Chapter 4
Constrained Optimization

4.1 Constrained Bayesian Optimization

Constrained optimization is just the extension of unconstrained optimization to the case of having constraints. A constraint is a condition, equality, or inequality that must be satisfied in order for the solution to be valid. We can write the constrained optimization problem as

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} \, f(x) \;\text{ subject to }\; c(x) \leq 0, \tag{4.1}$$

where $f : \mathcal{X} \to \mathbb{R}$ denotes a scalar-valued objective function for $\mathcal{X} \subset \mathbb{R}^d$ and $c : \mathcal{X} \to \mathbb{R}^m$ denotes a vector of m constraint functions. These constraints could be in the form of inequalities, $c_j(x) \leq 0$, equalities, $c_j(x) = 0$, or binary constraints represented with an indicator function, $c_j(x) = \mathbb{1}_{\{x \in \mathcal{X}_{c_j}\}}(x) = 0$.

Constrained optimization problems are typically more difficult than unconstrained problems because the constraints often operate at odds with the objective function, so trying to meet the constraint will drive the solution away from the global optimum. In simple cases, the optimum is not along a constraint boundary, in which case the problem is essentially an unconstrained problem. But in most cases, the optimum is on a boundary, with the unconstrained solution lying in a region that does not meet all the constraints. We refer to the subset of $\mathcal{X}$ that satisfies the constraints as the *valid* region. So one needs to be able to search for the best solution in the valid region, with the constraints taking first priority, and the objective function being second priority. More complex problems will have non-convex constraint boundaries, or even disconnected valid regions, both of which make searching quite difficult.

T. Pourmohamad, H. K. H. Lee, *Bayesian Optimization with Application to Computer Experiments*, SpringerBriefs in Statistics,
https://doi.org/10.1007/978-3-030-82458-7_4

Thus Bayesian optimization in the constrained case needs to learn both the objective function and the valid region. For both the objective and the constraints, exploitation needs to be balanced with exploration, although the interplay between the objective and the constraints needs to be taken into account. Exploration of the objective only needs to occur in the valid region, and exploration of the valid region only needs to occur for more optimal values of the objective. An efficient algorithm will be able to quickly hone in on the most promising parts of the space.

Again, constrained optimization is usually difficult because at least one of the constraints operates in opposition to the objective function, that is, they are negatively correlated. When the outputs are known or suspected to be correlated, it is common practice to use latent processes to induce a correlation structure between them (Sammel et al. 1997; Moustaki and Knott 2000, for example), however, this modeling choice comes at the cost of increased model complexity. Likewise, there is an increase in the time and computational burden of joint modeling as compared to independent modeling of the objective and constraint functions, however, as seen in Pourmohamad and Lee (2016), there can be significant gains in predictive accuracy and statistical coverage by the use of joint modeling when the objective and constraint functions are indeed correlated. Moreover, when the objective and constraint function are correlated, the shared information in modeling the functions jointly can lead to better model fits and prediction, which should lead to far fewer function evaluations of the expensive computer model in converging to the global solution to the optimization problem.

So why not simply use a joint model, such as a joint Gaussian process model (Wackernagel 2003), to model the objective and constraint functions jointly? After all, a joint Gaussian process model for the objective and constraint function will theoretically perform at least as well as using independent Gaussian processes models for each. The answer is two-fold. First off, if the objective and constraint function are not strongly correlated, then the added overhead and complexity in fitting the joint Gaussian process does not justify its use when, given the expensive nature of the computer model, computational speed of the BO algorithm is of utmost importance. Little is to be gained when there is not strong information to share across the models. Secondly, and quite frankly, independent Gaussian process surrogate models typically just work. Even without exploiting the joint information that exists for correlated outputs, independent Gaussian process surrogate models often do a fantastic job of predicting the objective and constraint functions well, and in a fraction of the time as compared to joint models. In the context of BO, it usually makes practical sense to use independent surrogate models for the objective and constraint functions.

With all of this in mind, generalizing the unconstrained BO algorithm (Algorithm 1 of Sect. 3.1) to the constrained case requires only a few additional pieces.

Here we describe the general formulation for constrained BO via the following algorithm:

Algorithm 2: The general constrained BO algorithm

Initialization:
Start with an initial data set D_0
for $n = 1, \ldots, N$ **do**
Fit surrogate models for the objective and constraint functions;
Select $x_n = \text{argmax}_{x \in \mathcal{X}} a_{n-1}(x)$;
Evaluate f and c at x_n to obtain y_n;
Augment data $D_n = D_{n-1} \cup \{(x_n, y_n)\}$;
end
Return:
$x^* = \text{argmin}_{x \in \mathcal{X}} f(x)$

Note that, in general, the only difference between the BO algorithm in the constrained case versus the unconstrained case is that a (either joint or independent) model must be specified for the constraint function, $c(x)$, as well as the objective function, $f(x)$. Otherwise, all other steps proceed similarly to the unconstrained case, i.e., start with an initial sample chosen from a space-filling design, and fit appropriate surrogate models that can be used to maximize an acquisition function for choosing the best next input at which to evaluate the computer model. This iterative algorithm repeats, updating the observed data after every iteration, until all computational budget has been exhausted. Different acquisition functions are needed in the presence of constraints, and that is the topic of the next section.

4.2 Choice of Acquisition Function

Just like the case of unconstrained Bayesian optimization, the acquisition function is the key to constrained Bayesian optimization. We introduce three different ways to think about the choice of an acquisition function here, although there are additional possibilities. These three are: (1) joint exploration of the objective function and the constraint functions, (2) focusing on staying inside the valid region, and (3) combining statistical modeling with numerical methods for effective exploration and exploitation. These approaches can overlap, and some examples of methods we present here are examples of more than one of these approaches.

One approach to constrained optimization is to choose an acquisition function that drives simultaneous learning of both the objective function and the constraint functions. By creating a good surrogate model for both f and c, one can find the global minimum using the surrogate model. In practice, the surrogate model for the objective function only needs to be accurate in the valid region, and the surrogate model for the constraints only needs to be accurate in regions of relatively

lower objective function values. Two examples discussed in more detail in this chapter are Constrained Expected Improvement (Schonlau et al. 1998; Gardner et al. 2014) (Sect. 4.2.1) and Augmented Lagrangian methods (Gramacy et al. 2016) (Sect. 4.2.3). Additional examples include integrated expected conditional improvement (Gramacy and Lee 2011), expected volume minimization (Picheny 2014), and constrained BO for noisy experiments (Letham et al. 2019). As these approaches are learning both f and c, they tend to sample a relatively larger number of points near the constraint boundary but outside of the valid region, where f may be more desirable and c is close to being satisfied. These observations can help the search, although they will not ultimately be the optimum point because the constraints are not all satisfied.

A second approach is to focus on staying inside the valid region. This approach is motivated by interior point methods from numerical optimization, where the acquisition function is chosen so that if you have a starting point inside the valid region, the search attempts to remain inside the valid region, driving toward the boundary without crossing it. A prime example would be Barrier Methods (Pourmohamad and Lee 2021) (Sect. 4.2.4). Another approach to focusing on the valid region is Asymmetric Entropy (Lindberg and Lee 2015) (Sect. 4.2.2). The formulation of the constrained optimization problem in (4.1) basically assumes that the objective function f and the constraint function c can be evaluated for all $x \in \mathcal{X}$. In some cases, the computer model might not return any value when x is not in the valid region. For example, some inputs x might lead to trying to take the logarithm of a negative number somewhere in the code, or a matrix may become numerically singular in double precision and thus become not invertible. The problems may occur deep in the calculations, and a significant amount of computing may be necessary before discovering the issue. This situation is a case where the optimum must be found within the domain for which the computer model is able to return values. Outside of this valid region, the code fails to run and does not return a value. Thus c is observable as a binary variable, and f is only observable when $c = 0$. In the standard case of (4.1), every new observation provides some information toward optimization. However, in the case of code that does not run outside the valid region, information is only gained for runs inside the valid region. When the computer model fails to complete a run, the time spent on that attempted run is wasted. Thus it is much more important in this case to keep as many runs as possible inside the valid region. This class of methods aims to find the constrained optimum while limiting the number of observations with $c > 0$.

