Black-box combinatorial optimization using models with integer-valued minima

Bliek, Laurens; Verwer, Sicco; de Weerdt, Mathijs

DOI
10.1007/s10472-020-09712-4

Publication date
2020

Document Version
Final published version

Published in
Annals of Mathematics and Artificial Intelligence

Annals of Mathematics and Artificial Intelligence
https://doi.org/10.1007/s10472-020-09712-4

Black-box combinatorial optimization using models with integer-valued minima

Laurens Bliek[1] · **Sicco Verwer**[1] · **Mathijs de Weerdt**[1]

Accepted: 4 September 2020 Published online: 19 September 2020

Abstract
When a black-box optimization objective can only be evaluated with costly or noisy measurements, most standard optimization algorithms are unsuited to find the optimal solution. Specialized algorithms that deal with exactly this situation make use of surrogate models. These models are usually continuous and smooth, which is beneficial for continuous optimization problems, but not necessarily for combinatorial problems. However, by choosing the basis functions of the surrogate model in a certain way, we show that it can be guaranteed that the optimal solution of the surrogate model is integer. This approach outperforms random search, simulated annealing and a Bayesian optimization algorithm on the problem of finding robust routes for a noise-perturbed traveling salesman benchmark problem, with similar performance as another Bayesian optimization algorithm, and outperforms all compared algorithms on a convex binary optimization problem with a large number of variables.

Keywords Surrogate models · Bayesian optimization · Black-box optimization

1 Introduction

Traditional optimization techniques such as first order methods or branch and bound make use of a known mathematical formulation of the objective function, for example by calculating the derivative or a lower bound. However, many objective functions in real-life situations have no complete mathematical formulation. For example, smart grids or railways are complex networks where every decision influences the whole network in such a way that the objective cannot be easily captured in one mathematical description. In such applications, we can observe the effect of decisions either in real life, or by running a simulation. Waiting for such a result can take some time, or may have some other cost associated with it. Furthermore, the outcome of two observations with the same decision variables may be different due to randomness that may be present in the real-life scenario, or due to

✉ Laurens Bliek
l.bliek@tudelft.nl

[1] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

artificial randomness in the simulation. Such problems have been approached using methods such as black-box or Bayesian optimization [1], simulation-based optimization [2], and derivative-free optimization [3]. Here, a model fits the relation between decision variables and objective function, and then standard optimization techniques are used on the model instead of the original objective. These so-called surrogate modeling techniques have been applied successfully to continuous optimization problems in signal processing [4], optics [4], machine learning [5], robotics [6], and more. However, it is still an on-going research question on how these techniques can be applied effectively to combinatorial optimization problems. A common approach is to simply round to the nearest integer, a method that is known to be sub-optimal in traditional optimization, and also in black-box optimization [7]. Another option is to use discrete surrogate models from machine learning such as regression trees [8] or linear model trees [9]. Although powerful, this makes both model fitting and optimization computationally expensive.

This work describes an approach where the surrogate model is still continuous, but where finding the optimum of the surrogate model gives an integer solution. The main contributions are as follows:

- This surrogate modeling algorithm, called IDONE, with two variants (one with a basic and one with a more complex surrogate model).
- A proof that finding the optimum of the surrogate model gives an integer solution.
- Experimental results that show when IDONE outperforms random search, simulated annealing and Bayesian optimization.

Section 2 gives a general description of the problem and an overview of related work. Section 3 describes the IDONE algorithm and the proof. In Section 4, IDONE is compared to random search, simulated annealing and Bayesian optimization on two different problems: finding robust routes in a noise-perturbed traveling salesman benchmark problem, and a convex binary optimization problem. Finally, Section 5 contains conclusions and future work.

2 Problem description and related work

Consider the problem of minimizing an objective $f : \mathbb{R}^d \to \mathbb{R}$ with integer and bound constraints:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \mathbf{s.t.} \quad & \mathbf{x} \in \mathbb{Z}^d, \\ & l_i \leq x_i \leq u_i, \ i = 1, \ldots, d. \end{aligned} \tag{1}$$

These bounds are also assumed to be integer. It is assumed that f does not have a known mathematical formulation, and can only be accessed via noisy measurements $y = f(x) + \epsilon$, $y \in \mathbb{R}$, with $\epsilon \in \mathbb{R}$ zero-mean noise with finite variance. Furthermore, taking a measurement y is computationally expensive or is expensive due to a long measuring time, human decision making, deteriorating biological samples, or other reasons. Examples are hyper-parameter optimization in deep learning [10], contamination control of a food supply chain [11], and structure design in material science [12].

Although many standard optimization methods are unsuitable for this problem, there exists a vast number of methods that were designed with most of the above assumptions in mind. For example, local search heuristics [13] such as hill-climbing, simulated annealing,

or taboo search are general enough to be applied to this problem, and have the advantage of being easy to implement. These heuristics are considered as the baseline in this work, together with random search (simply assign the variables completely random values and keep the best results). Population-based heuristics such as genetic algorithms [14], particle swarm optimization [15], and ant colony optimization [16] operate in the same way as local search algorithms, but keep track of multiple candidate solutions that together form a population. These algorithms have been applied successfully in many applications, but are unsuitable for the problem described in this paper since evaluating a whole population of candidate solutions is not practical if each measurement is expensive. The same holds for algorithms that filter out the noise by averaging, such as COMPASS [17] or stochastic programming methods like sample average approximation [18], since they evaluate one candidate solution multiple times.

One of the most successful methods for solving decision problems involving integer decision variables is (Mixed) Integer Linear Programming [19, 20]. This field has brought forward strong and fast solvers that only need a problem description to compute solutions, even for nonlinear objective functions [21] as we consider in this work. However, if the problem cannot be completely formulated, because the objective function has no mathematical formulation, these solvers cannot be straightforwardly used. The core algorithm that makes these solvers so successful is the combined use of branch-and-bound search and (linear) relaxation: the value for a certain assignment to decision variables for a relaxed problem is compared to that of an earlier found solution. If the relaxed problem value is not better, that assignment can never lead to a better solution and the respective possibilities are pruned from the search. In our context, even though observations of the value are possible, these may be noisy. This seriously invalidates this approach: even if it would be possible to also observe values for relaxations of the problem, if these are noisy we may incorrectly prune parts of the search space. Reducing this effect by repeating the observation multiple times would be too expensive under the assumption that each evaluation of the objective is expensive.

