Stochastic mixed optimization for constrained resource optimization problems.

Larry Fenn and Felisa Vázquez-Abad

***Abstract*— Resource allocation problems as found in telecommunications or transportation require the determination of both the fewest resources needed to satisfy some quality condition and, implicitly, how to use those resources. In applications there is no closed form expression for the quality condition as a function of resource allocation, motivating a treatment of these problems as stochastic optimization problems. One feature, found in transportation problems and elsewhere, is that the resource parameter is a discrete variable. We present and compare non-gradient and gradient methods for solving these problems.**

I. Introduction

Consider the constrained optimization problem

$$\arg\min_{b\in\mathbb{N}} \left(\min_{u\in I\,:\,\mathcal{G}(b,u)\le c} C_b(u) \right) \tag{1}$$

where $I \subset \mathbb{R}^d$ is an interval, and for each value of b the function C_b is a cost function of the continuous parameter $u \in I$. The parameter b represents a "resource". The constraint function satisfies $\mathcal{G}(b,\cdot) \in C^2$ for any fixed b, and $\mathcal{G}(\cdot,u)$ is monotonic decreasing.

In many applications the (optimal) cost function increases with increasing resources, so the optimization problem reduces to a constraint satisfaction problem. Because $\mathcal{G}$ decreases with increasing b, then the original problem is equivalent to finding the solution of the simplified problem:

$$\arg\min_{b\in\mathbb{N}} \; f(b) \le c, \tag{2}$$

where $f(b) = \min_{u\in I} \; \mathcal{G}(b,u)$. We will assume that $f(b)$ is strictly decreasing (if not, then multiple values of resources yield constant satisfaction and ties can be broken by minimizing the value of b).

The **stochastic** optimization problem is to solve (1) when $\mathcal{G}$ is not available in closed form. We assume here that for any given values of b and u an estimate of $\mathcal{G}(b,u)$ may be obtained by simulation. In the general scenario (which we consider in our model), long simulations can approximate a stationary average value $\mathcal{G}(b,u)$. The focus of this research is how to efficiently find a solution with as few simulations as possible. We therefore try to find the least amount of simulations while exploring solutions, and seek early termination of sub-problems to determine the correct value of b.

This work is partially supported by the CUNY Institute CoSSMO

L. Fenn and F. Vázquez-Abad are with the Department of Computer Science, Hunter College and CUNY Institute for Computer Simulation, Stochastic Modeling and Optimization (CoSSMO), New York

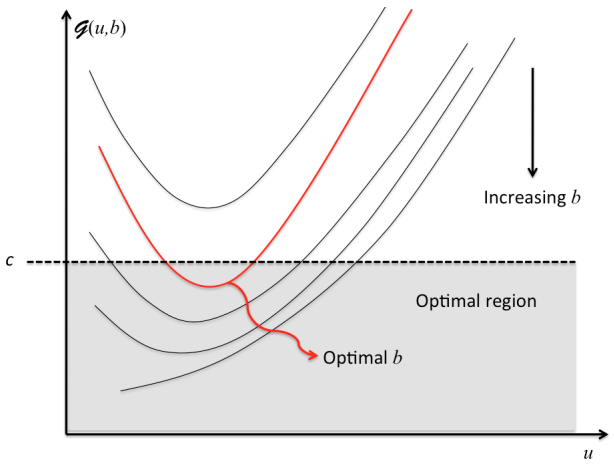

Fig. 1. Illustration of the constraint $\mathcal{G}(b,u)$.

Problems of this type include resource allocation for transit networks, with b as the number of buses and u as a control variable for headway between buses. Specifically, the problem that motivated the current research is studied in [1] and it dates back to 2005 when the Melbourne airport needed advice to buy a new fleet of buses that bring people to and from the terminals and the parking lots. It was implicit that an optimal scheduling (here represented by a quantity $u \in \mathbb{R}^+$ called the "headway") would be applied, so it was a problem of deciding how many buses b to buy, ensuring a quality of service. In particular, they required that at least 95% of the passengers should wait less than 10 minutes for a bus.

