Sarafoglou, N. (1998). The most influential DEA publications: A comment on Seiford. *Journal of Productivity Analysis, 9*(3), 279–281.
Sarafoglou, N., & Haynes, K. E. (1996). University productivity in Sweden: A demonstration and explanatory analysis for economics and business programs. *The Annals of Regional Science, 30*, 285–304.
Schroeder, R. G. (1973). A survey of management science in university operations. *Management Science, 19*, 895–906.
Seiford, L. M. (1997). A bibliography for data envelopment analysis (1978–1996). *Annals of Operations Research, 73*(1), 393–438.
Sinuany-Stern, Z., Mehrez, A., & Barboy, A. (1994). Academic departments efficiency via DEA. *Computers and Operations Research, 21*, 543–556.
Stallaert, J. (1997). Automated timetabling improves course scheduling at UCLA. *Interfaces, 27*(4), 67–81.
Stevens, C. P. (2003). Enterprise resource planning: A trio of resources. *Information Systems Management, 20*(3), 61–67.
Tomkins, C., & Green, R. (1988). An experiment in the use of data envelopment analysis for evaluating the efficiency of UK university departments of Accounting. *Financial Accountability Management, 4*(2), 147–164.
Turban, E., Aronson, J. E., & Liang, T. P. (2005). *Decision support systems and intelligent systems*. Upper Saddle River, NJ: Prentice-Hall.
Welsh, J. F., & Metcalf, J. (2003). Faculty and administrative support for institutional effectiveness activities: A bridge across the chasm? *Journal of Higher Education, 74*(4), 445–468.
White, G. P. (1987). A survey of recent management science applications in higher education. *Interfaces, 17*(2), 97–108.
Wilson, J. A. (Ed.). (1981). *Management science applications to academic administration* (New directions for higher education, Vol. 35). San Francisco: Jossey-Bass.
Yancey, B. D. (Ed.). (1988). *Applying statistics in institutional research* (New directions for institutional research, Vol. 58). San Francisco: Jossey-Bass.

## Hirsch Conjecture

The Hirsch conjecture has a long history in linear programming. For a bounded ($m \times n$) linear-programming problem, the conjecture concerns how many simplex iterations (basis changes) are necessary in going from one extreme point to another. In a 1957 verbal communication with George B. Dantzig, Warren M. Hirsch (a probabilist from New York University who had worked earlier with Dantzig in the Pentagon) asked: “Does there exist a sequence of $m$ or less pivot operations, each generating a new basic feasible solution, which starts with some given basic feasible solution and ends with some other given basic feasible solution, where $m$ is the number of equations?” (Dantzig 1963, p. 160; Dantzig and Thapa 2003, pp. 25, 31, 33, 34). Over the years, there have been many attempts to prove or disprove the Hirsch conjecture; all of them were eventually shown to be false until Francisco Santos, University of Cantabria, Spain, announced and published his paper, “On a counterexample to the Hirsch conjecture, ”(Santos 2010; also see De Loera 2011; Ziegler 2011).

In geometric terms, the Hirsch conjecture states that if a polytope (bounded polyhedron) is defined by $n$ linear inequalities in $d$ variables, then the length of the longest shortest path among all possible pairs of vertices (its diameter) should be at most $(n - d)$. That is, any two vertices of the polytope may be connected to each other by a path of at most $(n - d)$ edges (Santos 2010). Santos showed that the conjecture was false by constructing a 43-dimensional polytope with 86 facets and a diameter greater than 43.

### References

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton, NJ: Princeton University Press.
Dantzig, G. B., & Thapa, M. N. (2003). *Linear programming 2: Theory and extensions*. New York: Springer.
De Loera, J. (2011). New insights into the complexity and geometry of linear optimization. *Optima, 87*(November), 1–13.
Santos, F. (2010). On a counterexample to the Hirsch conjecture. *La Gaceta de la Real Sociedad Matemàtica Española, 13*(3), 525–538.
Ziegler, G. (2011). Comments on new insights into the complexity and geometry of linear optimization. *Optima, 87*(November), 13–14.

## Hit-and-Run Methods

Zelda B. Zabinsky[1] and Robert L. Smith[2]
[1]University of Washington, Seattle, WA, USA
[2]University of Michigan, Ann Arbor, MI, USA

### Introduction

Hit-and-run is a Markov chain Monte Carlo (MCMC) sampling technique that iteratively generates a sequence of points in a set by taking steps of random length in randomly generated directions. Hit-and-run can be applied to virtually any bounded region in $\Re^n$, and has nice convergence properties. Hit-and-run can generate a sequence of points that asymptotically approach a uniform distribution on open sets, and modifications of hit-and-run can approximate arbitrary multivariate distributions, including the Boltzmann distribution. The versatility of hit-and-run to approximate an arbitrary distribution makes it useful in a number of settings, including global optimization, identification of redundant constraints, simulation, volume estimation and integration estimation. In addition to converging to a target distribution, a good MCMC sampler will converge quickly from an arbitrary starting point, also known as rapid mixing. The mixing time of the original version of hit-and-run is polynomially bounded for convex sets, as opposed to an exponential mixing time for a ball walk.