Surrogate modeling techniques operate in a different way from the above methods: past measurements are used to fit a model, which is then used to select a next candidate solution. Bayesian optimization algorithms [1, 22], for example, have been successfully applied in many different fields. These methods use probability theory to determine the most promising candidate point according to the surrogate model. However, when the variables are discrete, the typical approach is to relax the integer constraints, which often leads to suboptimal solutions [7]. The authors in [7] tackled this problem by modifying the covariance function used in the surrogate model. Another approach, based on semi-definite programming, is given in [11]. The HyperOpt algorithm [23, 24] takes yet a different approach by using a Tree-structured Parzen Estimator as the surrogate model, which is discrete in case the variables are discrete. HyperOpt is considered the main contender in this paper.

A downside of many Bayesian optimization algorithms is that the computation time per iteration scales quadratically with the number of measurements taken up until that point. This causes these methods to become slower over time, and after a certain number of iterations they may even violate the assumption that the bottleneck in the problem is the cost of evaluating the objective. This downside is recognized and overcome in two similar algorithms: COMBO [12] and DONE [4]. Both algorithms use the power of random features [25] to get a fixed computation time every iteration, but COMBO is designed for combinatorial optimization while DONE is designed for continuous optimization. A disadvantage of COMBO is that it evaluates the surrogate model at every possible candidate point, a number that grows exponentially with the input dimension d. Though evaluating

the surrogate model takes very little time (compared to evaluating the original objective f), this still makes the algorithm unsuitable for problems where the input dimension d is large. A variant of DONE named CDONE has been applied to a mixed-integer problem, where the integer constraints were relaxed [10], but as mentioned earlier, this can lead to sub-optimal solutions.

However, the downside of having to relax the integer constraints can be circumvented. By choosing the basis functions in a certain way, we show how a model can be constructed for which it is known beforehand that the minima of the model exactly coincide with integer points. This makes it possible to apply the algorithm to combinatorial problems, as explained in the next section.

3 IDONE algorithm

The DONE algorithm [4] and its variants are able to solve problem (1) without the integer constraint by making use of a surrogate model. Every time a new measurement $y = f(\mathbf{x}) + \epsilon$ comes in, the surrogate model is updated, the minimum of the surrogate model is found, and an exploration step is performed. To make the algorithm suitable for combinatorial optimization we propose a variant of DONE called IDONE (integer-DONE), where the surrogate model is guaranteed to have integer-valued minima. This section starts with a theoretical contribution where we propose a piece-wise linear surrogate model. The second theorem in this section is our main contribution, which guarantees integer-valued minima of the proposed surrogate model. The section proceeds with an explanation of the model fitting step, as well as a visualization of the surrogate model. Next, the section shows how the minimum of the surrogate model can be found, and how this minimum is then used to choose a new point of evaluation for the objective. This section ends with a short overview of the whole algorithm.

3.1 Piece-wise linear surrogate model

The proposed surrogate model $g : \mathbb{R}^d \to \mathbb{R}$, with d the number of variables, is a linear combination of rectified linear units $\text{ReLU}(z) = \max(0, z)$, a basis function that is commonly used in the deep learning community as an activation function [26]:

$$\begin{aligned} g(\mathbf{x}) &= \textstyle\sum_{k=1}^{D} c_k \max\{0, z_k(\mathbf{x})\}, \\ z_k(\mathbf{x}) &= \mathbf{w}_k^T \mathbf{x} + b_k, \end{aligned} \tag{2}$$

with $\mathbf{x} \in \mathbb{R}^d$, $D \in \mathbb{N}$ the number of basis functions, $z_k : \mathbb{R}^d \to \mathbb{R}$ the *ReLU input function*, and $\mathbf{w}_k \in \mathbb{R}^d$, $b_k \in \mathbb{R}$, $c_k \in \mathbb{R}$ for $k = 1, \ldots, D$. Unlike what is common practice in the deep learning community, the parameters $\mathbf{w}_k$ and b_k remain fixed in this surrogate model. This makes the model linear in its parameters (c_k), allowing it to be trained via linear regression instead of iterative methods. This is explained in Section 3.2.

Because of the choice of basis functions, the surrogate model is actually piece-wise linear, which causes its local minima to be located in one of its corner points:

Theorem 1 *Any strict local minimum of g is located in a point $\mathbf{x}^*$ with $z_k(\mathbf{x}^*) = 0$ for d linearly independent z_k.*

Algorithm 1 Basic model parameters.

Input $d, l_i, u_i, i = 1, \ldots, d$
Output $\mathbf{w}_k, b_k, k = 1, \ldots, D$
  $k \leftarrow 1$
  $\mathbf{w}_k \leftarrow [0, \ldots, 0]^T$; $b_k \leftarrow 1$; $k \leftarrow k + 1$ ▷ model bias
  for $i = 1, \ldots, d$ **do**
    for $j = l_i, \ldots, u_i$ **do**
      $\mathbf{w} \leftarrow \mathbf{e}_i$ ▷ $\mathbf{e}_i$ = Unit vector in dimension i
      $b = -j$
      if $j = l_i$ **then** ▷ Lower bound
        $\mathbf{w}_k \leftarrow \mathbf{w}$; $b_k \leftarrow b$; $k \leftarrow k + 1$
      else if $j = u_i$ **then** ▷ Upper bound
        $\mathbf{w}_k \leftarrow -\mathbf{w}$; $b_k \leftarrow -b$; $k \leftarrow k + 1$
      else ▷ Between the bounds
        $\mathbf{w}_k \leftarrow \mathbf{w}$; $b_k \leftarrow b$; $k \leftarrow k + 1$
        $\mathbf{w}_k \leftarrow -\mathbf{w}$; $b_k \leftarrow -b$; $k \leftarrow k + 1$

The reverse of this theorem is not necessarily true: if $\hat{\mathbf{x}}$ satisfies $z_k(\hat{\mathbf{x}}) = 0$ for d linearly independent z_k, then it depends on the parameters c_k of the model whether $\hat{\mathbf{x}}$ is actually a local minimum or not.