A third approach is based on hybrid optimization algorithms. These combine direct numerical optimization methods with statistical surrogate modeling. The idea is that the surrogate model contributes good exploration, and the numerical method contributes good exploitation. The two methods are combined in a way that balances exploration and exploitation, in order to efficiently find the global constrained minimum. Augmented Lagrangian (Sect. 4.2.3) and Barrier Methods (Sect. 4.2.4) are discussed in this chapter. Additional examples include the slack-variable augmented Lagrangian (Picheny et al. 2016), the ADMM algorithm for solving an augmented Lagrangian relaxation (Ariafar et al. 2019), Filter Methods

(Pourmohamad and Lee 2020), and scalable constrained BO based on trust regions (Eriksson and Poloczek 2021). Direct numerical optimization algorithms can be very efficient at finding a local optimum, and thus work quite well when run with a starting point in the domain of attraction of the global optimum, but can often become stuck in a local mode when started elsewhere. Statistical surrogate models can efficiently approximate the full surface. By using the surrogate model to guide the numerical algorithm, the hybrid approach can efficiently identify promising regions and find the local optimum of each, thus finding the global optimum as the best of the local optima. Some hybrid approaches work by iterating between the surrogate model and the numerical method. The ones we discuss in this chapter combine the two approaches into a single acquisition function.

4.2.1 Constrained Expected Improvement

A natural extension of the unconstrained acquisition functions to the case of constrained optimization is to impose some sort of restriction on where the unconstrained acquisition function can search. A simple, and intuitive, way to achieve this goal is to take an unconstrained acquisition function and weight its function value by the probability that the input satisfies the constraint. Perhaps the most well known use of this idea is that of constrained expected improvement (Schonlau et al. 1998; Gardner et al. 2014). As the name suggests, the unconstrained component of the acquisition function relies upon the EI acquisition function of Sect. 3.3.2, and for a given input, its corresponding EI value is weighted by the probability that the input satisfies the constraints. Thus, the constrained expected improvement (CEI) acquisition function can be formulated as follows

$$a_{\text{CEI}}(x) = \mathbb{E}\{I(x)\} \times \Pr(c(x) \leq 0), \tag{4.2}$$

where $\Pr(c(x) \leq 0)$ is the probability of satisfying the constraint. Here, the improvement function, $I(x)$, uses an $f^n_{\min}$ defined over the region of the input space where the constraint functions are satisfied since we are only concerned with improving upon the current solution in the valid region of the input space.

Fortunately, the derivation of $\mathbb{E}\{I(x)\}$ in equation (3.9) still holds in the constrained case. And so, all we are left to deal with is how to handle modeling the probability of satisfying the constraint. There is no one universal model used for calculating the probability, but rather, the choice of modeling strategy is usually dependent upon the type of values the constraint function returns. The two type of constraint functions that exists are those that return a continuous value and those that return a binary value. In the case of a continuous constraint, the constraint function provides a real-valued measure of constraint satisfaction. When the input does not satisfy the constraint the value returned gives a sense of how far away the input is to satisfying the constraint. For a continuous constraint function, it is simple enough to take the same approach that we do for the objective function, and model the

constraint function output using a Gaussian process. For a single constraint, one can use the posterior predictive distribution of the Gaussian process in order to calculate the $\Pr(c(x) \leq 0)$, i.e.,

$$\begin{aligned}\Pr(c(x) \leq 0) &= \int_{-\infty}^{0} N(Y_c(x); \mu_c(x), \sigma_c(x)) dY_c(x) \\ &= \Phi\left(\frac{-\mu_c(x)}{\sigma_c(x)}\right) \end{aligned} \tag{4.3}$$

where $Y_c(x)$ is the Gaussian process surrogate model for the constraint function, and this integral reduces to the Gaussian cumulative distribution function $\Phi(\cdot)$. In the case of multiple constraints, if we assume that the constraints are conditionally independent given x, then the probability of satisfying the constraints factorizes into

$$\Pr(c_1(x) \leq 0, \ldots, c_m(x) \leq 0) = \prod_{i=1}^{m} \Pr(c_i(x) \leq 0), \tag{4.4}$$

and everything proceeds as before in the case of the single constraint. In the case of dependent constraints, the joint probability $\Pr(c_1(x) \leq 0, \ldots, c_m(x) \leq 0)$ can be computed using numerical methods (Cunningham et al. 2013).

On the other hand, a continuous Gaussian process does little good when the constraint function only returns a binary outcome which specifies whether or not the constraint was satisfied. In the case of binary constraints, the available modeling choices become somewhat more interesting in that any classification model that can calculate the probability of belonging to one of two classes can be used. Here, in keeping with the spirit of BO, a natural choice might be to use a Gaussian process classification model, although other popular choices include such things as tree based classifiers like random forests (Breiman 2001) or Bayesian additive regression trees (BART) (Chipman et al. 2010), to name a few. Readers interested in learning more about Gaussian process classification models are encouraged to see (Rasmussen and Williams 2006).

Example Consider the following two-dimensional constrained optimization problem

$$\begin{aligned} \min \ & f(x_1, x_2) = 4x_1^2 - x_1 - x_2 - 2.5 \\ \text{s.t. } & c_1(x_1, x_2) = -x_2^2 + 1.5x_1^2 - 2x_1 + 1 \\ & c_1(x_1, x_2) = 3x_1^4 + x_2^2 - 2x_1 - 4.25 \end{aligned}$$

where $-1.5 \leq x_1 \leq 2.5$, and $-3 \leq x_2 \leq 3$. The optimal solution to the constrained problem is $f(x_1, x_2) = -4.6958$, which occurs along the border of the valid region at $(x_1, x_2) = (0.1708, 2.1417)$ (see Fig. 4.1). Although the functional forms are known in this example, we will treat the objective function, $f(x)$, as if it were an

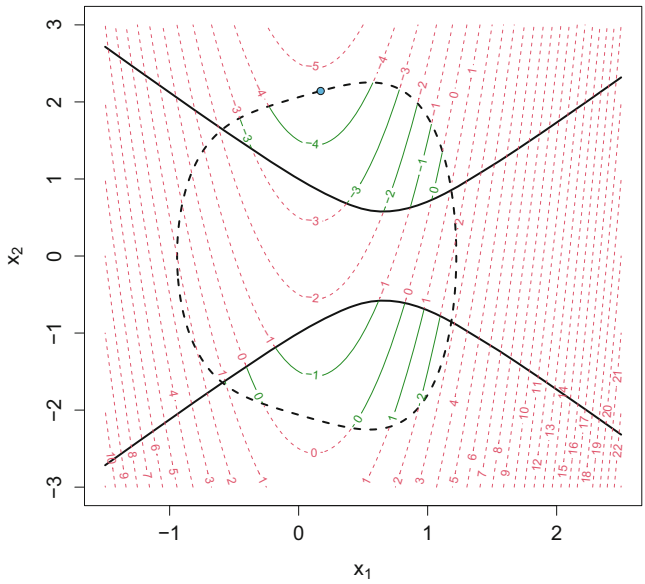

Fig. 4.1 Contours of the objective function colored by the two constraints. The solid black line denotes one constraint function, while the dashed black line denotes the other constraint function. Contours that are red are areas where the constraints are not satisfied, while green contours indicate areas where the constraints are satisfied. The blue point represents the global solution to the problem

expensive black-box function, and similarly for the two constraint functions, $c_1(x)$ and $c_2(x)$ so that we may solve it using constrained BO.

Following the general constrained BO algorithm outlined in Algorithm 2 of Sect. 4.1, we start with an initial LHS sample of size $n = 10$, and sequentially select 50 more inputs to evaluate based on the CEI acquisition function. Here we choose to model the objective and constraint functions using independent Gaussian process surrogate models, i.e., $Y_f(x)$ for the objective function, and $Y_{c_1}(x)$ and $Y_{c_2}(x)$ for the constraint functions. This modeling choice allows us to pick the next input to evaluate by maximizing the following easy to evaluate CEI acquisition function

$$\begin{aligned} a_{\text{CEI}}(x) &= \mathbb{E}\{I(x)\} \times \Pr(c(x) \leq 0) \\ &= \left[(f^n_{\min} - \mu_f(x))\Phi\left(\frac{f^n_{\min} - \mu_f(x)}{\sigma_f(x)}\right) + \sigma_n(x)\phi\left(\frac{f^n_{\min} - \mu_f(x)}{\sigma_f(x)}\right)\right] \\ &\quad \times \left[\prod_{i=1}^{2} \Phi\left(-\frac{\mu_{c_i}(x)}{\sigma_{c_i}(x)}\right)\right]. \end{aligned} \tag{4.5}$$

