where more samples are taken in promising subregions. The interval estimation of $y(\delta)$ must be adapted to account for non-uniform sampling. We hypothesize that sampling on a posterior distribution based on the lowest observed function values in subregions (in contrast to uniform sampling) will improve efficiency.

We define two variations to Original PBnB, namely Multilevel PBnB that modifies the branching strategy, and Multilevel PBnB with Importance Sampling that additionally uses a posterior distribution to focus on promising subregions. Figure 1 highlights the differences between Original PBnB and the two variations.

We present computational results to answer three research questions. First, we determine if branching on only promising subregions will improve efficiency. Second, we examine the sensitivity of the algorithms to two parameter values that impact branching. Third, we determine if sampling on a posterior distribution based on the lowest observed function values will improve efficiency. To answer the questions above, we conduct a computational experiment on Original PBnB (Algorithm A), Multilevel PBnB (Algorithm B) and Multilevel PBnB with Importance Sampling (Algorithm C) while varying algorithm parameters over test functions in varying dimensions.

2 Algorithm Details of PBnB Variations for Level Set Approximation

We follow the notation of Zabinsky and Huang (2020). PBnB aims to approximate a level set with respect to a performance function $f(x)$ of a black-box simulation model. The optimization problem is

$$\min_{x \in S} f(x),$$

where $f(x) : S \to \mathbb{R}$, and $S \subset \mathbb{R}^{dim}$. The decision variable x is a vector in dim dimensions, and the values may be integer or real-valued. We are interested in the δ quantile associated with $f(x)$, denoted $y(\delta)$,

$$y(\delta) = \operatorname*{argmin}_{y} \{P(f(X) \leq y) \geq \delta\}, \quad \text{for } 0 < \delta < 1,$$

where X is a random variable uniformly distributed on the domain S. We let $L(\delta)$ be our desired set of best δ-quantile solutions, where

$$L(\delta) = \{x \in S : f(x) \leq y(\delta)\}, \quad \text{for } 0 < \delta < 1.$$

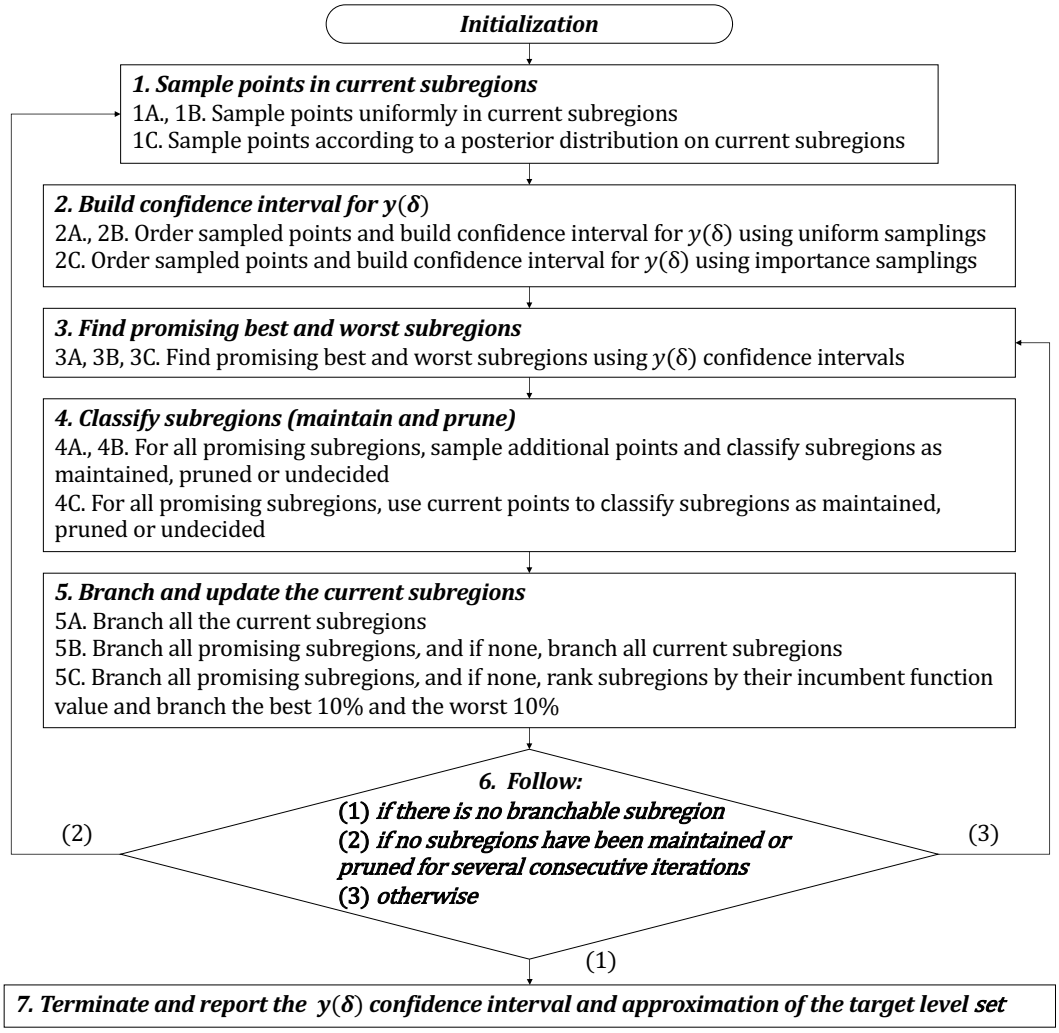


Figure 1: Procedure of PBnB variations for level set approximation: A is Original PBnB, B is Multilevel PBnB, and C is Multilevel PBnB with Importance Sampling. The target quantile δ is a user-input, and the estimated objective function value is denoted $y(\delta)$.