Hit-and-run was introduced by Smith (1984) as a way to approximate uniformly distributed points in an open set, but many other uses emerged. Diverse applications of hit-and-run include: identifying non-redundant constraints in linear programs (Berbee et al. 1987); evaluation of multidimensional integrals (Chen and Schmeiser 1996); volume estimation of convex sets (Kannan et al. 1997); statistical model validation; construction of a confidence interval for Bayesian inference; discrete-event simulation (Rubinstein and Kroese 2008), and global optimization (Bertsimas and Vempala 2004; Kalai and Vempala 2006; Mete et al. 2011; Romeijn and Smith 1994; Shen et al. 2007; Zabinsky 2003; Zabinsky et al. 1992, 1993).

Hit-and-run in its simplest form is discussed next, followed by its convergence to a uniform distribution and its mixing time. Then a generalized form of hit-and-run that converges to an arbitrary target distribution is discussed, followed by specific variations and implementation considerations. Next, forms of hit-and-run that operate on discrete or mixed continuous/integer sets are discussed, as the previous algorithms assume that the set to be sampled from is continuous. The final section describes simulated annealing-type algorithms for global optimization that embed hit-and-run as a part of their sampling method.

## Definition of Hit-and-Run

Hit-and-run, in its simplest form for a bounded open set S in $\Re^n$, makes a one-step transition from a point $x \in S \subset \Re^n$ to another point $y \in S$ by generating a direction vector uniformly distributed on the surface of a unit hypersphere centered around $x$, and then generating a point $y$ uniformly distributed on the union of the line segments created by the intersection of a line along the direction vector and $S$. This line sampling is typically accomplished by employing a one-dimensional rejection method on the line segment intersected by an enclosing box for $S$.

Hit-and-run generates a sequence of points $\{X_k, k = 0, 1, \ldots\}$ in a bounded open set $S \subseteq \Re^n$ as follows.

**Algorithm 1 (Hit-and-Run)**

```
Step 0  Initialize $X_0 \in S$ and set $k = 0$
Step 1  Generate a random direction $D_k$ uniformly distributed over the surface of a unit hypersphere centered around $X_k$.
Step 2  Generate a random point $X_{k+1} = X_k + \lambda D_k$ uniformly distributed over the line set
            $L_k = \{x : x \in S \text{ and } x = X_k + \lambda D_k,$
                    $\lambda \text{ a real-valued scalar}\}.$
        If $L_k = \emptyset$, go to Step 1.
Step 3  If a stopping criterion is met, stop. Otherwise increment $k$ and return to Step 1.
```

The hit-and-run chain has two distinguishing characteristics: (i) it is globally reaching, i.e., it can move from any point $x \in S \subset \Re^n$ to a neighborhood of any other point $y \in S$ in one step, and (ii) it can be implemented easily even when the feasible set $S$ is defined by membership oracles. As Andersen and Diaconis (2007) describe, the algorithm "hits a point on the sphere and runs in that direction."

Smith (1984) proved that hit-and-run converges in total variation to a uniform distribution. Of course, a direct way to sample a point uniformly from $S$ is to enclose it in a box and sample uniformly from the box until a point lands in $S$. Then this point is exactly uniformly distributed. However, the expected number of points sampled until one lands in $S$ is exponential in dimension, so this is an impractical method for a large-dimensional set. Thus, Markov chain samplers become attractive as a means to approximately sample from a uniform distribution in much less time. Of the MCMC samplers, hit-and-run converges in polynomial time, and is considered to be the most efficient algorithm known to date for generating an asymptotically uniform point in a convex set (Lovász 1999; Lovász and Vempala 2006).

The analysis of the mixing time of hit-and-run on a convex body in $\Re^n$ by Lovász (1999) assumed that the initial distribution of the Markov chain was not far from uniform, i.e., a 'warm-start'. This assumption was later relaxed, preserving hit-and-run's polynomial efficiency, making it the only known random walk that converges efficiently to a uniform distribution starting from any point inside a convex body (Lovász and Vempala 2006). In contrast, the ball walk takes an exponential time to get out of a corner. Moreover, hit-and-run was also shown to be polynomially efficient for sampling from log-concave distributions over convex bodies.