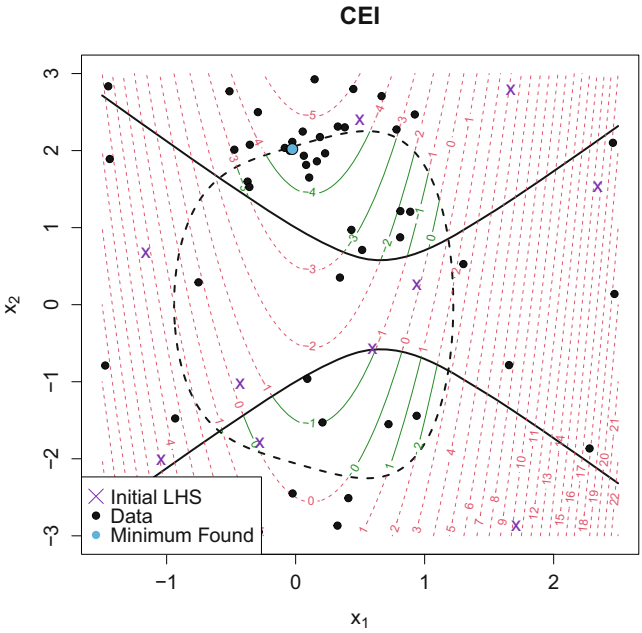

Fig. 4.2 A view of the performance of the BO algorithm using the constrained expected improvement (CEI) acquisition function for a single run of a Monte Carlo experiment

Given the setup, we proceed to search for the global solution to constrained optimization problem using the CEI acquisition function to choose the next 50 inputs to sequentially evaluate. Figure 4.2 shows the performance of the BO algorithm using the constrained expected improvement (CEI) acquisition function in order to give an idea of how the CEI acquisition searches the input space. Constrained optimization in this problem is hard due to the fact that the valid regions are small relative to the input space and disconnected. We see that with a starting LHS of size $n = 10$, only two inputs are selected in the valid region and that of the two disconnected valid regions, only one of them has any initial data. However, even without starting knowledge of the second valid region, the CEI algorithm performs quite well spending a lot of its effort searching near the global solution. Note that the CEI acquisition is just expected improvement weighted by the probability of being in the valid region, and so the CEI acquisition function can tend to explore invalid regions often when the estimate of the probability is either poor, or simply because the expected improvement is very high in the invalid region.

Obviously, the performance of the acquisition function for guiding the BO algorithm is highly dependent upon the initial sample used to initialize the algorithm. To get an idea of the performance, as well as the robustness, of the BO algorithm under the CEI acquisition function, we rerun the BO algorithm within 30 Monte Carlo experiments. The results of the 30 Monte Carlo experiments is captured in Fig. 4.3.

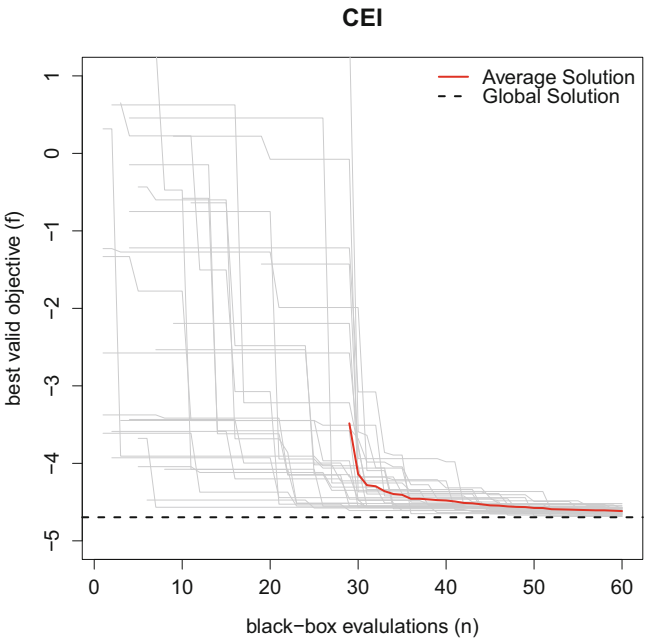

Fig. 4.3 A view of the performance of the BO algorithm using the constrained expected improvement (CEI) acquisition function for the 30 the Monte Carlo experiments. Here, each grey line represents the best value found over the search by the BO algorithm during a single run of the Monte Carlo experiment. The red average line starts when all of the 30 Monte Carlo experiments have found a valid solution

Note that, unlike the Monte Carlo progress plots in Chap. 3, the best valid average solution over the 30 Monte Carlo experiments need not be calculable for all of the black-box evaluations. This is due to the fact that each Monte Carlo experiment will potentially find its first valid objective function evaluation at a different point in time in the sequential evaluation process. For example, it is clear to see that the individual trajectories of the best valid solution (i.e., grey lines) all do not start at the same time and point. In fact, at least one of the 30 Monte Carlo initial LHSs is composed entirely of inputs selected in the invalid region, and so clearly the BO algorithm may even start with no knowledge of the valid regions. As we can see from Fig. 4.3, the worst of the 30 Monte Carlo experiments does not begin selecting inputs from the valid region until about the thirtieth input evaluation (the start of the red average solution curve). Thus, being able to locate or hone in on the valid region of the input space is critical to the success of the BO algorithm. We note that many publications in the literature assume knowledge of at least one point in the valid region, for example, by discarding an initial LHS without any valid points and generating new ones until a sample is obtained with at least one valid point. Such an approach leads to prettier plots, but it hides the computational expense of generating the additional LHSs.

4.2.2 Asymmetric Entropy

As discussed in Sect. 4.2, sometimes it is important to try to keep as many runs as possible inside the valid region while searching for the optimum. At the same time, the minimum may be expected to be along the constraint boundary, because the constraints are operating in opposition to the objective function. A well-studied approach for focusing on a boundary is to use entropy as a utility function.

Consider the problem of finding the constraint boundary by estimating the probability an input x will be inside the valid region, i.e.,

$$p(x) = \Pr(c(x) \leq 0). \tag{4.6}$$

We can estimate the boundary by finding x such that $p(x) = 0.5$. It turns out that we can recast the boundary-finding problem as an unconstrained BO problem. We can create a surrogate model for p, such as a classification Gaussian process or any other classifier. A common utility function for this search would be the Shannon entropy:

$$S(x) = -p(x) \times \log(p(x)) - (1 - p(x)) \times \log(1 - p(x)). \tag{4.7}$$

Using the expected value of $S(x)$ under the surrogate as the acquisition function leads to a BO algorithm for finding the boundary.

Our actual problem of interest is constrained optimization, which here is focused on finding an optimum along a boundary. Thus we may want to include entropy as part of an acquisition function, such as taking a product of EI and entropy. We might want to more heavily weight either EI or entropy, and thus a family of acquisition functions is

$$a(x) = \mathrm{EI}(x)^{\omega_1} \times S(x)^{\omega_2}, \tag{4.8}$$

where ω_1 and ω_1 act as weights for EI and entropy, respectively. As this acquisition function combines EI and entropy, it will tend to explore the promising space just beyond the boundary, primarily in the invalid region, where EI will be larger. The entropy term pulls it closer to the boundary, but not enough to pull it into the valid region. To address this imbalance, Lindberg and Lee (2015) introduced an acquisition function based on asymmetric entropy, which can focus the search inside the valid region. Asymmetric entropy was originally created by Marcellin et al. (2006) in the context of fitting decision trees when one class is relatively uncommon and thus can benefit from being favored in the tree growth algorithm. Asymmetric entropy is given by

$$S_a(x) = \frac{2p(x)(1 - p(x))}{p(x) - 2wp(x) + w^2}, \tag{4.9}$$

where w is a mode location parameter. Maximum asymmetric entropy is achieved at $p = w$ instead of the usual $p = 0.5$. Thus selecting $w > 0.5$ will push exploration more toward the valid region. Lindberg and Lee (2015) recommend $w = 2/3$. Substituting asymmetric entropy into the acquisition function, we now have

$$a_{\text{AE}}(x) = \text{EI}(x)^{\omega_1} \times S_a(x)^{\omega_2}.$$

Lindberg and Lee (2015) recommend $\omega_1 = 1$ and $\omega_2 = 5$, giving the acquisition function

$$a_{\text{AE}}(x) = \text{EI}(x) \times S_a(x)^5. \tag{4.10}$$

This formulation can be effective at solving the constrained optimization problem while searching points primarily inside the valid region.