The input parameters to PBnB defined by the user include: $\delta, \alpha, \varepsilon, B, k_b$, and c. The parameter δ, $0 < \delta < 1$, is used to define a δ-quantile for the target level set. For example, the user may be interested in the set of solutions in the best 10%, in which case $\delta = 0.1$.

The following two parameters, α and ε, are used to determine the quality of the level set approximation. The approximation can be wrong in two ways: it could prune a portion of the level set or it could maintain some area that is not in the level set. The parameter α, $0 < \alpha < 1$, is used in the confidence level of the estimation of $y(\delta)$ and in the probabilities of incorrectly pruning or maintaining. The choice of α will influence the sample size. As the confidence level $(1-\alpha)$ increases, a larger sample size is needed. The

parameter $\varepsilon > 0$ is the volume of solutions that can be tolerated to be categorized incorrectly. We also expect that a high confidence level (low α) will have fewer mistakes, since the probability of the incorrect volume exceeding ε decreases.

The analytical results of the algorithm (Huang and Zabinsky 2013; Zabinsky and Huang 2020) provide confidence intervals on the quantile estimation $y(\delta)$, and probability bounds on incorrectly pruning a volume of size ε, and on incorrectly maintaining a volume of size ε, respectively. To paraphrase, on each iteration, the probability that the volume of the incorrectly pruned region is no greater than ε is bounded by $(1-\alpha)^4$. Similarly, the probability that the volume of the incorrectly maintained region is no greater than ε is bounded by $(1-\alpha)^4$.

The branching scheme is defined by parameter B, $B \geq 2$, which is the number of evenly sized subregions to create by subdividing the longest dimension of the subregion. The parameter k_b is the maximum number of consecutive inner-loop iterations without maintaining or pruning before returning to Step 1, and c is the incremental sample size for sampling on the current subregions in Step 1.

Let $\Sigma^M_{iter}, \Sigma^P_{iter}$ and Σ^C_{iter} be collections of subregions that are maintained, pruned, and currently undecided (the set of subregions that are not pruned or maintained), respectively, on iteration $iter$.

In Step 1, Algorithms A and B sample points according to a uniform distribution on current subregions, whereas Algorithm C uses a posterior distribution based on the lowest function value observed in a subregion. In Step 2, the expressions to estimate the target quantile $y(\delta)$ with confidence intervals differ when points are sampled uniformly or using a posterior distribution. Step 3 is the same for all variations, and uses the confidence intervals from Step 2.

In Step 4, Algorithms A and B implement additional sampling in the best and worst subregions to statistically confirm classification of subregions as maintained or pruned. Algorithm C does not add more sample points in Step 4, but waits until enough points are accumulated through the posterior distribution to classify subregions with the same level of confidence.

In Step 5, Algorithm A branches *all* current subregions resulting in subregions that are always the same size, whereas Algorithms B and C with multi-level branching branch only the promising subregions that results in subregions with different sizes. In Step 6, Algorithms A and B may return to Step 3, based on their value of k_b, but Algorithm C never returns to Step 3 in an inner loop. Algorithm C relies on the importance sampling instead of an inner loop. In the numerical experiments, $k_b = 1$ for Algorithm C.

PBnB for level set approximation proceeds until all subregions are either maintained, pruned, or reach a user-defined minimum size that we term unbranchable. In this paper, we determine a subregion is "unbranchable" when its volume is less than a fraction of the volume of the feasible set S. If the stopping criteria has not been met, the algorithm checks whether it has been successful in maintaining and/or pruning. If no subregions have been maintained or pruned in k_b consecutive inner-loop iterations, it returns to Step 1. Otherwise, it goes to Step 3.

Upon termination, PBnB outputs a confidence interval on the target quantile $y(\delta)$ and the subregions that have been maintained as an approximation of the target level set.

Probabilistic Branch and Bound (PBnB) for Level Set Approximation

Step 0. Initialization: Input user-defined parameters, $\delta, \alpha, \varepsilon, B, k_b,$ and c. Also, initialize the maintain, prune, and current subregion collections as $\Sigma^M_1 = \emptyset$, $\Sigma^P_1 = \emptyset$, $\Sigma^C_1 = \{S\}$. Set $\delta_1 = \delta$, $c_1 = c$, and the iterative counter $iter = 1$.

Step 1. Sample points in current subregions: For the current subregions in Σ^C_{iter}, sample c additional points over all current subregions in Σ^C_{iter} such that the total number of points in Σ^C_{iter} is c_{iter}.
1A and 1B: To obtain an additional sample point that is uniformly distributed, randomly choose subregion σ_i from the subregions in Σ^C_{iter} with probability $p_i = v(\sigma_i)/v(\Sigma^C_{iter})$, where $v(\cdot)$ denotes the dim-dimensional volume of a set. Then, uniformly generate a sample point within the chosen subregion σ_i. Update the number of points that have been sampled in σ_i as N^i. Note that $\Sigma_{i=1}^{||\Sigma^C_{iter}||} N^i = c_{iter}$.