Some insight into hit-and-run's efficiency is presented by Ghate and Smith (2009), who showed that the network of points and arcs generated by hit-and-run is a small world network in which most nodes are not neighbors of one another, but most nodes can be reached from every other in a small number of steps. Thus another interpretation of hit-and-run is that it generates a small world on the fly.

Given hit-and-run's success at efficiently approximating a uniform distribution, many variations and generalizations have been developed.

## Generalizations of Hit-and-Run

The most celebrated Markov chain sampler, introduced by Metropolis et al. (1953), used the idea of an acceptance-rejection step to act as a filter and bias the chain towards a Boltzmann distribution. The original hit-and-run algorithm was extended by Romeijn and Smith (1994) to converge to a target distribution $\pi$ by adding an appropriate filter, and later further extended using a conditionalization on $\pi$ to the one-dimensional line segment (Bélisle et al. 1993). Thus, hit-and-run converges to an arbitrary target distribution $\pi$ in total variation.

Andersen and Diaconis (2007) proposed a generalization of hit-and-run algorithms for MCMC samplers and related it to the Gibbs sampler, Swendsen-Wang block spin dynamics, data augmentation, auxiliary variables, slice sampling, and the Burnside process under a unifying scheme. They describe choosing the point $X_{k+1}$ according to the density $\pi$ restricted to the line determined by the direction vector, as in Bélisle et al. (1993). The choice of the uniform distribution for direction is replaced by a general choice, and even the concept of a one-dimensional Euclidean line determined by the direction vector is generalized to include subsets of $S$.

The following algorithm generalizes hit-and-run with a general direction distribution and a Metropolis filter that converges to an arbitrary target distribution $\pi$ on $S$, where $\nu$ is an absolutely continuous probability distribution defined on the surface of an $n$-dimensional unit sphere, with density bounded away from zero.

**Algorithm 2 (Hit-and-Run for Target Distribution $\pi$)**

```
Step 0  Initialize $X_0 \in S$ and set $k = 0$.
Step 1  Generate a random direction $D_k$ from the direction distribution $\nu$ on the surface of a unit hypersphere centered around $X_k$.
Step 2  Generate a candidate point $Z = X_k + \lambda D_k$ uniformly distributed over the line set
        $L_k = \{x : x \in S \text{ and } x = X_k + \lambda D_k, \lambda \text{ a real-valued scalar}\}$
        If $L_k = \emptyset$, go to Step 1.
Step 3  Accept or reject the candidate point $Z$ with a Metropolis filter for the target distribution $\pi$,
        $$X_{k+1} = \begin{cases} Z & \text{w.p. } \min\{1, \pi(Z)/\pi(X_k)\} \\ X_k & \text{otherwise.} \end{cases}$$
Step 4  If a stopping criterion is met, stop. Otherwise increment $k$ and return to Step 1.
```

Note that if $\pi$ is a uniform distribution, then all candidate points are accepted and Algorithm 1 is a special case of Algorithm 2.

Specific variations and implementations of hit-and-run are discussed next.

## Variations and Implementations of Hit-and-Run

Several variations with specific direction distributions and candidate point sampling methods have been studied in the literature.

The most common direction distribution, and one that is readily implemented, is the uniform distribution on the surface of an $n$-dimensional hypersphere, termed hyperspherical direction (HD) in

Berbee et al. (1987); Zabinsky et al. (1992). It is easily implemented by generating $n$ independent values $d_i, i = 1, 2, \ldots, n$ from a standard normal distribution, $N(0, 1)$ and scaling them to determine the unit direction vector $D$:

$$D = (d_1, d_2, \ldots, d_n)\left(\sum_{i=1}^{n} {d_i}^2\right)^{-1/2}. \qquad (1)$$

Another natural choice for direction distribution, termed coordinate direction (CD), is a uniform distribution over the $n$ coordinate vectors (spanning $\Re^n$). Both HD and CD versions of direction choice were presented and applied to identifying nonredundant linear constraints in Berbee et al. (1987).

While hit-and-run is guaranteed to converge for a wide class of target distributions $\pi$ when using the HD choice, the same is not true when using the CD choice. It is possible to construct situations where CD will not converge to $\pi$. For example, CD will fail to converge to a uniform distribution on disconnected regions with the property that some points cannot be reached from others along a sequence of coordinate direction moves.

Another modification of the direction choice, introduced by Romeijn et al. (1999), is called a reflection generator. The reflection generator was motivated by the problem of stalling, which may occur if the line intersects a small portion of the feasible set. For example, when the current point $x$ is near a corner of a hypercube, there is a high probability that the next sample point is very close to $x$, and a very low probability that the next point generated is a substantial distance from $x$, especially when the number of dimensions is large. This problem is similar to jamming, a well-known problem in nonlinear programming. The reflection generator essentially lengthens the line associated with a chosen direction by reflecting it off the boundaries of the feasible region into the interior. This increases the probability of sampling a point far away from the current point. A general reflection generator is defined in Romeijn et al. (1999), with a straightforward component-by-component reflection implementation. Convergence results are preserved, and positive numerical experience was reported.