Example Returning to the two-dimensional constrained optimization example in Sect. 4.2.1, consider solving again the following problem

$$\begin{aligned} \min \ & f(x_1, x_2) = 4x_1^2 - x_1 - x_2 - 2.5 \\ \text{s.t. } & c_1(x_1, x_2) = -x_2^2 + 1.5x_1^2 - 2x_1 + 1 \\ & c_1(x_1, x_2) = 3x_1^4 + x_2^2 - 2x_1 - 4.25 \end{aligned}$$

where $-1.5 \leq x_1 \leq 2.5$, and $-3 \leq x_2 \leq 3$. Proceeding as before, we start with an initial LHS sample of size $n = 10$, and sequentially select 50 more inputs to evaluate based on the AE acquisition function. Given that both constraint functions return a continuous value, we choose to model the constraints using independent Gaussian process surrogate models, $Y_{c_1}(x)$ and $Y_{c_2}(x)$, and their respective predictive means and variances to calculate the probability $p(x)$ as

$$p(x) = \Pr(c(x) \leq 0) = \Phi\left(-\frac{\mu_{c_1}(x)}{\sigma_{c_1}(x)}\right) \times \Phi\left(-\frac{\mu_{c_2}(x)}{\sigma_{c_2}(x)}\right). \tag{4.11}$$

We then can calculate the asymmetric entropy by plugging $p(x)$ into $S_a(x)$ and setting $w = 2/3$. Likewise, we follow the recommendation set forth in Lindberg and Lee (2015) and set $\omega_1 = 1$, and $\omega_2 = 5$ to finish the specification of the acquisition function.

Running the BO algorithm under these settings, Fig. 4.4 shows the performance of the BO algorithm using the AE acquisition function. Recall that when $w > 0.5$, asymmetric entropy will push the search towards the inside of the boundary of the valid region. What we see in Fig. 4.4 is exactly that. When searching within the valid regions, the AE acquisition function pushes the search towards the boundaries of the valid regions when exploring.

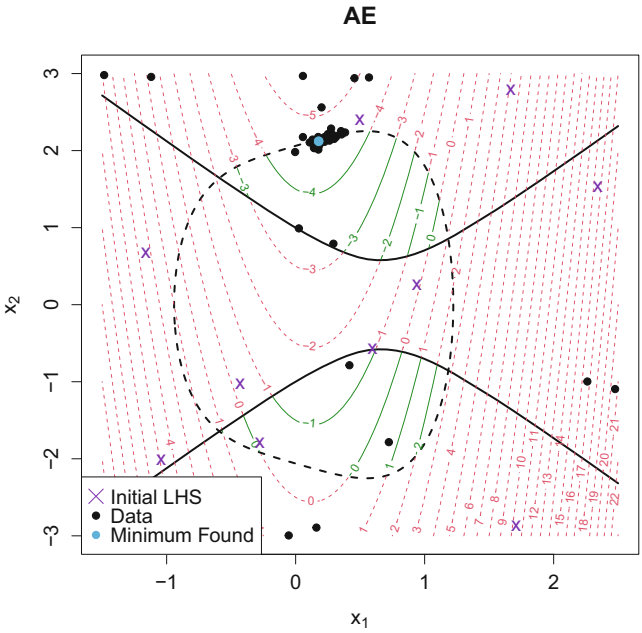

Fig. 4.4 A view of the performance of the BO algorithm using the asymmetric entropy (AE) acquisition function for a single run of a Monte Carlo experiment

To get an idea of the overall performance of the BO algorithm using the AE acquisition function, we run 30 Monte Carlo experiments of the BO algorithm. Figure 4.5 shows the individual and average performance of the BO algorithm over the 30 Monte Carlo experiments. It is not until after the twentieth input evaluation that the worst of the Monte Carlo experiments has found an input in a valid region; however, every single solution search path seems to hone in on the global solution very quickly. By about the twenty fifth iteration of the algorithm, all of the Monte Carlo runs have converged to the global minimum of the problem. Clearly, using the AE acquisition function is advantageous when the solution to the constrained optimization problem lies along the boundary of the input space since the natural behavior of the acquisition function is to want to search for the boundary and explore along it.

4.2.3 Augmented Lagrangian

Augmented Lagrangian methods (Bertsekas 1982; Nocedal and Wright 2006) are a class of methods based on using a penalty function to combine constraint satisfaction with objective function optimization into a new single scalar acquisition

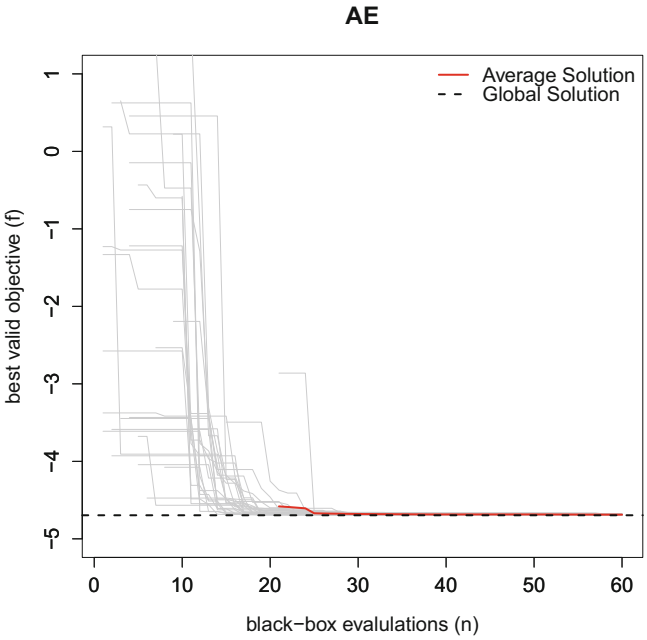

Fig. 4.5 A view of the performance of the BO algorithm using the asymmetric entropy (AE) acquisition function for the 30 the Monte Carlo experiments. Here, each grey line represents the best value found over the search by the BO algorithm during a single run of the Monte Carlo experiment. The red average line starts when all of the 30 Monte Carlo experiments have found a valid solution

function, which can then be solved as a sequence of unconstrained optimization problems. The Augmented Lagrangian approach combines the original objective function with a penalty parameter multiplied by the constraint function, plus a Lagrangian term, and seeks to minimize that combination. We work here with the negative of that term as the acquisition function that we maximize:

$$a_{\text{AL}}(x; \lambda, \rho) = -f(x) - \lambda^T c(x) - \frac{1}{2\rho} \sum_{j=1}^{m} \max(0, c_j(x))^2, \tag{4.12}$$

where f is the objective function, c is the vector of constraint functions, $\rho > 0$ is the penalty parameter, and $\lambda \in \mathbb{R}_+^m$ is the Lagrange multiplier. As we search the space, for given values of λ and ρ, we choose the next point x^* as

$$x^* = \underset{x \in \mathcal{X}}{\text{argmax}}\, a_{\text{AL}}(x; \lambda, \rho). \tag{4.13}$$

If we were to fix λ and ρ, this approach would convert the constrained optimization problem into an unconstrained optimization problem, typically simplifying the

problem. In practice, we use a sequence of values for λ and ρ, and so we have converted a constrained problem to a sequence of unconstrained problems.

The augmented Lagrangian approach has good theoretical convergence properties when $\rho \to 0$. However, as $\rho \to 0$, the problem becomes increasingly ill-conditioned, and difficult to solve numerically. Thus real-world algorithms use a sequence of values for ρ whose limit is 0. As ρ is updated, λ is updated at iteration k as a typical Lagrange multiplier

$$\lambda_j^k = \max\left(0, \lambda_j^{k-1} + \frac{1}{\rho^{k-1}} c_j(x^k)\right). \tag{4.14}$$

Augmented Lagrangian methods were developed as numerical methods, designed for direct numerical optimization. Gramacy et al. (2016) adapted the augmented Lagrangian approach for Bayesian optimization by incorporating Gaussian process surrogate modeling into the algorithm. An independent Gaussian process is used to approximate f and each constraint c_j, for a total of $m + 1$ Gaussian process models. In particular, Y_f is a Gaussian process surrogate for f and Y_{c_j} is a Gaussian process surrogate for c_j. These surrogates are important because (4.13) requires the selection of the next point, x^*, based on the unknown f and c. The Gaussian process surrogates are used to guide this choice of x^*. The acquisition function is thus approximated with:

$$-Y_f(x) - \lambda^T Y_c(x) - \frac{1}{2\rho} \sum_{j=1}^{m} \max(0, Y_{c_j}(x))^2. \tag{4.15}$$

Given this approximation, how does one choose the next x^* in a BO routine? As we update our Gaussian process surrogate, Y will have a distribution derived from the posterior distributions of each of the Gaussian processes, and so Y is not a scalar function that can be directly minimized. Gramacy et al. (2016) suggest several methods for guiding this choice, stemming from two conceptual approaches: following the predictive mean, or expected improvement.