The number of local minima and their locations depend on the parameters $\mathbf{w}_k$ and b_k. In this work, we provide two options for choosing these parameters in such a way that the local minima are always found in integer solutions. In the first case, the functions z_k are simply chosen to have zeros on hyper-planes that together form an integer lattice:

Definition 1 (Basic model) Let g be as in (2). The parameters $\mathbf{w}_k$, b_k of the basic model are chosen according to Algorithm 1. That is, every ReLU input function z_k is zero on a $(d-1)$-dimensional hyper-plane with $x_i = j$ for some dimension $i \in \{1, \ldots, d\}$ and some integer $l_i \leq j \leq u_i$. This leads to ReLU input functions of the form $z_k(\mathbf{x}) = \pm(x_i - j)$, $i = 1, \ldots, d$, $j = l_i, \ldots, u_i$, $k = 1, \ldots, D$. The total number of basis functions is $D = 1 + 2\sum_{i=1}^{d}(u_i - l_i)$. The D real-valued c_k's are the only parameters of this model.

The basic model has $D = 1 + 2\sum_{i=1}^{d} u_i - l_i$ basis functions in total. The 1 comes from the model bias, a basis function that is equal to 1 everywhere. This allows the model to be shifted up or down.

As an example, consider a problem with two variables in the range $[2, 3]$. Then $D = 1 + 2 \cdot ((3-2) + (3-2)) = 5$ and thus the basic model takes the form $g(\mathbf{x}) = \sum_{k=1}^{5} c_k \max\{0, z_k(\mathbf{x})\}$ with $\mathbf{w}_k$ and b_k as defined in Algorithm 1, so $z_1(\mathbf{x}) = 1$ (model bias), $z_2(\mathbf{x}) = x_1 - 2$, $z_3(\mathbf{x}) = -x_1 + 3$, $z_4(\mathbf{x}) = x_2 - 2$, and $z_5(\mathbf{x}) = -x_2 + 3$. The exact location of the minimum of this model depends on the values for the parameters c_k, but as an example, suppose that $c_2 = c_4 = 1$ and $c_1 = c_3 = c_5 = 0$. In that case, for $\mathbf{x} \in [2, 3] \times [2, 3]$ we have $g(\mathbf{x}) = \max\{0, x_1 - 2\} + \max\{0, x_2 - 2\}$, with a non-strict local minimum located in $\mathbf{x} = (2, 2)$.

Since all the basis functions depend only on one variable, this basic model might not be suitable for problems where the decision variables have complex interactions. Therefore, in the advanced model, we use the same basis functions, but we also add basis functions that depend on two variables:

Algorithm 2 Advanced model parameters.

Input $d, l_i, u_i, i = 1, \ldots, d$
Output $\mathbf{w}_k, b_k, k = 1, \ldots, D$
  Perform Algorithm 1.
  for $i = 2, \ldots, d$ **do**
    for $j = l_i - u_{i-1}, \ldots, u_i - l_{i-1}$ **do**
      $\mathbf{w} \leftarrow \mathbf{e}_i - \mathbf{e}_{i-1}$ ▷ $\mathbf{e}_i$ = Unit vector in dimension i
      $b = -j$
      if $j = l_i - u_{i-1}$ **then** ▷ Lower bound
        $\mathbf{w}_k \leftarrow \mathbf{w}$; $b_k \leftarrow b$; $k \leftarrow k + 1$
      else if $j = u_i - l_{i-1}$ **then** ▷ Upper bound
        $\mathbf{w}_k \leftarrow -\mathbf{w}$; $b_k \leftarrow -b$; $k \leftarrow k + 1$
      else ▷ Between the bounds
        $\mathbf{w}_k \leftarrow \mathbf{w}$; $b_k \leftarrow b, k \leftarrow k + 1$
        $\mathbf{w}_k \leftarrow -\mathbf{w}$; $b_k \leftarrow -b$; $k \leftarrow k + 1$

Definition 2 (Advanced model) Let g be as in (2). The parameters $\mathbf{w}_k$ and b_k of the advanced model are chosen according to Algorithm 2. That is, every ReLU input function z_k different from the ones in the basic model is zero on a $(d-1)$-dimensional hyper-plane with $x_i - x_{i-1} = j$ for some dimension $i \in \{2, \ldots, d\}$ and some integer $l_i - u_{i-1} \leq j \leq u_i - l_{i-1}$. This leads to ReLU input functions of the form $z_k(\mathbf{x}) = \pm(x_i - x_{i-1} - j)$, $i = 2, \ldots, d$, $j = l_i - u_{i-1}, \ldots, u_i - l_{i-1}$. This model has $2\sum_{i=2}^{d} u_i - l_i + u_{i-1} - l_{i-1}$ more basis functions than the basic model.

The ReLU input functions of this type are zero on diagonal hyper-planes through two variables, see Fig. 1.

To continue the example of the basic model above, we can turn it into an advanced model, which consequently has $D = 9$ input functions. Algorithm 2 defines additional input functions $z_6(\mathbf{x}) = x_2 - x_1 + 1$, $z_7(\mathbf{x}) = x_2 - x_1$, $z_8(\mathbf{x}) = -x_2 + x_1$, and $z_9(\mathbf{x}) = -x_2 + x_1 + 1$. Again, the location of the minimum of this advanced model depends on the values for the parameters c_k. For example, if $c_5 = 1$, $c_7 = 1$, and $c_k = 0$ for $k \notin \{5, 7\}$, then we have $g(\mathbf{x}) = \max\{0, -x_2 + 3\} + \max\{0, x_2 - x_1\}$, with a non-strict local minimum located in $\mathbf{x} = (3, 3)$.

We now show our main theoretical result.

Theorem 2 *(I) If $\mathbf{x}^*$ is a strict local minimum of the basic model, then $\mathbf{x}^* \in \mathbb{Z}^d$ and $l_i \leq x_i \leq u_i$, $\forall i = 1, \ldots, d$.*

(II) If $\mathbf{x}^*$ is a strict local minimum of the advanced model, then $\mathbf{x}^* \in \mathbb{Z}^d$.