Kaufman and Smith (1998) exploited the robustness in direction distribution to accelerate the rate of convergence of hit-and-run. They derived a unique non-uniform direction distribution that optimizes the rate of convergence of hit-and-run to a uniform distribution on a convex set. They used sampled points to fit an ellipsoid to the convex set, and used the parameters of the ellipsoid as bootstrap parameters in the direction distribution to approximate the optimal direction distribution; calling the Markov chain artificial centering hit-and-run.

In addition to variations on choosing the direction distribution in Step 1, there are variations on choosing the random candidate point on the line in Step 2. Theoretically, the point could be chosen according to the target distribution $\pi$ restricted to the line. However, in practice, this may be computationally difficult to implement. The line sampling is often referred to as step-size distribution. In hit-and-run as stated in Algorithm 1 and Algorithm 2, the step size $\lambda$ is uniformly distributed on the intersection of the random bidirection with the feasible region. Other variations include a fixed step-size or a variable length interval that can shrink or expand.

A parametrized step-size distribution is used in Ghate and Smith (2009) for solving the Small World problem. The probability density function for the step-size $\lambda$ is parametrized by $a$, and is roughly proportional to $(1/|\lambda|^a)$. When $a = 0$, the distribution is the familiar uniform sampling distribution. Ghate and Smith (2009) showed that the expected hitting time for the Small World problem is minimized when the parameter $a = 1$ for the step-size distribution, and that $a = 1$ is the unique choice of $a$ that is scale invariant among all nonnegative values. This parameterized step-size distribution was further explored with hit-and-run in the context of global optimization.

Another consideration in implementing Step 2 is the difficulty in identifying the intersection of the line determined by the random direction, and the feasible set $S$, even when $S$ is convex. Step 2 can be straightforward to implement if it is possible to determine the points of intersection on the line, i.e., find $\lambda_{min}$ and $\lambda_{max}$ such that $X_k + \lambda D_k \in S$ for $\lambda_{min} \le \lambda \le \lambda_{max}$. When $S$ is defined by linear inequalities, or analytically invertible functions, the intersection points can be easily expressed (Zabinsky 2003). Then $\lambda$ can be chosen uniformly over that interval, or according to the conditionalization of $\pi$, thus producing the random candidate point.

However, if the feasible region $S$ is nonconvex, and/or the intersection points are not easily determined, then a common implementation is to enclose the feasible set $S$ in a box $B$, or any regular shape that is easy to determine intersection points, and use a one-dimensional acceptance-rejection scheme to produce the random candidate point.

The following algorithm on an enclosing box $B$ is a modification of hit-and-run with a general direction distribution, as in Bélisle et al. (1993), with details of the one-dimensional acceptance-rejection sampling, as provided in Kiatsupaibul et al. (2011).

**Algorithm 3. (Hit-and-Run on a Box)**

**Step 1** Generate a random direction $D_k$ with direction distribution $\nu$ and set $i = 1$.

**Step 2** Generate $\lambda_{k,i}$ from the step-size (typically uniform) distribution on

$$R_k = \{r \in \Re : X_k + rD_k \in B\}.$$

**Step 3** If $X_k + \lambda_{k,i}D_k$ is not in $S$, set $i = i + 1$ and return to Step 2. Otherwise, set $Z = X_k + \lambda_{k,i}D_k$.

**Step 4** Accept or reject the candidate point $Z$ with a Metropolis filter for the target distribution $\pi$,

$$X_{k+1}\begin{cases} Z & \mathrm{w.p}\min\{1,\pi(\mathrm{Z})/\pi(\mathrm{X_k})\} \\ X_k & \text{otherwise} \end{cases}$$

**Step 5** If a stopping criterion is met, stop. Otherwise increment $k$ and return to Step 1.

The additional computation due to the one-dimensional acceptance-rejection has been analyzed by Kiatsupaibul et al. (2011) for the case when $\pi$ is a uniform distribution. They show that the size of the box is not a critical factor to the overall computational effort. More precisely, bounds on the expected mixing time of hit-and-run on a box including all sample points increases only by a linear function of the box diameter (i.e., longest chord in the box).

Another variation to speed up the convergence rate and reduce the number of rejected sample points is to incorporate the shrinking algorithm, also known as a slice sampler, into hit-and-run. The idea is to shrink the interval for selecting $\lambda$, as follows.