The first approach for choosing the point, x^*, that maximizes (4.12) based on the approximation (4.15) is to choose the x^* that maximizes the posterior predictive mean of (4.15). At any point x, $Y_f(x)$ will have a posterior predictive mean that is Gaussian with mean $\mu_f(x)$ and variance $\sigma_f^2(x)$. Similarly, each $Y_{c_j}(x)$ will have a Gaussian posterior predictive mean with mean $\mu_{c_j}(X)$ and variance $\sigma_{c_j}^2(x)$. So we can now write

$$\begin{aligned}\mathbb{E}\left[a_{\text{AL}}(x)\right] &\approx -\mathbb{E}\left[Y_f(x)\right] - \lambda^T \mathbb{E}\left[Y_c(x)\right] - \frac{1}{2\rho} \sum_{j=1}^{m} \mathbb{E}\left[\max(0, Y_{c_j}(x))^2\right] \\ &= -\mu_f(x) - \lambda^T \mu_c(x) - \frac{1}{2\rho} \sum_{j=1}^{m} \mathbb{E}\left[\max(0, Y_{c_j}(x))^2\right]. \end{aligned} \tag{4.16}$$

Gramacy et al. (2016) provide an expansion of that last expectation as

$$\mathbb{E}\left[\max(0, Y_{c_j}(x))^2\right] = \sigma_{c_j}^2(x)\left\{\left(1+\left(\frac{\mu_{c_j}(x)}{\sigma_{c_j}(x)}\right)^2\right)\Phi\left(\frac{\mu_{c_j}(x)}{\sigma_{c_j}(x)}\right)+\phi\left(\frac{\mu_{c_j}(x)}{\sigma_{c_j}(x)}\right)\right\}, \tag{4.17}$$

where Φ and ϕ are the cumulative distribution function and probability density function for the standard Gaussian distribution, respectively.

The second approach is based on expected improvement, choosing the x^* that has the largest expected improvement in Y. Because this EI is not available in closed form, Gramacy et al. (2016) suggest a Monte Carlo approximation. Draw T Monte Carlo samples $y_f^{(t)}(x), y_{c_1}^{(t)}(x), \ldots, y_{c_m}^{(t)}(x)$ from Gaussian distributions $N\left(\mu_f(x), \sigma_f^2(x)\right)$ and $N\left(\mu_{c_j}(x), \sigma_{c_j}^2(x)\right)$, for $t = 1, \ldots, T$. The approximation is

$$\mathbb{E}\left[\mathbb{1}_Y(x)\right] \approx \frac{1}{T}\sum_{t=1}^{T}\max\left\{0, y_{\min}-\left[y_f^{(t)}(x)+\lambda^T y_c^{(t)}(x)+\frac{1}{2\rho}\sum_{j=1}^{m}\max\left(0, y_{c_j}^{(t)}(x)\right)^2\right]\right\} \tag{4.18}$$

where $y_{\min}$ is the smallest value of (4.15) that has been observed across all previous iterations.

For both approaches, full optimization would be impractical, and x^* can be chosen by drawing a random sample of candidates and choosing the x^* which is the best among the candidates evaluated. Gramacy et al. (2016) provide additional discussion on generating improved candidate sets. They also discuss variations on these two approaches that approximate the acquisition functions by removing the max operator, which leads to simplified expressions that can be written in closed form in certain cases.

Example Using the AL acquisition function, let us again solve the following two-dimensional constrained optimization problem

$$\begin{aligned} \min \;\; & f(x_1, x_2) = 4x_1^2 - x_1 - x_2 - 2.5 \\ \text{s.t. } & c_1(x_1, x_2) = -x_2^2 + 1.5x_1^2 - 2x_1 + 1 \\ & c_1(x_1, x_2) = 3x_1^4 + x_2^2 - 2x_1 - 4.25 \end{aligned}$$

where $-1.5 \leq x_1 \leq 2.5$, and $-3 \leq x_2 \leq 3$. Starting with an initial LHS of size $n = 10$, we run the BO algorithm using the AL acquisition function to sequentially choose the next 50 inputs to evaluate. Figure 4.6 shows the performance of the BO algorithm over a single Monte Carlo run. The AL acquisition function performs quite well in this example, spending most of its time exploring input space of one of the valid regions. Unlike the AE acquisition function, the AL acquisition function

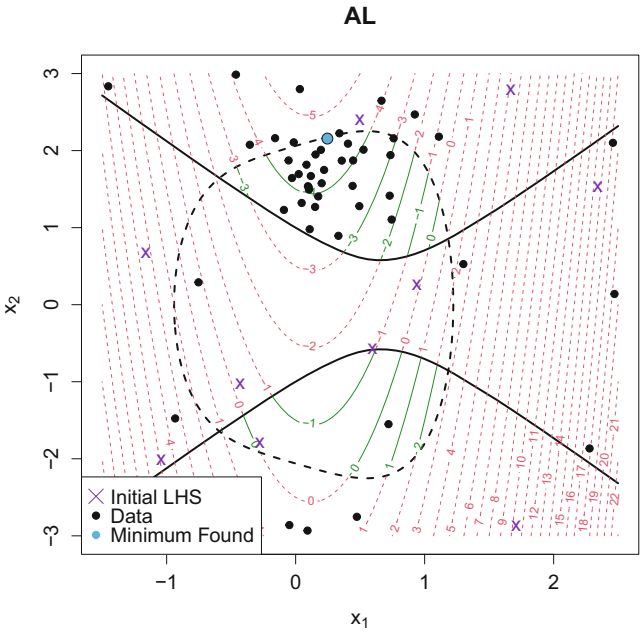

Fig. 4.6 A view of the performance of the BO algorithm using the augmented Lagrangian (AL) acquisition function for a single run of a Monte Carlo experiment

does not have a preference for solely trying to search the boundary of the input space and so it takes a more exploratory approach to searching the valid region.

Similar to the CEI acquisition function, it takes about 30 input evaluations before all of the 30 Monte Carlo experiments find a valid input (Fig. 4.7). However, the AL acquisition function still does a quite good job at converging, on average, to the global solution of the problem.

4.2.4 Barrier Methods

Barrier methods (Nocedal and Wright 2006), also known as interior point methods, are a natural strategy for solving black-box constrained optimization problems as they try to decrease the objective function as much as possible while ensuring that the boundary of the constraint space is never crossed. In order to ensure that the boundary of the constraint space is never crossed, barrier methods replace the inequality constraints in the constrained optimization problem in (4.1) with an extra term in the objective function that can be viewed as a penalty for approaching the

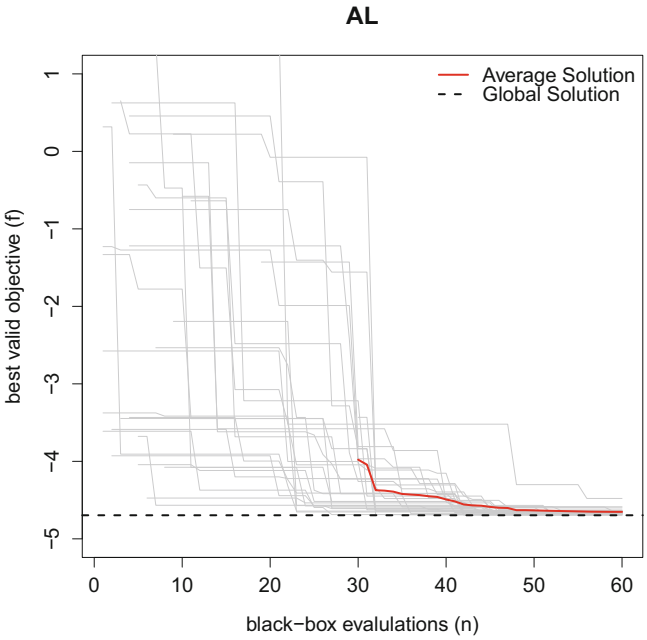

Fig. 4.7 A view of the performance of the BO algorithm using the augmented Lagrangian (AL) acquisition function for the 30 the Monte Carlo experiments. Here, each grey line represents the best value found over the search by the BO algorithm during a single run of the Monte Carlo experiment. The red average line starts when all of the 30 Monte Carlo experiments have found a valid solution

boundary. Here, the constrained optimization problem in (4.1) can be re-written as the following unconstrained optimization problem

$$\min_{x} \left\{ f(x) + \sum_{i=1}^{m} \mathbf{B}_{\{c_i(x) \leq 0\}}(x) \right\}, \tag{4.19}$$

where $\mathbf{B}_{\{c_i(x) \leq 0\}}(x) = 0$ if $c_i(x) \leq 0$ and ∞ otherwise. Although mathematically equivalent, the introduction of $\mathbf{B}_{\{c_i(x) \leq 0\}}(x)$ in the reformulation of the original constrained optimization problem is not particularly useful as it introduces an abrupt discontinuity when $c_i(x) > 0$. This discontinuity eliminates the use of calculus to minimize (4.19). To remedy this issue, the discontinuous function in (4.19) can be replaced with a continuous approximation, $\xi(x)$, that is ∞ when $c_i(x) > 0$ but is finite for $c_i(x) \leq 0$. This continuous approximation, $\xi(x)$, is referred to as the barrier function as it will create a “barrier" to exiting the valid region for the