1C: To obtain an additional sample point using importance sampling, randomly choose subregion σ_i from the subregions in Σ^C_{iter} with probability $\tilde{p}_i$, where

$$\tilde{p}_i = \begin{cases} v(\sigma_i)/v(\Sigma^C_{iter}) & \text{if } iter = 1 \\ \left(\left(\tilde{f}^*_i - \hat{f}^* + 1\right) \sum_{j=1,\ldots,||\Sigma^C_{iter}||} \left(\frac{1}{\tilde{f}^*_j - \hat{f}^* + 1} \right) \right)^{-1} & \text{if } iter > 1, \end{cases}$$

where $\tilde{f}^*_i$ is the lowest sampled value in subregion σ_i and $\hat{f}^* = \min_{i=1,\ldots,||\Sigma^C_{iter}||}\{\tilde{f}^*_i\}$. Then, uniformly generate a sample point within the chosen subregion σ_i. Update the number of points that have been sampled in σ_i as N^i. Note that $\Sigma_{i=1}^{||\Sigma^C_{iter}||} N^i = c_{iter}$.

Step 2. Build confidence interval for $y(\delta)$: Order all sampled points, $z_{(1)},\ldots,z_{(c_{iter})}$, in all current subregions in Σ^C_{iter} by their function values, so that

$$f(z_{(1)}) \leq \ldots \leq f(z_{(c_{iter})}).$$

2A and 2B: Calculate the lower and upper bounds of δ_{iter} as

$$\delta_{iter,l} = \delta_{iter} - \frac{v(\Sigma^P_{iter})\varepsilon}{v(S)v(\Sigma^C_{iter})} \quad \text{and} \quad \delta_{iter,u} = \delta_{iter} + \frac{v(\Sigma^M_{iter})\varepsilon}{v(S)v(\Sigma^C_{iter})}.$$

Then calculate the confidence interval with lower and upper bounds as

$$CI_l = f(z_{(r)}) \quad \text{and} \quad CI_u = f(z_{(s)}),$$

where r and s are selected by

$$\max r : \sum_{i=0}^{r-1} \binom{c_{iter}}{i} (\delta_{iter,l})^i (1-\delta_{iter,l})^{c_{iter}-i} \leq \frac{\alpha_{iter}}{2}, \text{ and}$$

$$\max s : \sum_{i=0}^{s-1} \binom{c_{iter}}{i} (\delta_{iter,u})^i (1-\delta_{iter,u})^{c_{iter}-i} \geq 1 - \frac{\alpha_{iter}}{2},$$

where $\alpha_{iter} = \alpha/B^{iter}$. The estimate of $y(\delta)$ is

$$\hat{y}(\delta) = (CI_l + CI_u)/2.$$

2C: Following Chu and Nakayama (2012) , the δ_{iter} quantile estimator is

$$\hat{y}(\delta) = \tilde{F}^{-1}_{c_{iter}}(\delta_{iter}) = f(z_{(i_{\delta_{iter}})}),$$

where $\tilde{F}^{-1}_{c_{iter}}(\delta_{iter})$ is determined empirically from $f(z_{(i_{\delta_{iter}})})$, and $i_{\delta_{iter}}$ is the smallest integer for which $\sum_{i=1}^{i_{\delta_{iter}}} L(z_{(i)}) \geq \delta_{iter} c_{iter}$, and $L(z_{(i)})$ depends on the subregion σ_j containing $z_{(i)}$, that is

$$L(z_{(i)}) = \frac{\tilde{p}_j}{p_j},$$

and $\tilde{p}_j$ and p_j are given in Step 2. The $100(1-\alpha)\%$ confidence interval for $\hat{y}(\delta)$ is $[CI_l, CI_u]$ where the lower and upper bounds are

$$\begin{aligned} CI_l &= \hat{y}_{iter}(\delta) - z_{1-\alpha/2}\tilde{\kappa}_{\delta_{iter},c_{iter}}/\sqrt{c_{iter}}, \\ CI_u &= \hat{y}_{iter}(\delta) + z_{1-\alpha/2}\tilde{\kappa}_{\delta_{iter},c_{iter}}/\sqrt{c_{iter}}, \end{aligned}$$

and $z_{1-\alpha/2}$ is from a standard normal distribution. We compute $\tilde{\kappa}_{\delta_{iter},c_{iter}}$ by taking the product of $\bar{\phi}_{\delta_{iter},c_{iter}}(a)$ and $\tilde{\psi}_{\delta_{iter},c_{iter}}$,

$$\tilde{\kappa}_{\delta_{iter},c_{iter}} = \bar{\phi}_{\delta_{iter},c_{iter}}(a) \cdot \tilde{\psi}_{\delta_{iter},c_{iter}}$$

for some constant $a > 0$ ($a = 0.1$ in the experiments). The estimate of variance constant $\tilde{\psi}_{\delta_{iter},c_{iter}}$ is

$$\tilde{\psi}_{\delta_{iter},c_{iter}} = \left(\frac{1}{c_{iter}} \sum_{i=1}^{c_{iter}} I(f(z_{(i)}) \leq \hat{y}_{iter}(\delta, S))\, L(z_{(i)})^2 \right) - \delta_{iter}^2,$$

and

$$\bar{\phi}_{\delta_{iter},c_{iter}}(a) = \tilde{\phi}_{\delta_{iter},c_{iter},1}(a) + \tilde{\phi}_{\delta_{iter},c_{iter},2}(a)$$

where

$$\begin{aligned} \tilde{\phi}_{\delta_{iter},c_{iter},1}(a) &= \frac{\tilde{F}^{-1}_{c_{iter}}(\delta_{iter} + a/\sqrt{c_{iter}}) - \tilde{F}^{-1}_{c_{iter}}(\delta_{iter})}{2a/\sqrt{c_{iter}}} \\ \tilde{\phi}_{\delta_{iter},c_{iter},2}(a) &= \frac{\tilde{F}^{-1}_{c_{iter}}(\delta_{iter} + a/\sqrt{c_{iter}}) - \tilde{F}^{-1}_{c_{iter}}(\delta_{iter} - a/\sqrt{c_{iter}})}{2a/\sqrt{c_{iter}}}. \end{aligned}$$