(III) If $\mathbf{x}^*$ is a non-strict local minimum of the basic model, it holds that the model retains the same value when going from $\mathbf{x}^*$ to the nearest point $\hat{\mathbf{x}}$ that satisfies $\hat{\mathbf{x}} \in \mathbb{Z}^d$ and $l_i \leq \hat{x}_i \leq u_k$, $\forall i = 1, \ldots, d$.

(IV) If $\mathbf{x}^*$ is a non-strict local minimum of the advanced model, it holds that the model retains the same value when rounding $\mathbf{x}^*$ to the nearest integer.

Proof (I) Let $\mathbf{x}^*$ be a strict local minimum of the basic model. By Theorem 1, there are d linearly independent z_k with $z_k(\mathbf{x}^*) = 0$. From Algorithm 1 it can be seen that all functions z_k have the form $z_k(\mathbf{x}) = \pm(x_i - j)$, for some $i = 1, \ldots, d$, $j = l_i, \ldots, u_i$. Since d of

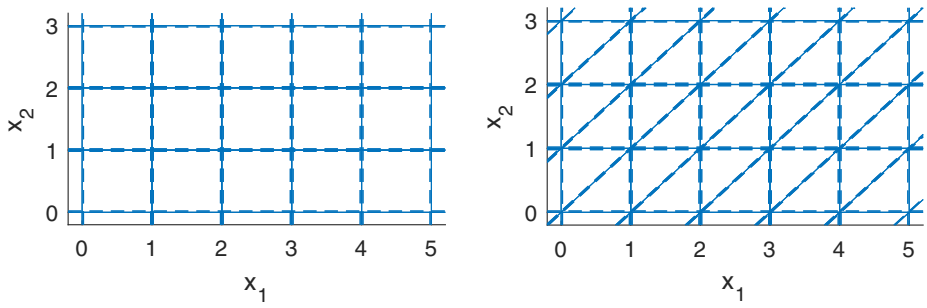


Fig. 1 Regions where the ReLU input functions z_k of the surrogate model are exactly zero, for the basic model (left) and the advanced model (right), for a problem with two variables with lower bounds (0, 0) and upper bounds (5, 3). The functions z_k have been chosen in such a way that they cross exactly at integer points within the bounded region. This ensures that the model has its minimum in one of these points, making the model more suitable for combinatorial optimization problems

these functions are linearly independent, all d of them must have a different i. Since all d of them satisfy $z_k(\mathbf{x}^*) = 0$, it holds that $x_i^* = j$, for some $j = l_i, \ldots, u_i$, $\forall i = 1, \ldots, d$, which is what is claimed.

(II) Let $\mathbf{x}^*$ be a strict local minimum of the advanced model. By Theorem 1, there are d linearly independent z_k with $z_k(\mathbf{x}^*) = 0$. This means that all z_k together must depend on all x_i, $i = 1, \ldots, d$. From Algorithm 2 it can be seen that all functions z_k have the same form as in the basic model, that is, $z_k(\mathbf{x}) = \pm(x_i - j)$, $i = 1, \ldots, d$, $j = l_i, \ldots, u_i$, or they have the form $z_k(\mathbf{x}) = \pm(x_i - x_{i-1} - j)$, $i = 2, \ldots, d$, $j = l_i - u_{i-1}, \ldots, u_i - l_{i-1}$. No matter the form, thus

$$x_i^* - x_{i-1}^* \in \mathbb{Z}\ \forall i = 2, \ldots, d. \tag{3}$$

To arrive at a contradiction, suppose that $\exists s \in \{1, \ldots, d\}$ such that $x_s^* \notin \mathbb{Z}$. Then by (3), $x_i^* \notin \mathbb{Z}$ for all $i \in \{1, \ldots, d\}$. However, this is only possible if none of the z_k have the form $z_k(\mathbf{x}) = \pm(x_i - j)$, and all d of the z_k have the form $z_k(\mathbf{x}) = \pm(x_i - x_{i-1} - j)$, for d different i. But by construction, there are only $d - 1$ of these last ones available, see Algorithm 2 (the for-loop starts at 2). Therefore, it is not true that $\exists s \in \{1, \ldots, d\}$ such that $x_s^* \notin \mathbb{Z}$. Hence, $x_i^* \in \mathbb{Z}\ \forall i = 1, \ldots, d$, which is what the theorem claims.

(III) Let $\mathbf{x}^*$ be a non-strict local minimum of the basic model and suppose $x_s^* \notin \mathbb{Z} \cup [l_s, u_s]$ for some $s \in \{1, \ldots, d\}$. Let L be the line segment from x_s^* to the nearest point $\hat{x}_s$ in the set $\mathbb{Z} \cup [l_s, u_s]$, without including that point. Since the only z_k functions that depend on x_s have the form $z_k(\mathbf{x}) = \pm(x_s - j)$, $j = l_s, \ldots, u_s$, it follows that $z_k(\mathbf{x}) = 0$ does not happen on L for any z_k that depends on x_s. Therefore, model g is linear on this line segment, and since $\mathbf{x}^*$ is a non-strict local minimum and g is continuous, g retains the same value when replacing x_s^* by $\hat{x}_s$. This can be repeated for all s for which $x_s^* \notin \mathbb{Z} \cup [l_s, u_s]$, which proves the claim.