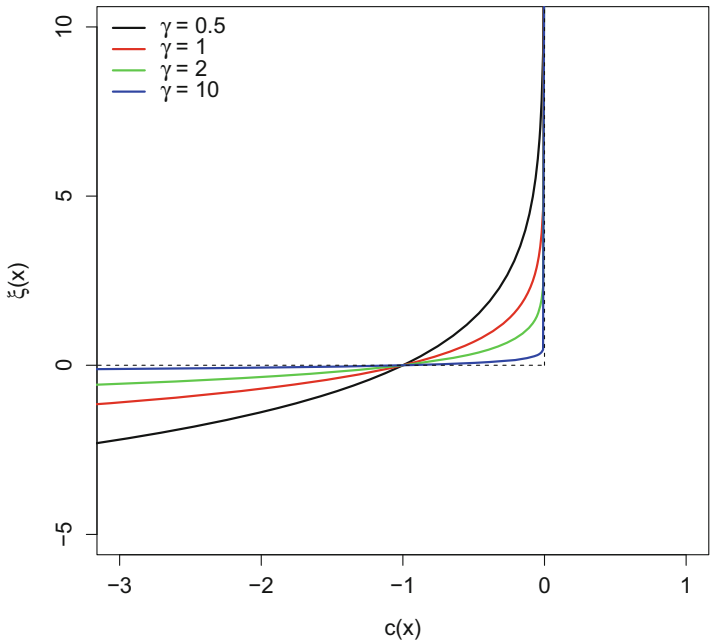

Fig. 4.8 As γ approaches ∞, the log barrier function, $\xi(x)$, becomes a better approximation to $\mathbf{B}_{\{c_i(x)\leq 0\}}(x)$ (i.e., the dashed line)

search algorithm. Many barrier functions exists, however, a popular choice of barrier function is the log barrier function which is defined as

$$\xi(x) = -\left(\frac{1}{\gamma}\right)\sum_{i=1}^{m}\log(-c_i(x)) \tag{4.20}$$

for $\gamma > 0$. Note that the log barrier function, $\xi(x)$, is a smooth approximation of $\sum_{i=1}^{m}\mathbf{B}_{\{c_i(x)\leq 0\}}(x)$ when $c_i(x) < 0$, and that this approximation improves as γ goes to ∞ (see Fig. 4.8). Now, replacing $\mathbf{B}_{\{c_i(x)\leq 0\}}(x)$ with the log barrier function, $\xi(x)$, we can approximate the problem in (4.19) as

$$\min_x\{B(x;\gamma)\} = \min_x\left\{f(x) - \left(\frac{1}{\gamma}\right)\sum_{i=1}^{m}\log(-c_i(x))\right\}. \tag{4.21}$$

Solving the minimization problem in (4.21) becomes a much more manageable and tractable problem as compared to the minimization problem in (4.19).

Recognizing the attractive qualities of barrier methods for constrained optimization, Pourmohamad and Lee (2021) extended barrier methods to the BO framework by modeling the quantity $B(x;\gamma)$ in (4.21), using independent Gaussian process

surrogates $Y_f(x)$ and $Y_c(x) = (Y_{c_1}(x), \ldots, Y_{c_m}(x))$ for the objective and constraint functions, i.e.

$$Y(x) = Y_f(x) - \left(\frac{1}{\gamma}\right) \sum_{i=1}^{m} \log(-Y_{c_i}(x)). \tag{4.22}$$

Pourmohamad and Lee (2021) suggests that optimization can then proceed by minimizing the predictive mean surface of $Y(x)$. At first glance, this may sound like a poor idea since minimizing the expectation of a Gaussian process typically leads to a greedy search algorithm (i.e., think back to the discussion in Sect. 3.2 about minimizing the predictive mean of the Gaussian process, $\mu_n(x)$). However, as will be shown, in this case minimizing the predictive mean surface of $Y(x)$ leads to an acquisition function that searches the space both locally and globally. The result of minimizing the predictive mean surface of $Y(x)$ results in:

$$\min_x \mathbb{E}(Y(x)) \approx \min_x \mu_f(x) - \left(\frac{1}{\gamma}\right) \sum_{i=1}^{m} \left(\log(-\mu_{c_i}(x)) + \frac{\sigma^2_{c_i}(x)}{2\mu^2_{c_i}(x)} \right) \tag{4.23}$$

The details of the derivation of the expectation in (4.23) can be found in Pourmohamad and Lee (2021). To finally recast this in the language of BO, we instead pivot to maximizing the negative value of this equation and thus establish the barrier method (BM) acquisition function, i.e.,

$$a_{\text{BM}}(x) = -\mu_f(x) + \left(\frac{1}{\gamma}\right) \sum_{i=1}^{m} \left(\log(-\mu_{c_i}(x)) + \frac{\sigma^2_{c_i}(x)}{2\mu^2_{c_i}(x)} \right). \tag{4.24}$$

Two problems arise from this acquisition function. The first problem is that there is no explicit rule, in the context of BO, on how to set γ. In the mathematical programming literature (e.g., Nocedal and Wright (2006)), it is common practice to have the value of $\gamma \to \infty$ such that, at iteration $k+1$ of the barrier method, $\gamma_{k+1} > \gamma_k$. In effect, this leads to steadily decreasing the penalty for approaching the boundary of the valid region throughout the optimization. However, much like the LCB acquisition function of Sect. 3.3.3, γ is still a tuning parameter left to user's discretion. The second problem is that (4.24) contains no variability term associated with the objective function, but rather only with the constraints, i.e., $\sigma^2_{c_i}(x)$. Without a term like $\sigma^2_f(x)$ in (4.24) to measure the prediction uncertainty for the objective function, the acquisition function will tend to favor exploitation, rather than exploration, as it will assume that it is predicting the objective function at untried inputs without error. Solving both of these problems at once, Pourmohamad and Lee (2021) recommended setting $\gamma = 1/\sigma^2_f(x)$, where $\sigma^2_f(x)$ is the predictive

variance associated with the Gaussian process surrogate model for the objective function f. This leads to the revised acquisition function

$$a_{\text{BM}}(x) = -\mu_f(x) + \sigma_f^2(x) \sum_{i=1}^{m} \left(\log(-\mu_{c_i}(x)) + \frac{\sigma_{c_i}^2(x)}{2\mu_{c_i}^2(x)} \right), \tag{4.25}$$

which (1) gives a rule for setting γ based on the current level of uncertainty in the predictions, and (2) injects an uncertainty term for the objective function into the acquisition function.

A second approach was also proposed in Pourmohamad and Lee (2021) which was to replace the surrogate model for the objective function, $Y_f(x)$, with the improvement function $-I(x)$ in (4.22), i.e.,

$$\min_x \mathbb{E}\left(-I(x) - \left(\frac{1}{\gamma}\right) \sum_{i=1}^{m} \log(-c_i(x)) \right) = \min_x -\mathbb{E}(I(x)) - \left(\frac{1}{\gamma}\right) \sum_{i=1}^{m} \left(\log(-\mu_{c_i}) + \frac{\sigma_{c_i}^2}{2\mu_{c_i}^2} \right). \tag{4.26}$$

The idea here being that if you instead take the expectation of the improvement function in (4.26), that you would incur all of the benefits of the expected improvement acquisition function, i.e., a variance term for the objective function and the natural exploration-exploitation search characteristics. Note that since we are minimizing in (4.23) we will need to use the negative improvement function. The minimization problem in (4.26) leads to the following acquisition function

$$\begin{aligned} a_{\text{BM}}(x) = (f_{\min}^n - \mu_f(x))\Phi\left(\frac{f_{\min}^n - \mu_f(x)}{\sigma_f(x)} \right) + \sigma_f(x)\phi\left(\frac{f_{\min}^n - \mu_f(x)}{\sigma_f(x)} \right) \\ + \left(\frac{1}{\gamma}\right) \sum_{i=1}^{m} \left(\log(-\mu_{c_i}) + \frac{\sigma_{c_i}^2(x)}{2\mu_{c_i}^2(x)} \right), \end{aligned} \tag{4.27}$$

where again, Pourmohamad and Lee (2021) suggest setting $\gamma = 1/\sigma_f^2(x)$.