Step 3 Find promising best and worst subregions:
3A, 3B and 3C: In each subregion σ_i in Σ^C_{iter}, order all sampled points, $x_{i,(1)}, \ldots, x_{i,(N^i)}$, by their function values, so that $f(x_{i,(1)}) \leq \ldots \leq f(x_{i,(N^i)})$. Construct the collections of promising best and worst subregions $\mathscr{P}_b$ and $\mathscr{P}_w$ using the quantile confidence interval as

$$\mathscr{P}_b = \{\sigma_i |\ f(x_{i,(N^i)}) < CI_l,\ \text{for } \sigma_i \in \Sigma^C_{iter}, i \in 1, \ldots, ||\Sigma^C_{iter}||\}$$

$$\mathscr{P}_w = \{\sigma_i |\ f(x_{i,(1)}) > CI_u,\ \text{for } \sigma_i \in \Sigma^C_{iter}, i \in 1, \ldots, ||\Sigma^C_{iter}||\}.$$

Step 4. Classify subregions (maintain and prune):
4A, 4B: For all promising subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$, uniformly sample additional points such that the total number of points is N^i_k where k is the level of subregion σ_i,

$$N^i_k = \left\lceil \frac{\ln\left(\frac{\alpha}{B^k}\right)}{\ln\left(1 - \frac{\varepsilon}{v(S)}\right)} \right\rceil.$$

To provide a cap on N^i_k, let $N^i_k \leftarrow \min\{N^i_k, 100^{dim} v(\sigma_i)/v(S)\}$. In each subregion, reorder all of its N^i_k sampled points, $x_{i,(1)}, \ldots, x_{i,(N^i_k)}$, by their function values so that $f(x_{i,(1)}) \leq \ldots \leq f(x_{i,(N^i_k)})$. Then update the maintaining indicator functions M_i, for $\sigma_i \in \mathscr{P}_b$, and the pruning indicator functions P_i, for $\sigma_i \in \mathscr{P}_w$, as

$$M_i = \left\{ \begin{array}{ll} 1, & \text{if } f(x_{i,(N^i_k)}) < CI_l \\ 0, & \text{otherwise} \end{array} \right\} \quad \text{and} \quad P_i = \left\{ \begin{array}{ll} 1, & \text{if } f(x_{i,(1)}) > CI_u \\ 0, & \text{otherwise} \end{array} \right\}$$

Update the maintained set $\Sigma^M_{iter+1} \leftarrow \Sigma^M_k \bigcup_{i=\mathscr{P}_b : M_i = 1} \sigma_i$, the pruned set $\Sigma^P_{iter+1} \leftarrow \Sigma^P_k \bigcup_{i=\mathscr{P}_w : P_i = 1} \sigma_i$, and the current set Σ^C_{iter+1} to no longer include maintained and pruned subregions.
4C: For all promising subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$, calculate $\tilde{\alpha}_i$ using the current number of observed points in subregion σ_i, N^i,

$$\tilde{\alpha}_i = B^k e^{N^i} \left(1 - \frac{\varepsilon}{v(S)}\right)$$

and k is the level of subregion σ_i. Maintain the subregions in $\mathscr{P}_b$ and prune the subregions in $\mathscr{P}_w$ if $\tilde{\alpha}_i < \alpha$. Update the maintained set Σ^M_{iter+1}, pruned set Σ^P_{iter+1}, and current set Σ^C_{iter+1}, accordingly.

Step 5. Branch and update current subregions:
5A: Branch all remaining subregions in Σ^C_{iter}.
5B: Branch all promising subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$. If $\mathscr{P}_b$ and $\mathscr{P}_w$ are empty, then branch all remaining subregions in Σ^C_{iter}.
5C: Branch all promising subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$. If $\mathscr{P}_b$ and $\mathscr{P}_w$ are empty, rank the subregions in Σ^C_{iter} by their incumbent function value and branch the best 10% and worst 10%.

Step 6. Decision:
6A, 6B, 6C: If all subregions $\sigma_i \in \Sigma^C_{iter}$ are not branchable, go to Step 7 and terminate the algorithm. If there are still branchable subregions in Σ^C_{iter}, and subregions have been maintained or pruned in less than k_b inner-loop iterations, go to Step 3 to continue classifying. Otherwise, if no subregions have been maintained or pruned in k_b consecutive inner-loop iterations, then set

$$\delta_{iter+1} = \frac{\delta v(S) - v(\Sigma^M_{iter})}{v(\Sigma^C_{iter})} \quad \text{and set} \quad c_{iter+1} = c_{iter} + c,$$

and increment the counter $iter \leftarrow iter + 1$, and go to Step 1.

Step 7. Output results:
7A, 7B, 7C: Output $\hat{y}(\delta, S)$, $[CI_l, CI_u]$ and maintained subregions in Σ^M_{iter} on the last iteration.