**Algorithm 4. (Hit-and-Run on a Box with Shrinking Step-Size)**

**Step 0** Initialize $X_0 \in S$ and set $k = 0$.

**Step 1** Generate a random direction $D_k$ with direction distribution $\nu$, defining the step-size set as

$$R_k = \{r \in \Re : X_k + rD_k \in B\}.$$

Set $l_1^+ = max_r R_k$ and $l_1^- = min_r R_k$, and set $i = 1$.

**Step 2** Generate $\lambda_{k,i}$ from the uniform distribution on the open interval $(l_i^-, l_i^+)$.

**Step 3** If $X_k + \lambda_{k,i}D_k$ is not in $S$, set $l_{i+1}^+$ and $l_{i+1}^-$ as follows:
if $\lambda_{k,i} > 0$, set $l_{i+1}^+ = \lambda_{k,i}$ and keep $l_{i+1}^- = l_i^-$;
if $\lambda_{k,i} < 0$, set $l_{i+1}^- = \lambda_{k,i}$ and keep $l_{i+1}^+ = l_i^+$.
Then, set $i = i + 1$ and return to Step 2.
Otherwise, if $X_k + \lambda_{k,i}D_k$ is in $S$, set $Z = X_k + \lambda_{k,i}D_k$.

**Step 4** Accept or reject the candidate point $Z$ with a Metropolis filter for the target distribution $\pi$,

$$X_{k+1} = \begin{cases} Z & \text{w.p. } \min\{1, \pi(\mathrm{Z})/\pi(\mathrm{X_k})\} \\ X_k & \text{otherwise.} \end{cases}$$

**Step 5** If a stopping criterion is met, stop. Otherwise increment $k$ and return to Step 1.

Algorithm 4 differs from Algorithm 3 in that the step-size interval is shrinking. This shrinkage increases the probability of acceptance in Steps 2 and 3. Because every open subset $S$ can still be reached in one step, the convergence property of the new Markov chain remains the same.

When $S$ is convex, the iteration point process generated by Algorithm 4 is the same as that generated by Algorithm 3, so the mixing rate of the two processes is the same. However, when $S$ is not convex, the iteration point processes from the two algorithms distribute differently, and, hence, the mixing rates may be different. Computational results in Kiatsupaibul et al. (2011) suggest that Algorithm 4 is faster than Algorithm 3 when $S$ is not convex.

Other computational results are given in Chen and Schmeiser (1996), where empirical comparisons are made between variations of hit-and-run and other sampling methods including the Gibbs sampler.

## Hit-and-run for Discrete and Mixed Continuous/Integer Sets

Given the exceptional performance of hit-and-run on continuous sets in $\Re^n$, it is natural to wonder if it can be extended to discrete sets, or mixed sets in $\Re^n \times Z^m$. The generalized line set in Andersen and Diaconis (2007) with conditions for convergence allows a wide variety of versions that converge to a target distribution. Baumert et al. (2009) provide a detailed definition of discrete hit-and-run (DHR) with mixing times for some specific classes of problems. Mete et al. (2011) introduces a variation on DHR, called pattern hit-and-run (PHR) that is efficiently implemented on both discrete and mixed continuous/integer sets. Both DHR and PHR, summarized next, maintain many of the nice convergence properties of hit-and-run, including polynomial mixing time for some classes of sets.

DHR defines its line set using a bidirectional random walk, called a biwalk. Whereas classical Markov chains such as the nearest neighbor random walk or the coordinate direction random walk fail to converge to a target distribution $\pi$ on general discrete sets, because they can get trapped in isolated regions of the support set, DHR converges because it retains the global reaching property of hit-and-run.

Consider a finite set $S$ with a membership oracle that is a subset of B given by a a bounded hyper-rectangle intersected with the $n$ dimensional integer lattice $\mathbb{Z}^n$. The third step applies a Metropolis filter with respect to the target distribution to accept or $\pi$ reject the candidate point and complete the transition of DHR. The DHR algorithm follows.

## Discrete Hit-and-Run (DHR)

**Step 0** Initialize $X_0 \in S$ and set $k = 0$.

**Step 1** Generate a biwalk by generating two independent, nearest neighbor random walks in $B$ that start at $X_k$ and end before they step out of $B$. The biwalk may have loops but has finite length with probability one. The sequence of points visited by the biwalk is stored in an ordered list.

**Step 2** Generate a candidate point $Z$ by choosing a point uniformly distributed from the intersection of the list and $S$. Note the intersection always contains at least one point, the current point $X_k$.