Example Using the BM acquisition function, we solve one last time the following two-dimensional constrained optimization problem

$$\begin{aligned} \min \ & f(x_1, x_2) = 4x_1^2 - x_1 - x_2 - 2.5 \\ \text{s.t. } & c_1(x_1, x_2) = -x_2^2 + 1.5x_1^2 - 2x_1 + 1 \\ & c_1(x_1, x_2) = 3x_1^4 + x_2^2 - 2x_1 - 4.25 \end{aligned}$$

where $-1.5 \leq x_1 \leq 2.5$, and $-3 \leq x_2 \leq 3$. Once again, we start with an initial LHS of size $n = 10$, and sequentially pick 50 additional inputs to evaluate based on the BM acquisition function. For illustration here, we shall use the form of the acquisition function in (4.25). The performance of the BM acquisition function

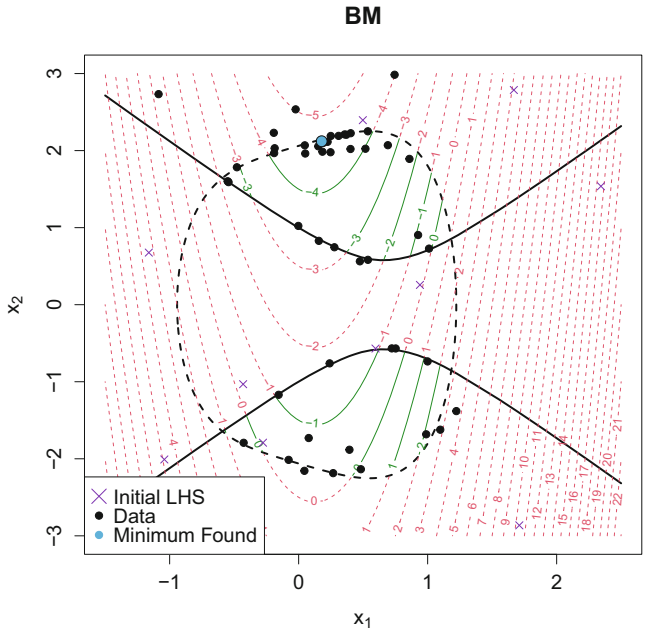

Fig. 4.9 A view of the performance of the BO algorithm using the barrier method (BM) acquisition function for a single run of a Monte Carlo experiment

for guiding the BO algorithm can be seen in Fig. 4.9. As opposed to the CEI, AE, and AL acquisition functions, the BM acquisition places higher importance on trying to stay within the valid region, and so exploring the invalid regions far less. This characteristic can be both desirable and undesirable. Staying within the valid region makes a lot of sense since we are concerned with finding a valid solution to the problem and, based on how computationally expensive the computer model is, searching in the invalid region can be viewed as wasteful since those inputs will not be the solution to the optimization problem and their evaluation is costly. On the other hand, as seen in Sects. 4.2.2 and 4.2.3, evaluating inputs in the invalid region is beneficial with helping the Gaussian process surrogate models learn both the objective and constraint function surfaces better, which ultimately leads to better prediction and uncertainty reduction, both of which are important components of a good acquisition function. Much like the AE acquisition function, the BM acquisition as well has a tendency to explore the boundary of the valid regions. Note though that while the original barrier methods were designed to never cross the border of the valid region, the BM acquisition function explores along the boundary, sometimes crossing it, since the location of the boundary is being estimated, and the Gaussian process surrogate model learns the boundary by sometimes going just beyond it.

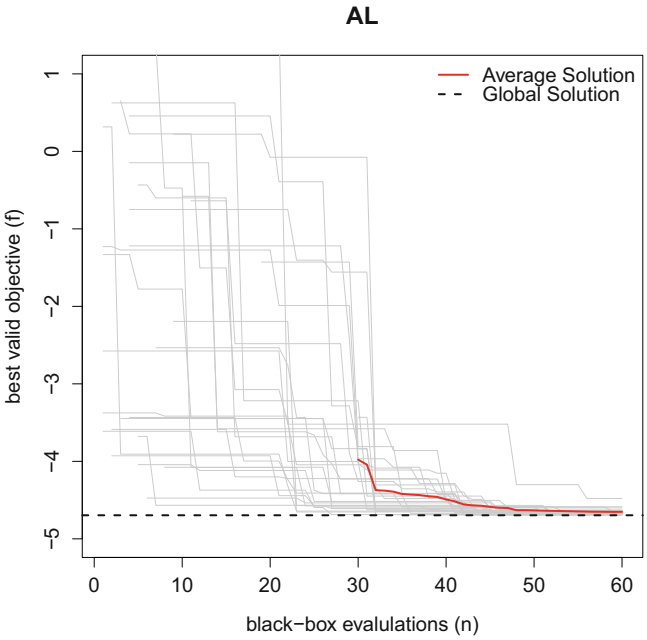

Fig. 4.10 A view of the performance of the BO algorithm using the barrier method (BM) acquisition function for the 30 the Monte Carlo experiments. Here, each grey line represents the best value found over the search by the BO algorithm during a single run of the Monte Carlo experiment. The red average line starts when all of the 30 Monte Carlo experiments have found a valid solution

Figure 4.10 captures the behavior of 30 Monte Carlo experiments for running the BO algorithm using the BM acquisition function. Overall, the performance of the BM acquisition function is quite good, with all of the 30 different Monte Carlo runs obtaining a valid input by about the twentieth input evaluation. After the initial LHS, the BM acquisition steadily guides the BO algorithm to the global solution of the problem on average.

For sake of comparison, we plot the average performance of the CEI, AE, AL, and BM algorithms over their respective 30 Monte Carlo experiments (Fig. 4.11). Here, the initial 30 LHSs, across the Monte Carlo experiments, are the same starting inputs for each acquisition function. What we see from Fig. 4.11 is vastly different average performance of the BO algorithm under the four different acquisition functions. Here, BM and AE are much better at finding valid inputs earlier on which is due to their tendencies to approach, or stay within, the boundaries of the valid region. CEI and AL are slower to find valid inputs due to the fact that they will allow for more exploration of the invalid region as compared to the AE and BM acquisition functions, i.e., there is no heavy penalty for exiting the valid region or need to search out the boundary of the valid region. Although the CEI and AL acquisition functions are slower to find valid starting inputs, they still perform as

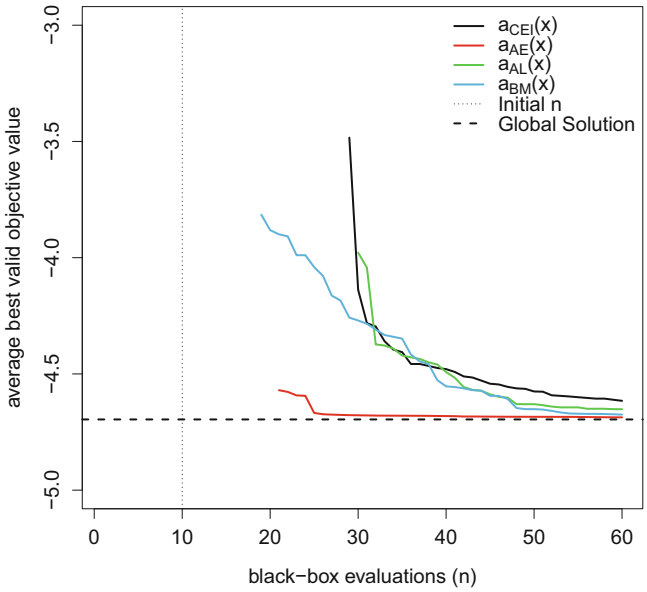

Fig. 4.11 The average performance of the CEI, AE, AL, and BM acquisition functions over the 30 Monte Carlo experiments

well as the BM acquisition function in this example. On the other hand, the AE acquisition function clearly dominates the other three acquisition functions with regards to the number of function evaluations needed to converge to the global solution of the problem. This superior performance is due in part to the fact that the global solution lies exactly on the border of the valid region which is exactly where the AE acquisition function wants to search. We emphasize that this a single example, and that on other examples, a different one of these methods may perform best.