In [1] we defined a simplified model with Poisson arrivals of passengers to the system. Buses have finite capacity, they start empty at the bus depot and proceed to pick up passengers in the Arrivals terminal, then go to all the parking lots first unloading and then loading passengers. The final stop is at the Departures terminal where passengers unload. Every u units of time a new bus should depart from the depot, unless there are no more buses there. When there a few buses, both large or small values of u lead to larger probability of waiting more than 10 minutes, because all the buses will be on route most of the time and the bus depot will be empty, which explains the convexity of the function $\mathcal{G}(b,\cdot)$. As the number of buses increases, for each value of u this probability decreases.

A realistic mathematical model is too complex for analysis. Our ghost simulation method uses a different model where passengers are represented with fluid approximations. The model corresponds to a Filtered Monte Carlo model that accelerates the simulation time (see [1] for details). Importantly, many replications of a one-day simulation are required in order to evaluate the tail probability with reasonable precision for each value of (b, u). In that case the resource variable b is not too large and exhaustive search works, but the optimization on u requires long simulations. In the current paper we study the generalization of such problems and envision in particular problems where b may be very large. We consider both non-gradient and gradient methods.

Section II considers the deterministic scenario, for which we introduce the idea of early stopping the exploration phase for the minimization subproblem, in order to achieve the fastest convergence to the solution b^*, u^*. In Section III we present a method that adapts the binary search and golden section (GS) search algorithms into a stochastic context, which is applicable when only estimates of $\mathcal{G}$ are available and when $u \in \mathbb{R}$ (we are presently working on the extension of stochastic GS for multidimensional problems).

Performance of the algorithm is presented in Section IV, including probability of correct selection (PCS) for b^* and an estimate for the final error in $f(b^*)$. In some cases, estimates of the gradient $\nabla_u \mathcal{G}(b, u)$ may be easy to calculate in the simulation, and in this case we use a truncated gradient search on u instead of the GS method. In Section V an example application is described and comparisons are made between GS and gradient search methods. Conclusions and suggestions on further research are given in Section VI.

II. Deterministic Solution Method

First we will discuss the deterministic case. These results concern problems where $\mathcal{G}(b, u)$ function evaluation is completely deterministic; later we will extend them to cover cases involving stochastic processes.

A. Golden Section

Fix b; define $\mathcal{G}_b(u) = \mathcal{G}(b, \cdot)$. First, we will define a procedure to find the minimum of $\mathcal{G}_b(u)$:

Recall $\mathcal{G}_b(u) \in C^2[0, 1]$; suppose further that

$$0 < \mathcal{G}_b''(u) \leq \frac{1}{K_b} \tag{3}$$

Denote the minimum of $\mathcal{G}_b(u)$ as $\mathcal{G}_b(u^*)$. The following well-known procedure finds a neighborhood containing u^* of size δ [2]:

Require: $a < u^* < b$
  while $b - a > \delta$ **do**
    $u_1 = a + (b - a)(1 - \varphi)$
    $u_2 = a + (b - a)\varphi$
    if $\mathcal{G}_b(u_1) < \mathcal{G}_b(u_2)$ **then**
      $b = u_2$
    else
      $a = u_1$
    end if
  end while

In the constrained optimization case, the additional escape condition to the loop occurs when either $\mathcal{G}_b(u_1)$ or $\mathcal{G}_b(u_2)$ falls below the constraint.