**Step 3** Accept or reject the candidate point $Z$ with a Metropolis filter for the target distribution $\pi$,

$$X_{k+1} = \begin{cases} Z & \text{w.p. } \min\{1, \pi(\mathrm{Z})/\pi(\mathrm{X_k})\} \\ X_k & \text{otherwise.} \end{cases}$$

**Step 4** If a stopping criterion is met, stop. Otherwise increment $k$ and return to Step 1.

The reason for employing two independent nearest neighbor walks to define the line set in Step 1 instead of one walk, and for working with the ordered sequence of points in Step 2 as opposed to the set of distinct points visited, is to ensure symmetry of the candidate generator Markov chain. It is easy to construct examples where symmetry fails by employing a single nearest neighbor random walk and/or use the set of distinct points visited (Baumert et al. 2009). The Markov chain of DHR is globally reaching. The global reaching property together with symmetry and other characteristics imply that DHR converges to the target distribution $\pi$ as desired.

An upper bound on the mixing time of DHR to a uniform distribution is given in Baumert et al. (2009), and polynomial upper bounds for four examples are given. The four examples include: a box within a box, a wedge inside a cube, multiple cubes inside a cube, and isolated yet aligned points within a cube. Note that conventional random walks such as the nearest neighbor random walk and the coordinate direction random walk also mix in polynomial time on the first two examples; however, both of these walks get stuck in isolated regions of $S$ in the third and fourth examples and fail to converge to a uniform distribution. A fifth example given in Baumert et al. (2009) of diagonal points inside a cube only yields exponential bounds for the mixing time, although convergence is still maintained.

The success of discrete hit-and-run with random biwalks inspired the development of pattern hit-and-run for mixed continuous/integer domains. The biwalk in DHR is computationally expensive to implement because each move in the biwalk requires a randomization, and the list associated with the biwalk must be stored to perform the acceptance-rejection step. A more efficient implementation was introduced in Mete et al. (2011), where the biwalk is defined with the use of patterns, visualized as a repetition of $n$ step-sizes. An advantage to the use of a pattern-generated biwalk is that the pattern is easily generated with only $n$ random number generations, and the acceptance-rejection on the biwalk can be performed by generating a single random number and analytically mapping it to a point on the biwalk. In Mete et al. (2011), two methods for generating patterns are defined; a sphere biwalk and a box biwalk. PHR with either sphere or box biwalk preserves the convergence properties of hit-and-run to a target distribution, and PHR with sphere biwalk converges to continuous hit-and-run as the mesh of the discretized points becomes finer, approaching a continuum. PHR with box biwalk converges to a variation of hit-and-run where the direction distribution is uniform on the surface of a box, instead of the common surface of a hypersphere.

When the feasible set $S$ is ill-structured, the acceptance-rejection on the intersection of the biwalk and $S$ is inevitable. However, a well-structured set of interest is an integer or mixed continuous/integer polytope, as often arises in integer programming or mixed integer linear programming feasible sets. Mete and Zabinsky (2012) remove the inefficiency that arises from rejecting infeasible points, and utilize the linearity of the constraints defining the polytope to directly sample from the intersection of the biwalk and the polytope. This provides an efficient variation of pattern hit-and-run that converges to a target distribution on a discrete or mixed continuous/discrete polytope.

Convergence to $\pi$ on a general discrete polytope is not simple to attain. For example, a nearest neighbor random walk will not converge to a uniform distribution on a thin polytope that has isolated points without feasible adjacent neighbors. PHR is able to maintain the global reaching property on any polytope by determining all the feasible points on the biwalk, even though they may not be adjacent. Mete and Zabinsky (2012) derive a method to analytically generate a uniform point on the intersection of a biwalk and a discrete polytope by determining the number of feasible points on the biwalk and mapping a uniform point on $[0, 1]$ to a uniform feasible point on the biwalk. They extend the idea to a mixed continuous/discrete lattice of a polytope.

Moreover, PHR converges to a uniform distribution in polynomial time on a class of discrete polytopes; specifically, discrete polytopes that are defined by a finite number of knapsack constraints i.e., $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ where $a_{ij}$ are nonnegative and $b_i$ are positive for $i = 1, \ldots, m$ and the number of constraints $m$ is independent of the number of dimensions $n$. This polynomial time performance and the convergence of PHR to hit-and-run on continuous sets suggests the potential efficiency for hit-and-run samplers on a broad class of sets.

## Hit-and-Run for Global Optimization

Hit-and-run has been successfully applied to optimization, initially continuous problems, and expanded to mixed continuous/integer problems (Bertsimas and Vempala 2004; Kalai and Vempala 2006; Mete et al. 2011; Romeijn and Smith 1994; Shen et al. 2007; Zabinsky 2003; Zabinsky et al. 1993).