4.3 Constrained Sprinkler Computer Model

Recall that in Sect. 3.4 we maximized the range of the garden sprinkler via an unconstrained optimization problem. Given that the garden sprinkler computer model returns a multi-objective output, we can now optimize one of the objective function outputs subject to the constraint that either one or both of the other objective function values is above (or below) a certain value. For sake of example, let us assume that still want to maximize the range of the garden sprinkler but now subject to the constraint that the water consumption must not be greater than five. Here we will assume that constraining the speed of the garden sprinkler (i.e., the third output of the garden sprinkler computer model) is not of concern. Casting the

garden sprinkler computer model in the framework of a constrained optimization, we formulate the problem as follows:

$$x^* = \underset{x \in \mathcal{X}}{\text{argmin}}\{-f(x)\} \text{ subject to } c(x) - 5 \leq 0, \tag{4.28}$$

Here the objective function, $f(x)$, describes the range at which the garden sprinkler can spray water, while the constraint function, $c(x)$, determines whether an acceptable amount of water is used. Note that since we wish to maximize the range of the garden sprinkler, we shall instead minimize the negative objective function in order to find the input that maximizes it. The inputs $x = (x_1, \ldots, x_8)^T \in \mathcal{X}$ represent the eight physical attributes of the garden sprinkler (see Fig. 1.6 and Table 1.1) that can be set within the computer model. The computer model is essentially a black-box function since, for any input configuration evaluated by the model, the only information that is returned is that of the objective and constraint values.

Now, we shall solve for the maximum value of the range of the garden sprinkler, subject to the water consumption constraint, using the constrained expected improvement (CEI), asymmetric entropy (AE), augmented Lagrangian (AL), and barrier method (BM) acquisition functions, and shall compare and contrast their performances. We will initialize the BO algorithm using a LHS of size $n = 10$, and sequentially evaluate an addition 90 inputs for a total computational budget of 100 input evaluations. In order to assess the robustness of the solutions of the BO algorithm, under the four different acquisition functions, we repeat solving this constrained optimization problem using 30 Monte Carlo experiments. Figure 4.12 shows the results of the 30 Monte Carlo experiments for a given acquisition function.

Visually, it looks like the CEI, AL, and BM acquisition functions all have similar performance, while the AE acquisition perhaps has a few better runs of the Monte Carlo experiments (i.e., lower best valid objective values) as well as fewer worst solutions. For each acquisition function, taking the average of the solutions over the 30 Monte Carlo experiments reveals that the AE acquisition function indeed performed better than the other acquisition functions (Fig. 4.13). Although it took much longer, on average, for the AE acquisition to start evaluating valid inputs, it still was capable of finding a significantly better solution than the other three acquisition functions. On the other hand, the average performance of the CEI, AL, and BM acquisition functions was very similar with perhaps the exception of the BM acquisition function doing slightly better (i.e., lower average values) through several stretches of black-box iterations.

Table 4.1 gives a numerical summary of the performance of the four acquisition functions. Besides being better on average, the AE acquisition function also did the best with regards to both the best and worst final solutions found over the 30 Monte Carlo experiments as compared to the CEI, AL, and BM acquisition functions. The observed differences in worst and final solutions between the CEI, AL, and BM acquisition functions, over the 30 Monte Carlo experiments, were negligible, reaffirming the overall comparable performance of the three acquisition functions.

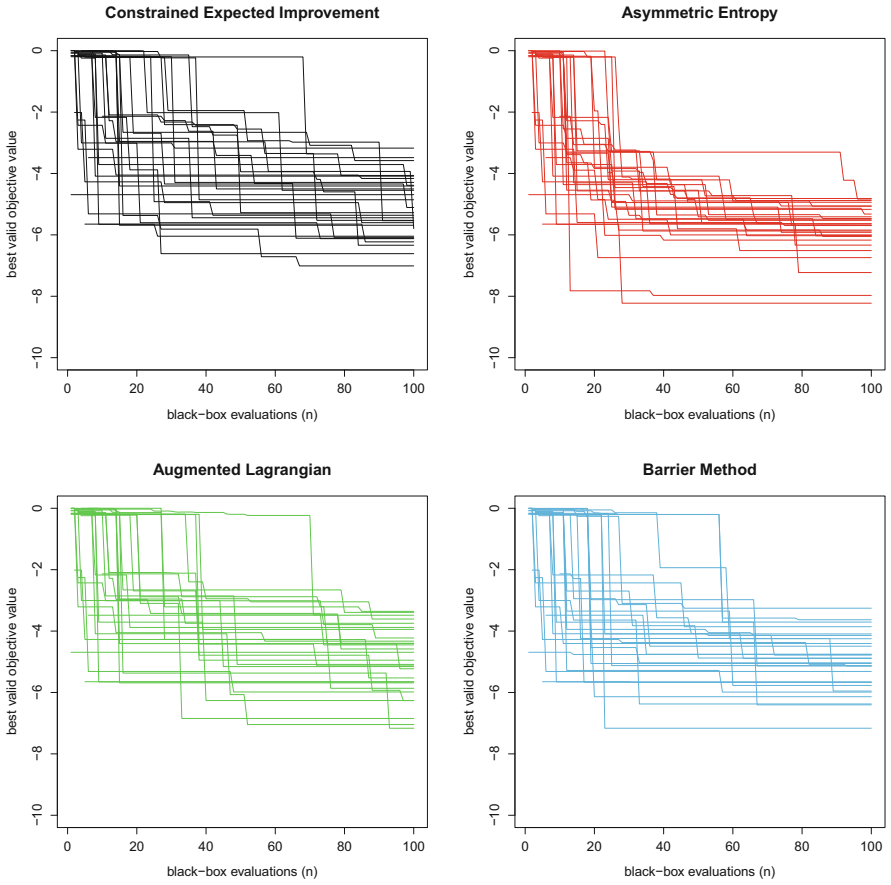

Fig. 4.12 A view of the performance of the BO algorithm, using the four different acquisition functions, for the 30 the Monte Carlo experiments. Here, each line represents the best value found over the search by the BO algorithm during a single run of the Monte Carlo experiment

Lastly, evaluating a LHS of size $n = 1,000,000$ inputs results in a global solution of -9.46. In this case, it is clear that the four acquisition functions have not yet converged to the global solution of the problem. This is consistent with the fact that in Fig. 4.13, all of the progress lines are still trending downward. Practically speaking, this means that the total budget for input evaluations needs to be increased past 100, and that the constrained BO algorithm may benefit from increasing the size of the initial LHS.

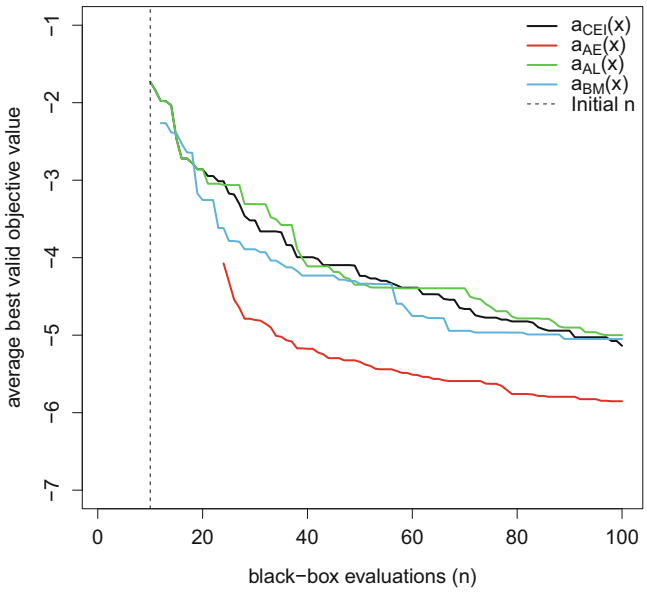

Fig. 4.13 The results of running 30 Monte Carlo experiments for each acquisition function. The plot shows the average best objective function values found over 100 black-box iterations

Table 4.1 The average, best, and worst solution found at the end of the 30 Monte Carlo experiments by each acquisition function

Acquisition function	Average final solution	Best final solution	Worst final solution
Constrained expected improvement	−5.14	−7.01	−3.17
Asymmetric entropy	−5.85	−8.22	−4.82
Augmented lagrangian	−5.00	−7.17	−3.36
Barrier method	−5.05	−7.16	−3.25