With the assumptions from earlier, Taylor's theorem guarantees a bound on $|\mathcal{G}_b(u) - \mathcal{G}_b(u^*)|$ for all u in a neighborhood of u^* of size δ:

$$|\mathcal{G}_b(u) - \mathcal{G}_b(u^*)| \leq \frac{\delta^2}{2K_b} \tag{4}$$

Thus by choosing an algorithm parameter for error tolerance $\epsilon_b > 0$ we can guarantee finding an interval containing u^* such that $|\mathcal{G}_b(u) - \mathcal{G}_b(u^*)| < \epsilon_b$ by choosing $\delta_b < \sqrt{2K_b\epsilon_b}$. Let L_b represent the length of the domain of u for this fixed value of b. Since every iteration of the algorithm above shrinks the interval by a factor of $\varphi = \frac{1+\sqrt{5}}{2}$, we can determine the number of iterations required by replacing δ with $L_b\varphi^n$, the size of the search interval after n iterations:

$$\begin{aligned} L_b\varphi^n &< \sqrt{2K_b\epsilon_b} \\ n\log\varphi &< \frac{1}{2}\left(\log 2 + \log K_b + \log\epsilon_b - 2\log L_b\right) \\ -n\log\varphi &> \frac{1}{2}\left(-\log 2 - \log K_b - \log\epsilon_b + 2\log L_b\right) \end{aligned}$$

$\varphi < 1$ so $\log\varphi < 0$; thus $-\log\varphi > 0$:

$$\begin{aligned} n &> \frac{-1}{2\log\varphi}\left(-\log 2 - \log K_b - \log\epsilon_b + 2\log L_b\right) \\ n &> \frac{-1}{2\log\varphi}\left(-\log 2 + \log\frac{1}{K_b} + \log\frac{1}{\epsilon_b} + 2\log L_b\right) \end{aligned}$$

This is a lower bound on the number of iterations n required phrased in terms of a curvature bound, tolerance level, and initial search size. For future reference define the function $n(K, \epsilon, L)$:

$$n(K, \epsilon, L) = \left\lceil \frac{-1}{2\log\varphi}\left(-\log 2 + \log\frac{1}{K} + \log\frac{1}{\epsilon} + 2\log L\right)\right\rceil$$

Lastly, the number of points the function $\mathcal{G}_b(u)$ is evaluated at is equal to $n(K_b, \epsilon_b, L_b) + 1$ (since the first iteration requires the use of two points to get started, and all of the others exploit the property of φ to re-use information from prior iterations).

B. Binary Search

Since $f(b) = \min_{u \in I}\ \mathcal{G}(b, u)$ is assumed to be monotonic, the binary search algorithm can be used on $f(b)$. Moreover, since the binary search algorithm is predicated on a binary

test then in the constrained optimization problem it suffices to test if the constraint is *not* satisfied by $f(b)$. If we can find any value of u such that $\mathcal{G}_b(u)$ is below the constraint, then bisect one way; else, if $f(b)$ is above the constraint, bisect the other. As a prerequisite for the algorithm, finite bounds must be placed on b; let l be a sufficiently small integer such that $f(l) > c$ and let r be a sufficiently large integer such that $f(r) \leq c$:

Require: $f(r) \leq c < f(l)$
  while $r - l > 1$ **do**
    $b = \left\lfloor \frac{r+l}{2} \right\rfloor$
    if $f(b) > c$ **then**
      $l = b$
    else
      $r = b$
    end if
  end while

If we let B represent the length of the domain of b then the number of iterations for the algorithm to terminate is bounded above by $\lceil \log_2 B \rceil$; however, this is the number of $f(b)$ function evaluations required. The number of $\mathcal{G}(b,u)$ function evaluations required is therefore bounded above at:

$$\lceil \log_2 B \rceil \left(n(K, \epsilon, L) + 1 \right) \tag{5}$$

where $K \stackrel{\text{def}}{=} \min K_b$, $\epsilon \stackrel{\text{def}}{=} \min \epsilon_b$, and $L \stackrel{\text{def}}{=} \max L_b$.