(IV) Let $\mathbf{x}^*$ be a non-strict local minimum of the advanced model and suppose $\mathbf{x}^* \notin \mathbb{Z}$. We first show that rounding $\mathbf{x}^*$ to the nearest integer does not change the sign of any z_k. Note that all the z_k of the advanced model have the form $z_k(\mathbf{x}) = \pm(x_i - j)$ or $z_k(\mathbf{x}) =$

$\pm(x_i - x_{i-1} - j)$, for some $i = 1, \ldots, d$ and some integer j. Let $\bar{x}_i$ denote rounding x_i to the nearest integer. Then we have (because j is integer):

$$\begin{aligned} x_i \leq j \quad &\Rightarrow \bar{x}_i \leq j, \text{ and } x_i \geq j \Rightarrow \bar{x}_i \geq j, \\ x_i - x_{i-1} \leq j \ &\Rightarrow \bar{x}_i \leq \overline{x_{i-1} + j} \Rightarrow \bar{x}_i - \bar{x}_{i-1} \leq j, \\ x_i - x_{i-1} \geq j \quad &\Rightarrow \bar{x}_i - \bar{x}_{i-1} \geq j. \end{aligned}$$

Since the sign of none of the z_k change when rounding, and model g is only nonlinear when going from $z_k(\mathbf{x}) < 0$ to $z_k(\mathbf{x}) > 0$ for some $k = 1, \ldots, D$, it follows that g is linear on the line segment from $\mathbf{x}^*$ to the nearest integer. Together with the fact that $\mathbf{x}^*$ is a non-strict local minimum, it follows that g retains the same value on this line segment. Finally, the claim is valid because g is continuous. □ □

3.2 Fitting the model

Because the surrogate model g is linear in its parameters c_k, fitting the model can be done with linear regression. Given input-output pairs $(\mathbf{x}_i, y_i, i = 1, \ldots, N)$, this is done by solving the regularized linear least squares problem

$$\min_{\mathbf{c}_N} \sum_{n=1}^{N} (y_n - g(\mathbf{x}_n, \mathbf{c}_N))^2 + \lambda ||\mathbf{c}_N - \mathbf{c}_0||_2^2, \tag{4}$$

with regularization parameter λ and initial weights $\mathbf{c}_0$. The regularization part is added to overcome ill-conditioning, noise, and model over-fitting. Furthermore, by choosing $\mathbf{c}_0 = [0, 1, \ldots, 1]^T$, it is ensured that the surrogate model is convex before the first iteration [10]. In this work, $\lambda = 0.001$ has been chosen.

To prevent having to solve this problem at every iteration (with runtime $O(N^3)$), (4) is solved with the recursive least squares algorithm [27]. This algorithm has runtime $O(D^2)$ per iteration, with D the number of basis functions used by the model. This implies that the computation time per iteration does not depend on the number of measurements, which is a big advantage compared to Bayesian optimization algorithms (which usually have complexity $O(N^2)$ per iteration). The memory is also $O(D^2)$, because a $D \times D$ covariance matrix needs to be stored. Since D scales linearly with the input dimension d and with the lower and upper bounds, the computational complexity of fitting the surrogate model is $O(p^2 d^2)$, with $p = \max_i(u_i - l_i)$.

3.2.1 Model visualization

To visualize the surrogate model used by the IDONE algorithm, the fitting procedure is applied to a simple traveling salesman problem with four cities. The distance matrix for the cities is shown in Table 1. The decision variables are chosen as follows: the route starts at city 1, then variable $x_1 \in \{1, 2, 3\}$ determines which of the three remaining cities is visited, then variable $x_2 \in \{1, 2\}$ determines which of the two remaining cities is visited; then the one remaining city is visited, then city 1 is visited again. This problem has two optimal solutions: $\mathbf{x} = [1, 2]^T$ (route 1-2-4-3-1) and $\mathbf{x} = [2, 2]^T$ (route 1-3-4-2-1), both with a total distance of 80. All other solutions have a total distance of 95.

Figure 2 shows what the surrogate model looks like after taking measurements in all possible data points for this problem, which is possible due to the low number of possibilities. It can be observed that this model is piece-wise linear and that any local minimum retains

Table 1 Distance matrix for the simple traveling salesman problem

	1	2	3	4
1		10	15	20
2	10		35	25
3	15	35		30
4	20	25	30	

the same value when rounding to the nearest integer. Furthermore, the diagonal lines (see also Fig. 1) make the advanced model more accurate.

3.3 Finding the minimum of the model

After fitting the model g at iteration N, the algorithm proceeds to find a local minimum using the new weights $\mathbf{c}_N$:

$$\begin{aligned} \mathbf{x}^* = \quad & \arg\min_{\mathbf{x}} g(\mathbf{x}, \mathbf{c}_N), \\ \mathbf{s.t.} \quad & \mathbf{x} \in \mathbb{Z}^d, \\ & l_i \leq x_i \leq u_k, \ i = 1, \ldots, d. \end{aligned} \tag{5}$$

The BFGS method [28] with a relaxation on the integer constraint was used to solve the above problem, with a provided analytical derivative of g. In this work, the derivative of the basis function $\mathrm{ReLU}(z) = \max(0, z)$ has been chosen to be 0.5 at $z = 0$. The optimal solution was rounded to the nearest integer per Theorem 2.

3.4 Exploration

After fitting the model and finding its minimum, a new point $\mathbf{x}_{N+1}$ needs to be chosen to evaluate the function f. As in DONE [4], a random perturbation δ is added to the found minimum: $\mathbf{x}_{N+1} = \mathbf{x}^* + \delta$, but instead of a continuous random variable, $\delta \in \{-1, 0, 1\}^d$ is

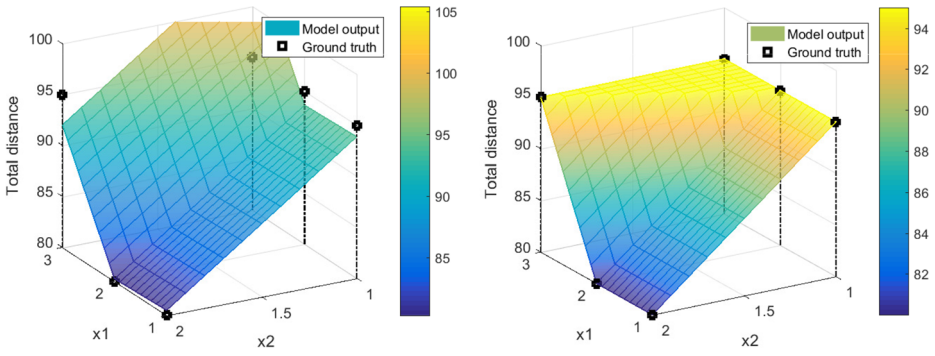


Fig. 2 Model output for the simple traveling salesman problem for the basic model (left) and the advanced model (right) . The starting city is city 1, x_1 determines which remaining city is visited next, x_2 determines which remaining city is visited third, then the only remaining city is visited, and then city 1 is visited again

a discrete random variable with the following probabilities:

$$
\begin{aligned}
P(\delta_i = 0) &= 1 - p, \\
P(\delta_i = 1) &= \begin{cases} p, & x_i^* = l_i, \\ 0, & x_i^* = u_i, \\ p/2, & \text{otherwise}, \end{cases} \\
P(\delta_i = -1) &= \begin{cases} 0, & x_i^* = l_i, \\ p, & x_i^* = u_i, \\ p/2, & \text{otherwise}. \end{cases}
\end{aligned} \tag{6}
$$

In this work, $p = 1/d$ has been chosen (d is the number of variables).