Consider the following global optimization problem:

$$\begin{aligned}&\text{minimize } f(x)\\&\text{subject to } x \in S \subset B.\end{aligned}$$

An initial application of hit-and-run to optimization was called Improving Hit-and-Run (IHR) by Zabinsky et al. (1993), which modifies Step 3 in Algorithm 2 by simply accepting a candidate point only if it has an improving objective function value, as follows:

**Step 3** Complete the transition to $X_{k+1}$ where,

$$X_{k+1} = \begin{cases} Z & \text{if } \mathrm{f(Z) < f(X_k)} \\ X_k & \text{otherwise.} \end{cases}$$

IHR has been successfully applied to realistic problems (Zabinsky et al. 1992, 2006). The complexity of IHR is, on average, of $O(n^{5/2})$ for a certain class of convex programs (Zabinsky et al. 1993). The direction distribution of IHR on elliptical programs, as defined in Zabinsky et al. (1993), is a multivariate normal distribution with mean zero and covariance matrix equal to the Hessian inverse of the objective function, $H^{-1}$. If the covariance matrix is the identity matrix, then the direction distribution is simply HD. Although the Hessian is not typically known, the results indicate the ability to guide the direction distribution for better performance.

Romeijn and Smith (1994) embedded hit-and-run into a simulated annealing algorithm and called it Hide-and-Seek. They added acceptance probabilities according to the Metropolis criterion with a temperature parameter $T_k$ so that Step 3 becomes **Step 3**. Accept or reject the candidate point $Z$ according to a Metropolis filter with temperature $T_k$,

$$X_{k+1} = \begin{cases} Z & \text{w.p. } \min\left\{1, e^{-(f(Z)-f(X_k))/T_k}\right\} \\ X_k & \text{otherwise.} \end{cases}$$

A property of Hide-and-Seek is that it converges to a Boltzmann $T$ distribution, for a fixed temperature (Bélisle et al. 1993). For a general cooling schedule, Romeijn and Smith (1994) showed that if Hide-and-Seek ran long enough at each temperature value to converge to its stationary Boltzmann distribution, then the number of these temperature values would be linear in dimension. This led to an analytically derived adaptive cooling schedule, which was later extended to apply to both continuous and discrete global optimization problems (Shen et al. 2007). The analysis was motivated by the result that a sequence of such Boltzmann distributions achieves a linear complexity on the average number of function evaluations. Hit-and-run embedded as a candidate generator in simulated annealing has both analytical and numerical success.

Even though the acceptance probability for simulated annealing is interpreted as aiding the algorithm to escape local optima, simulated annealing has also been successfully applied to convex programs. Bertsimas and Vempala (2004) and Kalai and Vempala (2006) used hit-and-run as a candidate generator in a simulated annealing-type algorithm for solving convex programs with a membership oracle. In Kalai and Vempala (2006), simulated annealing is shown to converge quickly, and under certain conditions, the Boltzmann distribution is proven to be optimal for annealing on convex problems.

Simulated annealing on finite combinatorial problems has been successful; however, the candidate point generator is specifically chosen for each problem. Pattern hit-and-run, for integer or mixed continuous/integer sets, has been embedded into simulated annealing in Mete et al. (2011) and numerically shown to be very effective on many test problems.

## See

- ▶ Global Optimization
- ▶ Markov Chain Monte Carlo
- ▶ Monte Carlo Simulation
- ▶ Simulation of Stochastic Discrete-Event Systems

## References

Andersen, H. C., & Diaconis, P. (2007). Hit and run as a unifying device. *Journal de la societe francaise de statistique & revue de statistique appliquee, 148*(4), 5–28.

Baumert, S., Ghate, A., Kiatsupaibul, S., Shen, Y., Smith, R. L., & Zabinsky, Z. B. (2009). Discrete hit-and-run for generating multivariate distributions over arbitrary finite subsets of a lattice. *Operations Research, 57*(3), 727–739.

Bélisle, C. J. P., Romeijn, H. E., & Smith, R. L. (1993). Hitand-run algorithms for generating multivariate distributions. *Mathematics of Operations Research, 18*, 255–266.

Berbee, H. C. P., Boender, C. G. E., Rinnooy Kan, A. H. G., Scheffer, C. L., Smith, R. L., & Telgen, J. (1987). Hit-and-run algorithms for the identification of nonredundant linear inequalities. *Mathematical Programming, 37*, 184–207.

Bertsimas, D., & Vempala, S. (2004). Solving convex programs by random walks. *Journal of the ACM, 51*(4), 540–556.

Chen, M. H., & Schmeiser, B. W. (1996). General hit-and run Monte Carlo sampling for evaluating multidimensional integrals. *Operations Research Letters, 19*, 161–169.