3 Computational Study

We ran computational experiments on three algorithms: Original PBnB (Algorithm A), Multilevel PBnB (Algorithm B) and Multilevel PBnB with Importance Sampling (Algorithm C). All three algorithms were run on three different test functions, the Rosenbrock problem, the centered sinusoidal problem, and the shifted sinusoidal problem, for dimensions 2, 5, 7, and 10. The test problems are defined in the appendix. For each test condition, 10 replications were performed with common random number seeds and the results averaged. The average number of function evaluations until the first subregion is maintained is the main metric of interest. We also examined other performance measures namely, number of subregions generated, volume of the maintained, pruned and undecided subregions, and incumbent function value. However, due to space limitations we present only the number of function iterations required to reach the first maintained region as the easiest to interpret and perform statistical analysis. For all test conditions, we set the common PBnB parameter values, $\delta = 0.2$, $\alpha = 0.1$, $\varepsilon = 0.025(\text{vol}(S))$, $c = dim * 100$, and the unbranchable size $= 0.025(\text{vol}(S))$. For Algorithms A and B, the branching parameter B was tested with $B = \{2,4\}$ and the parameter on successful consecutive classification k_b was tested with $k_b = \{1,2,3\}$. For Algorithm C, $B = 2$ and $k_b = 1$.

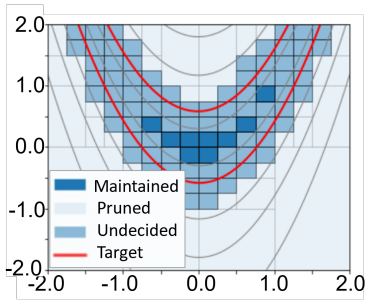
4 Results

We first compare Original PBnB (Algorithm A) with Multilevel PBnB (Algorithm B) in Section 4.1 to study the impact of multi-level branching. We conduct a sensitivity analysis on parameters B and k_b. In Section 4.2, we compare Algorithms A, B, and C to study the impact of importance sampling and multi-level branching.

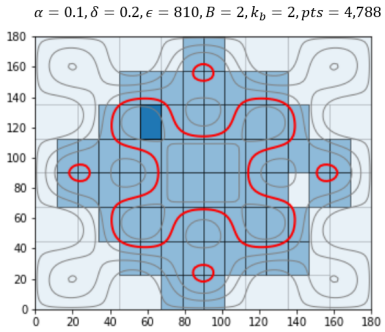
4.1 Comparing Original and Multilevel PBnB

We first examine the key differences between Original and Multilevel PBnB by plotting the subregions for a two-dimensional case and by plotting the lowest sampled point as the algorithm progresses. Figure 2 illustrates Algorithms A and B approximating a 20% level set after 10 iterations on three test functions

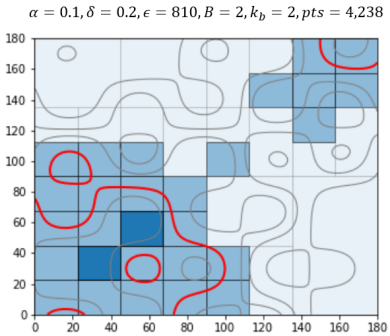
(see Appendix) in two dimensions. Notice that the subregions of Algorithm A are of the same size on each iteration, whereas the subregions in Algorithm B are of different sizes. In these cases, Multilevel PBnB was able to maintain more subregions by the tenth iteration.



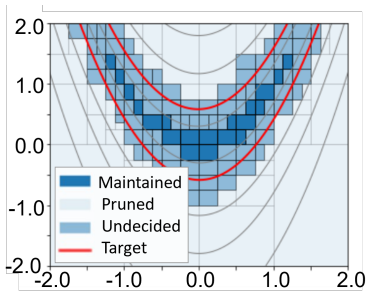
(a) Original PBnB (Algorithm A) on Rosenbrock.



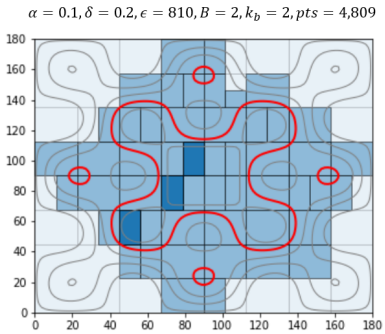
(b) Original PBnB (Algorithm A) on centered sinusoidal.



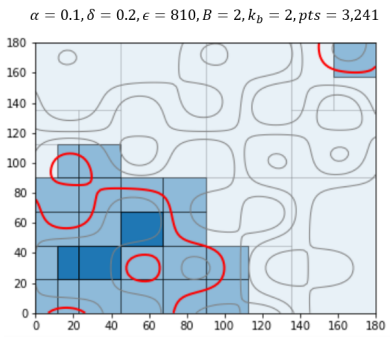
(c) Original PBnB (Algorithm A) on shifted sinusoidal.



(d) Multilevel PBnB (Algorithm B) on Rosenbrock.



(e) Multilevel PBnB (Algorithm B) on centered sinusoidal.



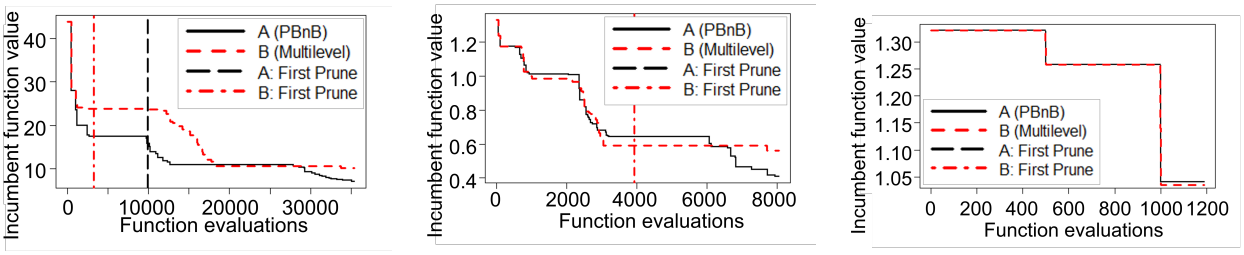
(f) Multilevel PBnB (Algorithm B) on shifted sinusoidal.