3.5 IDONE algorithm

The IDONE algorithm iterates over three phases: updating the surrogate model with recursive least squares, finding the minimum of the model, and performing the exploration step. The pseudocode for the algorithm is shown in Algorithm 3. Depending on which subroutine is used in the first line, we refer to this algorithm as either IDONE-basic (using the basic model) or IDONE-advanced (using the advanced model).

Algorithm 3 IDONE-advanced, IDONE-basic.

Input $\mathbf{x}_1 \in \mathbb{R}^d$, $\lambda \in \mathbb{R}$, $(l_i, u_i)\ \forall i = 1, \ldots, d$, $N \in \mathbb{N}$, $p \in [0, 1]$
Output $\mathbf{x}_N$, y_N
  Get $\mathbf{w}_k$, b_k, $k = 1, \ldots D$ from Algorithm 1 for IDONE-basic or from Algorithm 2 for IDONE-advanced
  $\mathbf{c}_0 \leftarrow [0, 1, \ldots, 1]^T \in \mathbb{R}^D$
  for $n = 1, \ldots, N$ **do**
    Evaluate $y_n = f(\mathbf{x}_n) + \epsilon$
    Calculate $\mathbf{c}_n$ from $\mathbf{c}_{n-1}$ with recursive least squares
    Compute $\mathbf{x}^*$ using (5)
    if $n < N$ **then**
      $\mathbf{x}_{N+1} \leftarrow \mathbf{x}^* + \delta$, with δ as in Section 3.4

4 Experimental results

The main idea put forward is to use a model that guarantees integer-valued minima. This idea is evaluated with two different models: a basic and an advanced model. We evaluate both models on two different benchmark problems: finding a robust route for a noise-perturbed asymmetric traveling salesman benchmark problem with 17 cities, and an artificial convex binary optimization problem with up to 150 integer variables. The first problem gives a first indication of the algorithm's performance on an objective function that follows from a simulation where there is a network structure: traveling between interconnected cities with uncertain travel times between them. The second problem shows an easier and more tangible situation - due to the convexity and the fact that we know the global optimum - which makes it easier to interpret results. It is also used to investigate the scalability of the proposed methods.

The algorithm is compared with two basic search strategies: random search (RS) and simulated annealing (SA), and two advanced algorithms representing the state of the art: Bayesian optimization [1, 22] (BO), using one of the existing Python implementations [1], and the Python library *HyperOpt* [23, 24] (HypOpt). HypOpt makes use of a Parzen estimator, which gives a probability distribution over the different discrete choices in the search space, based on how often they have been visited and whether the visited point was better or worse than the best solution so far. The other algorithms are also implemented in Python [2], and for random search we used HypOpt's implementation. All experiments were done on a cluster (32 Intel Xeon E5-2650 2.0 GHz CPUs), without making use of parallelization of the algorithms themselves. For BO and HypOpt, we used the default settings. It should be noted that BO and HypOpt are both aimed at minimizing black-box functions using as few function evaluations as possible. For SA, the settings are explained below.

In the context of the IDONE algorithm, the SA algorithm essentially consists of just the exploration step of the IDONE algorithm (see Section 3.4), coupled with a probability of returning to the previous candidate solution. Suppose the current best solution is $(\mathbf{x}_b, y_b)$, and that the exploration step as defined in Section 3.4 gives a new candidate solution $(\mathbf{x}_c, y_c)$. If $y_c < y_b$, then $\mathbf{x}_c$ is accepted as the new best solution. Else, there is a probability that $\mathbf{x}_c$ is still accepted as the new best solution. This probability is equal to $e^{(y_b - y_c)/T}$, with T a so-called temperature. In this work, the simulated annealing algorithm starts out with a starting temperature $T = T_0$, and the temperature is multiplied with a factor T_f every iteration. This strategy is called a cooling schedule. For the asymmetric traveling salesman problem, $T_0 = 4.48$ and $T_f = 0.996$ have been chosen. For the convex binary optimization problem, $T_0 = 1$ and $T_f = 0.95$ have been chosen.

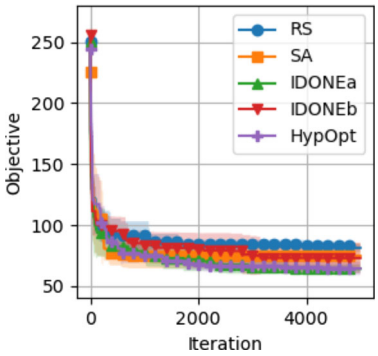
4.1 Robust routes for an asymmetric traveling salesman problem (17 cities)

Consider the asymmetric TSP benchmark called BR17. This benchmark was taken from the TSPLIB website [29], a library of sample instances for the traveling salesman problem. While there exist specific solvers developed for this problem, these solvers are not adequate if the objective to be minimized is perturbed by noise. Here, noise $\epsilon \in [0, 1]$, with a uniform distribution, was added to the distances between any two cities (for distances other than 0 or infinity, which both occurred once per city, the mean distance between cities is 16.43 for this instance). Furthermore, every time a sequence of cities has been chosen, we evaluate this route 100 times, with different noise samples. The objective is the worst-case (largest) route length of the chosen sequence of cities. Minimizing this objective should then result in a route that is robust to the noise in the problem. For the variables the same encoding as in Section 3.2.1 has been used, giving 15 integer variables in total.