III. Stochastic Method

A. Model

Suppose now that for each fixed choice of $b \in \mathbb{N}, u \in I$ there is an underlying Markov process $\{\xi_n(b,u)\}$ with a function defined on the process $g\left(\xi_n(b,u)\right)$. Now define $\mathcal{G}(b,u)$, the function in the constrained optimization problem, as follows:

$$\hat{\mathcal{G}}_N(b,u) = \frac{1}{N} \sum_{n=1}^{N} g\left(\xi_n(b,u)\right) \tag{6}$$

$$\mathcal{G}(b,u) = \lim_{N \to \infty} \hat{\mathcal{G}}_N(b,u) \tag{7}$$

As before, we will assume the following:

- For a fixed b that $\mathcal{G}(b,\cdot) \in C^2$
- Moreover, $0 < \mathcal{G}_b''(u) \leq \frac{1}{K_b}$
- For a fixed u that $\mathcal{G}(\cdot,u)$ is monotonic decreasing.
- Sampling and evaluating $g\left(\xi_n(b,u)\right)$ is allowed.
- For any choice of $b \in \mathbb{N}$ and $u \in I$, $\{\xi_n(b,u)\}$ and g satisfy conditions under which a central limit theorem holds for $\hat{\mathcal{G}}_N(b,u)$; an exposition of such conditions can be found in [3].

B. Golden Section

Once again, first we will produce a procedure to find the minimum of $\mathcal{G}(b,u)$ for a fixed b (again called $\mathcal{G}_b(u)$; for the remainder of the section the subscript b will be dropped in contexts where b is fixed). The golden section algorithm works by testing the statement "$\mathcal{G}(u_1) < \mathcal{G}(u_2)$"; now that $\mathcal{G}$ can only be estimated, this test will have to be modified. Under the assumption that a central limit theorem for $\hat{\mathcal{G}}_N(b,u)$ applies for any $b \in \mathbb{N}$ and $u \in I$, we have that

$$\sqrt{N}\left(\hat{\mathcal{G}}_N(b,u) - \mathcal{G}(b,u)\right) \xrightarrow{d} N(0, \sigma^2_{(b,u)}) \tag{8}$$

This allows for the construction of confidence intervals to estimate $\mathcal{G}(u)$), and this forms the basis of our approach. In this context, additional parameters are needed: an *initial sample size* n_0, a *significance level* $1-\alpha$, and an *indifference level* ϵ. For a chosen significance level, let $c = \Phi^{-1}(1 - \frac{\alpha}{2})$:

Require: $a < u^* < b$
  while $b - a > \delta$ **do**
    $u_1 = a + (b-a)(1-\varphi)$
    $u_2 = a + (b-a)\varphi$
    $\hat{\mathcal{G}}_{n_1}(u_1)$ and $\hat{\mathcal{G}}_{n_2}(u_2)$ are constructed with at least n_0 samples each; it may be the case that either n_1 or n_2 is greater than n_0 due to the golden section search reusing points. Let S_1 and S_2 refer to the estimated variance at u_1 and u_2, respectively:
    while $\left|\hat{\mathcal{G}}_{n_1}(u_1) - \hat{\mathcal{G}}_{n_2}(u_2)\right| < c\left(\frac{S_1}{\sqrt{n_1}} + \frac{S_2}{\sqrt{n_2}}\right)$ **do**
      if $S_1\sqrt{(n_2+1)n_2}\left(\sqrt{n_1+1} - \sqrt{n_1}\right) \geq S_2\sqrt{(n_1+1)n_1}\left(\sqrt{n_2+1} - \sqrt{n_2}\right)$ **then**
        Sample at u_1, update $n_1, \hat{\mathcal{G}}_{n_1}(u_1), S_1$.
      else
        Sample at u_2, update $n_2, \hat{\mathcal{G}}_{n_2}(u_2), S_2$.
      end if
      if $\max\left(c\frac{S_1}{\sqrt{n_1}}, c\frac{S_2}{\sqrt{n_2}}\right) < \epsilon$ **then**
        indifference level reached- break loop.
      end if
    end while
    if indifference level reached **then**
      either set $a = u_1$ or $b = u_2$
    else if $\hat{\mathcal{G}}_{n_1}(u_1) < \hat{\mathcal{G}}_{n_2}(u_2)$ **then**
      $b = u_2$
    else
      $a = u_1$
    end if
  end while