Figure 2: Approximating the 0.2-quantile level set (contour shown in red) on the tenth iteration of Original and Multilevel PBnB for three test functions in two dimensions.

The lowest function value that has been sampled (i.e., the incumbent solution) is plotted until the first subregion is maintained in Figure 3 for the three test functions in five dimensions with $B = 2$ and $k_b = 2$. As can be seen in Figure 3a, on the Rosenbrock test function Original PBnB is able to find a lower incumbent solution faster, but Multilevel PBnB is able to start pruning subregions sooner. However, on the centered sinusoidal function in Figure 3b, the incumbent values are very close, while Multilevel PBnB is still able to prune faster. On the shifted sinusoidal function, Multilevel PBnB is able to maintain a subregion even before pruning. This suggests that Multilevel PBnB is able to prune and maintain faster by narrowing in on promising subregions, whereas Original PBnB focuses on searching the entire space.

Table 1 provides the average over 10 replications of the number of function evaluations until first maintaining a subregion. The lowest mean number of function evaluations is shown in **bold**. As can be seen in the table, Multilevel PBnB outperforms Original PBnB on all of the 10 dimensional problems, and all but 4 out of 36 instances when $B = 2$.

Figure 4a shows the main effects for a statistical analysis of five factors, including test function, algorithm, dimension, parameter B, and parameter k_b. While dimension has a strong effect, the main effects plot shows that Multilevel PBnB performs slightly better than Original PBnB overall, and that parameter $B = 2$ is better than $B = 4$. In Figure 4b, the interaction between B and dimension shows that the sensitivity to B is magnified when dimension equals 10 (i.e., $B = 4$ performs worse than $B = 2$). The



(a) Rosenbrock (b) Centered sinusoidal (c) Shifted Sinusoidal

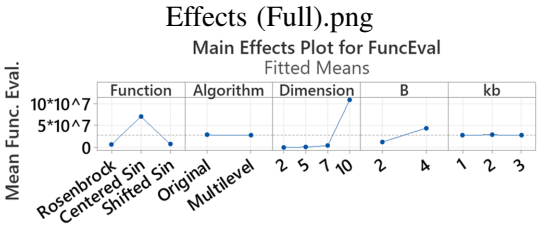
Figure 3: Incumbent function value plots with $B = 2$, $k_b = 2$, and dimension 5. Results are plotted until Algorithm B maintains its first subregion. The number of function evaluations until the first pruned subregion is denoted by the vertical dashed line for Algorithm A and vertical dot-dashed line for Algorithm B.

Table 1: Compare PBnB algorithms A and B on test function, dimension, B, and k_b.

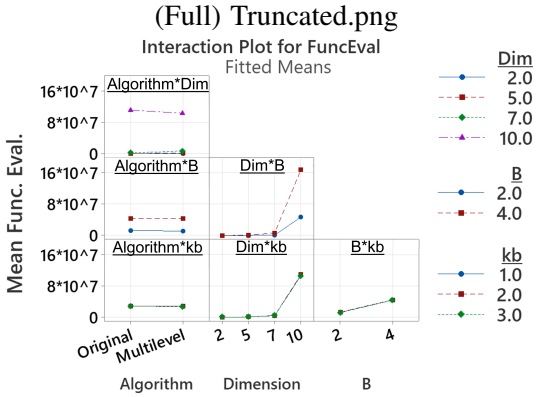
			Rosenbrock		**Centered Sinusoidal**		**Shifted Sinusoidal**	
			Original (Alg. A)	**Multilevel (Alg. B)**	**Original (Alg. A)**	**Multilevel (Alg. B)**	**Original (Alg. A)**	**Multilevel (Alg. B)**
dim	B	k_b	**Mean**	**Mean**	**Mean**	**Mean**	**Mean**	**Mean**
2	2	1	4,610	**3,830**	5,327	**4,476**	3,289	**2,667**
		2	4,688	**3,761**	5,508	**5,151**	3,376	**2,974**
		3	4,610	**4,351**	**5,755**	5,776	3,579	**2,930**
	4	1	5,748	**4,614**	6,700	**5,425**	5,102	**3,863**
		2	5,626	**4,626**	6,236	**5,049**	4,641	**3,926**
		3	5,525	**4,620**	6,398	**5,129**	4,785	**3,896**
5	2	1	74,715	**74,656**	323,781	**251,539**	2,988	**d2,187**
		2	76,852	**68,039**	291,674	**229,151**	3,106	**1,788**
		3	76,708	**48,131**	**249,553**	279,794	4,807	**2,263**
	4	1	**145,473**	179,222	**559,576**	1,834,108	172,785	**112,301**
		2	**126,895**	190,236	**525,015**	2,487,640	**147,492**	163,636
		3	**118,935**	188,129	**539,768**	1,566,468	141,887	**100,622**
7	2	1	1,134,184	**835,671**	1,836,140	**1,492,808**	**3,696**	3,782
		2	1,086,704	**706,705**	1,771,899	**1,261,945**	**3,823**	3,839
		3	1,134,184	**712,355**	1,760,036	**787,492**	5,151	**4,443**
	4	1	**517,694**	2,135,826	**1,865,113**	28,839,991	1,650,064	**1,621,246**
		2	**490,097**	2,040,385	**1,874,063**	30,983,735	**1,778,808**	1,798,716
		3	**456,404**	2,471,402	**1,831,739**	34,239,413	1,728,589	**1,519,452**
10	2	1	17,856,257	**11,039,015**	134,267,137	**118,484,057**	96,176	**71,516**
		2	16,068,785	**9,756,255**	131,429,897	**108,491,654**	121,254	**99,491**
		3	16,106,384	**7,820,062**	133,526,508	**110,648,929**	133,590	**120,259**
	4	1	42,914,162	**28,671,025**	414,277,547	**410,393,757**	57,813,728	**52,524,442**
		2	42,810,004	**24,514,233**	418,521,575	**412,343,257**	58,319,132	**57,820,349**
		3	49,700,106	**27,012,891**	409,989,531	**406,472,047**	61,957,231	**43,001,913**

performance is relatively insensitive to parameter k_b. We also performed a statistical analysis separately for both algorithms, and $B = 2$ performed significantly better than $B = 4$ for both algorithms. Again, k_b is relatively insensitive.