All algorithms were run 5 times on this problem, and the results are shown in Fig. 3. The BO algorithm was not included as it took over 80 hours per run. It can be seen that both HyperOpt and IDONE-advanced outperform the simpler benchmark methods. IDONE-advanced achieves similar results as HyperOpt while being twice as fast, and both are several orders of magnitude faster than Bayesian optimization. It seems IDONE-basic is unable to deal with the complex interaction between the variables due to the basic structure of the model, as it performs similarly to the simpler SA algorithm. All methods managed to outperform random search.

[1] https://github.com/fmfn/BayesianOptimization

[2] We have made the IDONE algorithm available on https://bitbucket.org/lbliek2/idone.



	Computation time [s]
RS	80.776 ± 0.271
SA	47.684 ± 0.087
IDONEa	316.147 ± 7.902
IDONEb	140.689 ± 0.962
HypOpt	668.497 ± 19.198

Fig. 3 Best found worst-case total distance (*left*) and corresponding computation time (*right*) of the noisy TSP with 17 cities for IDONE-advanced (IDONEa), IDONE-basic (IDONEb), random search (RS), simulated annealing (SA), and HyperOpt (HypOpt), averaged over 5 runs. The shaded area (*left*) visualizes the range across all 5 runs

4.2 Convex binary optimization

To gain a better understanding of the different algorithms and their scalability, the second experiment is done on a function with a known mathematical formulation. Consider the problem of minimizing the function

$$f(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^T A(\mathbf{x} - \mathbf{x}^*), \tag{7}$$

with A a random positive semi-definite matrix, and $\mathbf{x}^* \in \{0, 1\}^d$ a randomly chosen vector, with d the number of binary variables. The optimal solution $\mathbf{x}^*$ or the structure of the function is not given to the different algorithms, only the number of variables and the fact that they are binary. Starting from a matrix U where each element is randomly generated from a uniform [0, 1] distribution, matrix A is constructed as

$$A = (U + U^T)/d + I_{d\times d}, \tag{8}$$

with I the identity matrix. The function f can only be accessed via noisy measurements $y = f(\mathbf{x}) + \epsilon$, with $\epsilon \in [0, 1]$ a uniform random variable. We ran 100 experiments with this function, with IDONE and the other black-box optimization algorithms. For each run, A and $\mathbf{x}^*$ were randomly generated, as well as the initial guess $\mathbf{x}_0$. All algorithms were stopped after taking 1000 function evaluations, and the best found objective value was stored at each iteration. Figure 4 shows a convergence plot for the case $d = 100$. It can be seen that the two variants of IDONE have the fastest convergence. The large number of variables is too much for a pure random search, but also for HyperOpt, even though the latter is designed for problems with hundreds of variables [24]. Simulated annealing still gives decent results on this problem.

Figure 5 shows the final objective value and computation time after 1000 iterations for the same problem for different values of d. The number of variables d was varied between 5 and 150. Bayesian optimization was only evaluated once for $d = 5$ due to its large computation time. As can be seen, IDONE is the only algorithm that consistently gives a solution at or close to the optimal solution (which has an objective value between 0 and 1) for the highest dimensions. Where all algorithms get at or close to the optimal solution for problems with 10 variables or less, the difference between the algorithms becomes more distinguishable when more variables are considered. The strengths of HyperOpt, such as

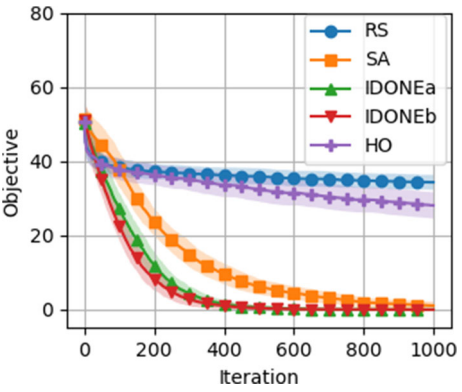


Fig. 4 Lowest objective value found at each iteration of the binary convex optimization example with 100 binary variables, averaged over 100 runs. The shaded area indicates the standard deviation. For every run, the initial value, matrix A, and vector $\mathbf{x}^*$ were chosen randomly

dealing with different types of variables that can have complex interactions, are not relevant for this particular problem, and the Parzen estimator surrogate model does not seem to scale well to higher dimensions compared to the piece-wise linear model used by IDONE. Both variations of IDONE also scale better than the other state-of-the-art algorithms in terms of computational time, with IDONE-basic being up to 20 times faster than HyperOpt, which is already several orders of magnitude faster than Bayesian optimization.

5 Conclusions and future work

The IDONE algorithm is a black-box optimization algorithm that is designed for combinatorial problems with binary or integer constraints, and has shown to be useful in particular when the objective can only be accessed via costly and noisy evaluations. By using a surrogate model that is designed in such a way that its optimum coincides with an integer solution, the algorithm can be applied to combinatorial optimization problems without having to resort to rounding in the objective function. IDONE has a fixed computation time per iteration that scales with the number of variables but is not influenced by the number of

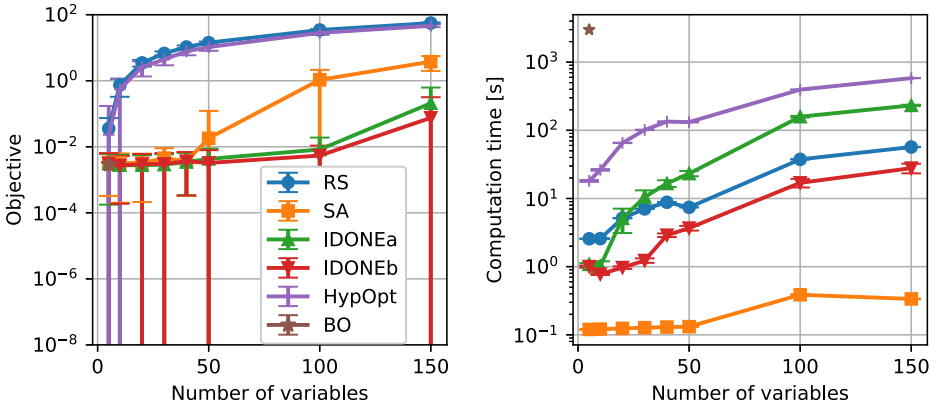


Fig. 5 Objective value (*left*) and computation time (*right*) of the convex binary optimization problem for the different algorithms after 1000 iterations, averaged over 100 runs, for problems with different numbers of variables d. Bayesian optimization (BO) was evaluated only for 1 run