Set up in this way, the choice of $a = u_1$ or $b = u_2$ is determined by means of confidence intervals estimating $\mathcal{G}(u_1)$ and $\mathcal{G}(u_2)$. While the confidence intervals overlap each other, sampling is done in such a way that the largest confidence interval is shrunken. If both confidence intervals are sufficiently small, then the means are sufficiently indistinguishable that either $a = u_1$ or $b = u_2$ is a viable choice; this corresponds in the deterministic case with $\mathcal{G}(u_1) = \mathcal{G}(u_2)$. If the confidence intervals no longer overlap, then the choice of $a = u_1$ or $b = u_2$ is made as in the deterministic case. Lastly, in the constrained optimization problem the additional escape condition to the loop is if a confidence interval is ever found to lie entirely below the constraint.

C. Binary Search

Just as in the deterministic case the procedure above enables the evaluation of the function $f(b) = \min_{u \in I} \mathcal{G}(b,u)$. In this case the caveat is that $f(b)$ has a probability of being within some indifference level of $\mathcal{G}(b,u)$. As in the deterministic case, a binary search procedure can be employed on the domain of b by first selecting a finite range of values to search over. The algorithm itself requires no modification, but now the results carry a probability of correct selection.

IV. Analysis

A. Golden Section

The same argument as in the deterministic case can be made with δ in order to bound the (now estimated) value of $\mathcal{G}(u^*)$: thus $n(K_b, \epsilon_b, L_b) + 1$ is the number of points that are estimated by confidence intervals. In each case the probability of the confidence interval containing the value it is estimating is $1-\alpha$; so the lower bound on the probability of correct selection is

$$(1-\alpha)^{n(K_b,\epsilon_b,L_b)+1} \tag{9}$$

In the worst-case scenario, the true minimum is located at one end of the interval I. In this situation, the following recursive formula describes the bound on the error at stage $k+1$:

$$\mathbb{E}[E_{k+1}] = \begin{cases} \mathbb{E}[E_k] & \text{correct selection} \\ \mathbb{E}[E_k] + L(1-\varphi)\varphi^k & \text{incorrect selection} \end{cases}$$

With the probability of correct selection at each stage being $1-\alpha$ this means that the expected error can be expressed recursively:

$$\begin{aligned} \mathbb{E}\left[E_{k+1}\right] &= (1-\alpha)\mathbb{E}\left[E_k\right] + \alpha\left(\mathbb{E}\left[E_k\right] + L(1-\varphi)\varphi^k\right) \\ \mathbb{E}\left[E_{k+1}\right] &= \mathbb{E}\left[E_k\right] + \alpha L(1-\varphi)\varphi^k \end{aligned} \tag{10}$$

Solving the recurrence relation yields

$$\mathbb{E}[E_k] = \alpha L\left(1-\varphi^k\right) \tag{11}$$

Hence, with the estimate from before the upper bound on expected error is:

$$\alpha L\left(1-\varphi^{n(K_b,\epsilon_b,L_b)+1}\right) \tag{12}$$

B. Binary Search

Recall that the structure of our algorithm is to operate over fixed b, changing b only after determining if it is a candidate solution or not. Suppose each determination carries a probability of correct selection $(1-\beta_k)$ such as eqn. (9); thus, if the domain of b is of finite size B then the same logic from the deterministic case implies that the probability of correct selection is bounded below by $\prod_{k=1}^{\lceil \log_2 B \rceil}(1-\beta_k)$.