Figure 5 presents the statistical analysis fixing $B = 2$. This analysis found that Multilevel PBnB outperforms Original PBnB with 95% confidence.

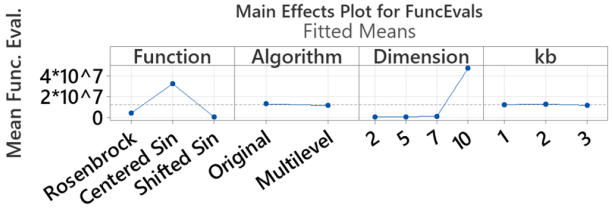


(a) Main effects plot.



(b) Interaction plots.

Figure 4: Plots showing the effects of test function, algorithm, dimension, B, and k_b



Effects B=2.png

Figure 5: Main effects plot for fixed $B = 2$.

4.2 Algorithm C: Multilevel PBnB with Importance Sampling

A third algorithm was developed to improve upon Multilevel PBnB by using importance sampling to update a posterior distribution. Multilevel PBnB with Importance Sampling was performed on Rosenbrock, Centered Sinusoidal, and Shifted Sinusoidal test problems for dimensions 2, 5, 7, and 10 with $B = 2$ (and $k_b = 1$) for 10 replications. These results are summarized alongside Original and Multilevel PBnB in Table 2. Multilevel PBnB with Importance Sampling (Algorithm C) performs better than both Multilevel PBnB and Original PBnB in terms of lower number of function evaluations to first maintain a subregion, which is statistically significant with 95% confidence.

5 Conclusion

In this study, we proposed and compared three variations of PBnB for level set approximation, namely Original PBnB, Multilevel PBnB, and Multilevel PBnB with Importance Sampling. Original PBnB involves branching the problem space until subregions can be pruned or maintained with statistical confidence. The proliferation of subregions and the extensive number of sample points required to classify subregions as maintained or pruned can be problematic, especially at high dimensions. Multilevel PBnB improves the algorithm by only branching on the most promising subregions, while adding importance sampling improves the algorithm by focusing sampling density on the subregions with better observed function values. Our numeric experiment demonstrates that Multilevel PBnB performs better than Original PBnB with statistical confidence in terms of number of function evaluations required to first maintain a subregion, and adding importance sampling performs even better. Additionally, our study indicates that the parameter value of B, the number of subregions partitioned into with each branch, greatly affects the performance of

Table 2: Comparing the mean number of function evaluations until the first maintaining a subregion for the three versions of PBnB where $B = 2$ and k_b=1 for Algorithm A (Original PBnB) and Algorithm B (Multilevel PBnB), and $B = 2$ and k_b=1 for Algorithm C (Multilevel PBnB with Importance Sampling)

	dim	Original (Alg. A) Mean	Multilevel (Alg. B) Mean	Importance Samp. (Alg. C) Mean
Rosenbrock	2	4,610	3,830	**1,527**
	5	74,715	74,656	**31,782**
	7	1,134,184	835,671	**215,721**
	10	17,856,257	11,039,015	**10,765,121**
Centered Sin.	2	5,327	4,476	**2,498**
	5	323,781	251,539	**187,795**
	7	1,836,140	1,492,808	**1,129,481**
	10	134,267,137	118,484,057	**56,145,091**
Shifted Sin.	2	3,289	2,667	**1,270**
	5	2,988	2,187	**1,490**
	7	3,696	3,782	**1,629**
	10	96,176	71,156	**29,965**

the algorithms, where $B = 2$ results in significantly lower function evaluations required to first maintain a subregion.

In future work, we hope to further improve on the PBnB algorithm to allow for more efficiency at higher dimensions. We plan to extend our sensitivity analysis to examine how the number of sample points per iteration, c, impacts algorithm efficiency. We also plan to utilize Gaussian processes to update the posterior distribution and provide an even more efficient algorithm.

A Appendix: Test Function Definitions

Rosenbrock Problem: The global minimum is at $(1,\ldots,1)$ and $f(x^*) = 0$.

$$\min_{x\in[-2,2]} f(x) \text{ where } f(x) = 0.1 \sum_{i=1}^{dim-1} \left((1-x_i)^2 + 100\left(x_{i+1} - x_i^2\right)^2 \right) \text{ and } x \in \mathbb{R}^{dim}$$

Centered Sinusoidal Problem: The global optimum is at $x^* = (90,\ldots,90)$ and $f(x^*) = 0$.

$$\min_{x\in[0,180]} f(x) \text{ where } f(x) = -\left[2.5 \prod_{i=1}^{dim} \sin\left(\frac{\pi x_i}{180}\right) + \prod_{i=1}^{dim} \sin\left(\frac{\pi x_i}{180}\right) \right] + 3.5 \text{ and } x \in \mathbb{R}^{dim}$$

Shifted Sinusoidal Problem: The global optimum is at $x^* = (30,\ldots,30)$ and $f(x^*) = 0$.

$$\min_{x\in[0,180]} f(x) \text{ where } f(x) = -\left[2.5 \prod_{i=1}^{dim} \sin\left(\frac{\pi(x+60)}{180}\right) + \prod_{i=1}^{dim} \sin\left(\frac{\pi(x+60)}{36}\right) \right] + 3.5 \text{ and } x \in \mathbb{R}^{dim}$$

REFERENCES

Bachoc, F., T. Cesari, and S. Gerchinovitz. 2021. "The Sample Complexity of Level Set Approximation". In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*. April 13th-15th, held virtually, 424-432.