times the function has been evaluated, which is an advantage compared to Bayesian optimization algorithms. One variant of the proposed algorithm, IDONE-advanced, has been shown to outperform random search and simulated annealing on the problem of finding robust routes in a noise-perturbed traveling salesman benchmark problem, and on a convex binary optimization problem with up to 150 variables. The other variant of the algorithm, IDONE-basic, mainly performed well in the second experiment, where it outperformed the state-of-the-art. HyperOpt, a popular surrogate modeling algorithm for problems with hundreds of variables, performs similar as IDONE-advanced on the traveling salesman benchmark problem, but does not scale as well on the binary optimization problem. On both problems, IDONE is faster than HyperOpt, which is already multiple orders of magnitude faster than regular Bayesian optimization algorithms. We conclude that the main idea to use surrogate models with integer-valued minima is successful, but that for smaller problems with many interactions between the variables, there seem to be some limitations, especially for simpler surrogate models. Understanding these limitations better is a challenging direction for future work.

The results show that there is room for improvement in the use of surrogate models for black-box combinatorial optimization, and that using continuous models with integer-valued local minima is a new and promising way forward. In future work, the special structure of the surrogate model will be further exploited to provide a faster implementation, and the algorithm will be tested on real-life applications of combinatorial optimization with expensive objective functions. The question also arises whether this algorithm would perform well in situations where the objective function is not expensive to evaluate, or does not contain noise. Population-based methods perform particularly well on cheap black-box objective functions, so it would be interesting to see if they could be combined with the surrogate model used in this paper. As for the noiseless case, it is known that for continuous variables it becomes easy in this case to estimate the gradient and use more traditional gradient-based methods, but in the case of discrete variables the traditional combinatorial optimization methods might still benefit from IDONE's piece-wise linear surrogate model. Where surrogate-based optimization techniques have had great success in continuous optimization problems from many different fields, we hope that this work opens up the route to success of these techniques for the plenty of open combinatorial problems in these fields.

Acknowledgments This work is part of the research programme Real-time data-driven maintenance logistics with project number 628.009.012, which is financed by the Dutch Research Council (NWO). The authors would also like to thank Arthur Guijt for helping with the python code.

References

1. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**(4), 455–492 (1998)

2. Gosavi, A.: Simulation-based optimization: parametric optimization techniques and reinforcement learning, Springer, 55 (2015)
3. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to derivative-free optimization, Siam, 8 (2009)
4. Bliek, L., Verstraete, H.R.G.W., Verhaegen, M., Wahls, S.: Online optimization with costly and noisy measurements using random Fourier expansions. IEEE Transactions on Neural Networks and Learning Systems **29**(1), 167–182 (2018)
5. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp. 2951–2959 (2012)
6. Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J., Doucet, A.: A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. Auton. Robot. **27**(2), 93–103 (2009)
7. Garrido-Merchán, E.C., Hernández-Lobato, D.: Dealing with integer-valued variables in Bayesian optimization with Gaussian processes. arXiv:1706.03673 (June 2017)
8. Verwer, S., Zhang, Y., Ye, Q.C.: Auction optimization using regression trees and linear models as integer programs. Artif. Intell. **244**, 368–395 (2017)
9. Verbeeck, D., Maes, F., De Grave, K., Blockeel, H.: Multi-objective optimization with surrogate trees. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 679–686, ACM (2013)
10. Bliek, L., Verhaegen, M., Wahls, S.: Online function minimization with convex random ReLU expansions. In: Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on, pp. 1–6, IEEE (2017)
11. Baptista, R., Poloczek, M.: Bayesian optimization of combinatorial structures. In: International Conference on Machine Learning, pp. 471–480 (2018)
12. Ueno, T., Rhone, T.D., Hou, Z., Mizoguchi, T., Tsuda, K.: Combo: An efficient Bayesian optimization library for materials science. Materials discovery **4**, 18–21 (2016)
13. Aarts, E.H.L., Lenstra, J.K.: Local search in combinatorial optimization, Princeton University Press (2003)
14. Rajeev, S., Krishnamoorthy, C.S.: Discrete optimization of structures using genetic algorithms. Journal of structural engineering **118**(5), 1233–1250 (1992)
15. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics: Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108, IEEE (1997)
16. Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial life **5**(2), 137–172 (1999)
17. Hong, L.J., Nelson, B.L.: Discrete optimization via simulation using COMPASS. Oper. Res. **54**(1), 115–129 (2006)
18. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on stochastic programming: modeling and theory, SIAM (2014)
19. Wolsey, L.A.: Integer programming, John Wiley & Sons, vol. 52 (1998)
20. Schrijver, A.: Theory of linear and integer programming, John Wiley & Sons (1998)
21. Li, D., Sun, X.: Nonlinear integer programming, Springer Science & Business Media, 84 (2006)
22. Mockus, J.: Bayesian approach to global optimization: theory and applications, Springer Science & Business Media, 37 (2012)
23. Bergstra, J., Yamins, D., Cox, D.D.: Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In: Proceedings of the 12th Python in science conference, pp. 13–20 (2013)
24. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on Machine Learning. Jmlr (2013)
25. Rahimi, A., Recht, B.: Uniform approximation of functions with random bases. In: 46th Annual Allerton Conference on Communication, Control, and Computing, pp. 555–561, IEEE (2008)
26. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
27. Sayed, A.H., Kailath, T.: Recursive least-squares adaptive filters, The Digital Signal Processing Handbook, 21, 1 (1998)
28. Wright, S., Nocedal, J.: Numerical optimization. Springer Science **35**, 67–68 (1999)
29. TSPlib: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html (2019)