Ghate, A., & Smith, R. L. (2009). A hit-and-run approach for generating scale invariant small world networks. *Networks, 53*(1), 67–78.

Kalai, A. T., & Vempala, S. (2006). Simulated annealing for convex optimization. *Mathematics of Operations Research, 31*(2), 253–266.

Kannan, R., Lovász, L., & Simonovits, M. (1997). Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms, 11*, 1–50.

Kaufman, D. E., & Smith, R. L. (1998). Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research, 46*(1), 84–95.

Kiatsupaibul, S., Smith, R. L., & Zabinsky, Z. B. (2011). An analysis of a variation of hit-and-run for uniform sampling from general regions. *ACM Transactions on Modeling and Computer Simulation (ACM TOMACS), 21*(3), 16:1–16:11.

Lovász, L. (1999). Hit-and-run mixes fast. *Mathematical Programming, 86*, 443–461.

Lovász, L., & Vempala, S. (2006). Hit-and-run from a corner. *SIAM Journal on Computing, 35*(4), 985–1005.

Mete, H. O., Shen, Y., Zabinsky, Z. B., Kiatsupaibul, S., & Smith, R. L. (2011). Pattern discrete and mixed hitand-run for global optimization. *Journal of Global Optimization, 50*(4), 597–627.

Mete, H. O., & Zabinsky, Z. B. (2012). Pattern hit-and-run for sampling efficiently on polytopes. *Operations Research Letters, 40*, 6–11.
Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*, 1087–1090.
Romeijn, H. E., & Smith, R. L. (1994). Simulated annealing for constrained global optimization. *Journal of Global Optimization, 5*, 101–126.
Romeijn, H. E., Zabinsky, Z. B., Graesser, D. L., & Neogi, S. (1999). New reflection generator for simulated annealing in mixed-integer/continuous global optimization. *Journal of Optimization: Theory and Applications, 101*(2), 403–427.
Rubinstein, R. Y., & Kroese, D. P. (2008). *Simulation and the Monte Carlo method* (2nd ed.). Hoboken, NJ: Wiley.
Shen, Y., Kiatsupaibul, S., Zabinsky, Z. B., & Smith, R. L. (2007). An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization, 38*, 333–365.
Smith, R. L. (1984). Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research, 32*, 1296–1308.
Zabinsky, Z. B. (2003). *Stochastic adaptive search for global optimization*. Boston: Kluwer.
Zabinsky, Z. B., Graesser, D. L., Tuttle, M. E., & Kim, G. I. (1992). Global optimization of composite laminate using improving hit and run. In C. A. Floudas & P. M. Pardalos (Eds.), *Recent advances in global optimization* (pp. 343–365). Princeton, NJ: Princeton University Press.
Zabinsky, Z. B., Smith, R. L., McDonald, J. F., Romeijn, H. E., & Kaufman, D. E. (1993). Improving hit and run for global optimization. *Journal of Global Optimization, 3*, 171–192.
Zabinsky, Z. B, Tuttle, M. E., Khompatraporn, C. (2006). A case study: Composite structure design optimization. In J. Pinter (Ed.), *Global optimization: Scientific and engineering case studies* (pp. 507–528). New York: Springer-Verlag.

## Homogeneous Lanchester Equations

Simple Lanchester equations with one equation for each side. These equations are used when the weapons for each side are homogeneous in nature (all small-arms) or as a simplified approximation of a heterogeneous situation.

### See

▶ Lanchester's Equations

## Homogeneous Linear Equations

A set of linear equations of the form $\boldsymbol{Ax} = \mathbf{0}$.

## Homogeneous Solution

A solution to the set of equations $\boldsymbol{Ax} = \mathbf{0}$. The solution $\boldsymbol{x} = \mathbf{0}$ is called a trivial solution, while a solution $\boldsymbol{x} \neq \mathbf{0}$ is called a nontrivial solution.

## Horn Clause

A logical expression of the form A $\rightarrow$ C, where A (the antecedent) is a simple conjunction of basic (atomic) propositions and C (the consequent) is either null or is a single atomic proposition.

### See

▶ Artificial Intelligence

## Hospitals

Yasar A. Ozcan
Virginia Commonwealth University, Richmond, VA, USA

### Introduction

Hospitals represented a growing $760 billion industry in the U.S. in 2009 and were responsible at that time for about 32.6% of the nation's health care expenditures. There are 5,815 registered hospitals in U.S., and they have treated 127 million people in emergency departments, admitted 35.1 million for in patient care, and provided 642 million outpatient visits. These hospitals employ 5.4 million professionals or 34.6% of the all health care jobs in U.S. The effect of hospital expenditures on total output in U.S. economy reaches $2.5 trillion (AHA 2011).