Chu, F., and M. K. Nakayama. 2012. "Confidence Intervals for Quantiles when Applying Variance-reduction Techniques". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 22(2):1–25.

Ha, H., S. Gupta, S. Rana, and S. Venkatesh. 2020. "High Dimensional Level Set Estimation with Bayesian Neural Network". In *Proceedings of the AAAI Conference on Artificial Intelligence*. February 2nd-9th, held virtually, 12095-12103.

Huang, H., and Z. B. Zabinsky. 2013. "Adaptive Probabilistic Branch and Bound with Confidence Intervals for Level Set Approximation". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 4146–4157. Washington, DC: Institute of Electrical and Electronics Engineers, Inc.

Huang, H., Z. B. Zabinsky, Y. Li, and S. Liu. 2016. "Analyzing hepatitis C screening and treatment strategies using Probabilistic Branch and Bound". In *Proceedings of the 2016 Winter Simulation Conference (WSC)*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, T. H. E. Zhou, and S. E. Chick, 2076–2086. Institute of Electrical and Electronics Engineers, Inc.

Mason, B., R. Camilleri, S. Mukherjee, K. Jamieson, R. Nowak, and LalitJain. 2021. "Nearly Optimal Algorithms for Level Set Estimation". *arXiv*:1–32. https://doi.org/10.48550/arXiv.2111.01768.

Priem, R., H. Gagnon, I. Chittick, S. Dufresne, Y. Diouane, and N. Bartoli. 2020. "An Efficient Application of Bayesian Optimization to an Industrial MDO Framework for Aircraft Design". In *AIAA Aviation 2020 Forum*. June 15-19, held virtually, 1-16.

Rahimi, M., R. Pon, W. Kaiser, G. Sukhatme, D. Estrin, and M. Srivastava. 2004. "Adaptive Sampling for Environmental Robotics". In *Proceedings of IEEE International Conference on Robotics and Automation*. April 26th-May 1st, New Orleans, LA, 3537-3544.

Shekhar, S., and T. Javidi. 2019. "Multiscale Gaussian Process Level Set Estimation". In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. April 16th-18th, Naha, Okinawa, Japan, 3283–3291.

Tsai, Y.-A., G. Pedrielli, L. Mathesen, Z. B. Zabinsky, H. Huang, A. Candelieri, and R. Perego. 2018. "Stochastic Optimization for Feasibility Determination: An Application to Water Pump Operation in Water Distribution". In *Proceedings of 2018 Winter Simulation Conference (WSC)*, edited by M. Rabe, A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1945–1956. Institute of Electrical and Electronics Engineers, Inc.

Zabinsky, Z. B., and H. Huang. 2020. "A Partition-based Optimization Approach for Level Set Approximation: Probabilistic Branch and Bound". In *Women in Industrial and Systems Engineering: Key Advances and Perspectives on Emerging Topics*, edited by A. E. Smith, 113–155. Springer Nature.

AUTHOR BIOGRAPHIES

HAO HUANG is an Assistant Professor in the Department of Industrial Engineering and Management at the Yuan Ze University. His Ph.D. degree is from the University of Washington in Industrial and Systems Engineering. His research interests include simulation optimization, simulation modeling, data analysis, and applications in Manufacturing and Healthcare Management. His email address is haohuang@saturn.yzu.edu.tw and his website is https://www.iem.yzu.edu.tw/labfile/Simulation/English-members.html.

PARIYAKORN MANEEKUL is a PhD student in the Department of Industrial and Systems Engineering at the University of Washington. She is also a researcher at Chula Social Innovation at Chulalongkorn University. Her research interest includes optimization under uncertainty and the interplay of statistical learning methods and optimization. Her email address is parim@uw.edu.

DANIELLE F. MOREY is a PhD student in the Department of Industrial and Systems Engineering at the University of Washington studying under the mentorship of Dr. Zabinsky. Her research interest include global optimization, multi-fidelity modeling, and multi-objective optimization. Her email is dmorey43@uw.edu.

ZELDA B. ZABINSKY is a Professor in the Department of Industrial and Systems Engineering at the University of Washington. She is an INFORMS Fellow and an IISE Fellow. Her Ph.D. is from the University of Michigan, in Industrial and Operations Engineering. Her research interests are in global optimization under uncertainty for complex systems, and she has worked in many application areas. Her email address is zelda@uw.edu. Her website is https://ise.washington.edu/facultyfinder/zelda-zabinsky.

GIULIA PEDRIELLI is an Assistant Professor in the School of Computing and Augmented Intelligence at the Arizona State University. She holds a Ph.D. degree in Mechanical Engineering from the Politecnico di Milano. Her research interests include global optimization, random algorithms, engineering applications in manufacturing, cyber-physical systems, and computational biology. She is a member of IEEE and INFORMS. Her email address is giulia.pedrielli@asu.edu. Her website is https://www.gpedriel.com/.