In the analysis of error in b the same idea from earlier can be applied. Suppose the search domain of b has size B: at iteration k of the binary search algorithm, we have:

$$\mathbb{E}[E_{k+1}] = \begin{cases} \mathbb{E}[E_k] & \text{correct selection} \\ \mathbb{E}[E_k] + \dfrac{B}{2^{k+1}} & \text{incorrect selection} \end{cases}$$

If we assume the probability of correct selection at any iteration is $(1-\beta)$ then the expected error is described recursively:

$$\begin{aligned} \mathbb{E}[E_{k+1}] &= (1-\beta)\mathbb{E}[E_k] + \beta\left(\mathbb{E}[E_k] + \frac{B}{2^{k+1}}\right) \\ \mathbb{E}[E_{k+1}] &= \mathbb{E}[E_k] + \frac{\beta B}{2^{k+1}} \end{aligned} \tag{13}$$

Solving this recurrence relation yields

$$\mathbb{E}[E_k] = B\beta\frac{\left(2^k-1\right)}{2^k} \tag{14}$$

In the worst-case, we already have that $\lceil \log_2 B \rceil$ is the highest number of iterations it would take; hence the upper bound on expected error is:

$$B\beta\frac{\left(2^{\lceil \log_2 B \rceil}-1\right)}{2^{\lceil \log_2 B \rceil}} \tag{15}$$

V. Experimental Results

The performance of a non-gradient method is compared with one using gradients in a simplified problem setting. More complex problems require specialized results justifying the existence of gradient estimators; an example of this kind of result can be found in [1]. Since b is a variable over a discrete domain the comparison will be made between methods for evaluating $f(b)$ for a fixed value of b. The simplified problem is thus one of finding a minimum over a finite domain. Let $X(u)$ be an exponentially distributed random variable with mean u. This is the test function used:

$$\mathcal{G}(u) = \mathbb{E}\left[\log\left(X(u)+1\right) + \frac{4}{2X(u)+1}\right] \tag{16}$$

The search interval will be $[.1, 1]$. The IPA derivative estimator is given by exchanging derivative and expectation, and employing a sample path derivative for $X(u)$:

$$\frac{\partial}{\partial u}\mathcal{G}(u) = \mathbb{E}\left[\frac{X(u)}{u}\left(\frac{1}{X(u)+1} - \frac{8}{(2X(u)+1)^2}\right)\right]$$

After 1000 simulation runs, the results are as follows:

Method	CPU	Samples	PCS	MSE
GS	41.265	3945.23	.7661	.6624
IPA	7.654	16017.2	.7113	.2967

VI. Conclusions

We are currently finishing simulation experiments for comparison purposes and we expect to have fully completed results in a months time.

In a more general multidimensional problem setting gradient estimates can become very expensive to compute. For problem settings where u is a multidimensional quantity we propose that the golden search method can be extended fruitfully by either a coordinate descent approach as documented in [4] or, more generally, a randomized line search.

The estimation of a function at separate locations may be done asynchronously; that is, rather than test and determine which location should be sampled next, both locations are sampled in parallel. An asynchronous stochastic coordinate descent algorithm using gradients has already been documented in [5]; we propose a similar approach would work here.

Another extension to the methods presented would be implementing some degree of error detection and correction. For instance, a variation on the TCP additive-increase/multiplicative-decrease control algorithm could be adapted to the binary search procedure: when it becomes clear that an erroneous selection has occurred during the search, the algorithm increments or decrements its control parameter additively until it is back on track.

References

[1] F. Vázquez-Abad, "Ghost simulation model for discrete event systems, an application to a local bus service," in *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, ser. WSC '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 655–666. [Online]. Available: http://dl.acm.org/citation.cfm?id=2675983.2676069

[2] J. Kiefer, "Sequential minimax search for a maximum," *Proc. Amer. Math. Soc.*, vol. 4, pp. 502–506, 1953.

[3] G. L. Jones, "On the markov chain central limit theorem," *Probab. Surveys*, vol. 1, pp. 299–320, 2004. [Online]. Available: http://dx.doi.org/10.1214/154957804100000051

[4] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10107-015-0892-3

[5] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *Journal of Machine Learning Research*, vol. 16, pp. 285–322, 2015. [Online]. Available: http://jmlr.org/papers/v16/liu15a.html