Iterative Improvement plus Random Restart Stochastic Search

by

X. Hu, R. Shonkwiler, and M. C. Spruill

School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332
email: shonkwiler@math.gatech.edu, spruill@math.gatech.edu

Keywords:
Monte Carlo Methods, Multistart Methods, Markov Chains, Parallelization.

AMS Subject Classification: 65K10.

Running head: Iterative Improvement plus Random Restart

Mail proofs to:
R. Shonkwiler
School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332

December 18, 1997

Abstract

The multistart or iterative improvement plus random restart (I^2R^2) method for global optimization combines local search with random initialization. This method partitions the solution space into basins of attraction to the local optima. The method's convergence rate to a global optimizer as the number of iterations t increases is given as $\frac{1}{s}\lambda^t$ in terms of two scalar parameters, acceleration s, and retention λ. Acceleration is estimated as $s > 1/\lambda > 1$. The expected running time of the algorithm in order to find a basin containing a global optimizer is estimated as

$$E \approx \frac{1}{s}\frac{1}{1-\lambda}.$$

There is associated with every I^2R^2 search a fundamental polynomial f of low degree having a unique root $\eta > 1$. Acceleration and retention are given in terms of f and η according to

$$\lambda = \frac{1}{\eta} \qquad s = \frac{\eta(\eta-1)f'(\eta)}{-f(1)}.$$

The coefficients of f can be estimated during the course of a run. Under mild conditions on the class of potential objective functions, $\mathcal{F}$, the most effective start/restart distribution μ is the uniform distribution. Parallel execution of an I^2R^2 search using m independent and identical processes (IIP) improves the convergence rate to $(1/s^m)\lambda^{mt}$; since $s > 1$, IIP parallel using m processes converges more than m times faster than a single process. Hence accelerated convergence can even be achieved by simulating m processes *in code* on a single processor. The method and our results are illustrated on two example problems.

1. Introduction

Given a real valued function C defined on a domain Ω, we analyze restart methods for finding some point $x^* \in \text{opt}(C)$ the set of global minimizers of C over Ω,

$$C(x^*) \leq C(x), \qquad x \in \Omega.$$

Referring to C as a *cost*, assume it is any function from a class $\mathcal{F}$ of functions all having domain Ω. In what follows, we may impose other conditions on $\mathcal{F}$ as well.

If C is sufficiently smooth, powerful numerical methods exist for identifying local minimizers, (Dennis and Schnabel, 1983). Often these methods utilize local derivative, or gradient, information to construct a sequence of points $x_0, x_1, x_2, \ldots \to x_\infty$ on which the function values $v_i = C(x_i)$ decrease until a local minimum is determined to within some tolerance. If C has numerous local minima, locating a global minimizer is problematic and success depends on the choice of starting value x_0.

Inspired by this gradient model, we seek to generalize it, even though derivatives may not be defined for functions in $\mathcal{F}$. Given $C \in \mathcal{F}$, throughout this work we assume there is a (deterministic) algorithm g such that (1) for every $x \in \Omega$, $g(x) \in \Omega$ and

$$C(g(x)) \leq C(x), \tag{1}$$

and (2) g is not *cyclic*, that is, if $x = g^k(x)$ for some $k > 0$, then

$$x = g(x) = g^2(x) = \ldots = g^k(x).$$

We refer to g as an *(iterative) improvement* algorithm. In a given application problem, g might be the implementation of some heuristic which improves the cost on each iteration.

Thus, as above, an improvement algorithm applied to its own output generates a sequence of points x_0, $x_1 = g(x_0)$, $x_2 = g^2(x_0)$, ... in Ω on which the cost values are non-increasing. The limit of such a sequence terminates in a local minimizer relative to C (which could also be a global minimizer as well) beyond which improvement is not possible by the application of g alone. All points of the domain which are attracted to a given local minimizer define its basin. That is to say, the relation $x \equiv y$ if and only if $\lim_{k\to\infty} g^k(x) = \lim_{j\to\infty} g^j(y)$ is an equivalence relation on Ω. The resulting equivalence classes, B_i, $i = 0, 1, \ldots$, are the *basins* of Ω relative to g. The *settling point* or *local minimizer* x_∞ of basin B is given as the limit, $\lim_{k\to\infty} g^k(x)$, where x is any point of B.

Once a local minimum has been reached, for instance when $g(x) = x$, the process must be *restart*. For this, we use a fixed probability distribution μ over Ω. Upon reaching a

local minimum, the search is continued by selecting a *restart point* $x_0' \in \Omega$ with probability $\mu(x_0')$. We also use μ to start the search; thus μ is the start/restart distribution. We assume that μ is not degenerate, that is $\mu(x) > 0$ for all $x \in \Omega$.

Our search process is described by a random variable $X(t)$ on Ω. To start, $X(1)$ is selected with probability $\mu(X(1))$. Then for $t = 1, 2, \ldots, t_1$ we take $X(t+1) = g(X(t))$ where t_1 is the least t such that $X(t+1) = X(t)$. The sequence $\{1, \ldots, t_1\}$ is the first *epoch*. Next $X(t_1 + 1)$ is (re-)selected with probability $\mu(X(t_1 + 1))$ and so on. The start/restart times are $1, t_1, t_2, \ldots$, a strictly increasing sequence, and define the epochs. (No restart check is made to previous epochs, i.e. that the re-selected $X(t_i + 1)$ equals $X(t_i)$, $i = 1, 2, \ldots$.)

We also define the *running minimizer* random variable $W(t) \in \Omega$ as follows. Let $v_t^* = \min\{C(X(\tau)) : 1 \le \tau \le t\}$, hence v_t^* is the best observation up to time t, or the *current best observation*. Put $W(t) = X(\tau)$ where $\tau = \min\{s : C(X(s)) = v_t^*\}$. Evidently $C(W(t)) = v_t^*$ for all t; if the current best observation has been encountered multiple times, $W(t)$ is the first point to achieve it.

In what follows we will assume that the domain Ω is a finite set, let N be its cardinality, $N = |\Omega|$. Nevertheless our results also hold for iterative improvement applied to smooth cost functions on domains in Euclidean space provided the number of improvement iterations are kept finite. This is always the case in a computer implementation. The reason is that the results depend only on the number of iterations taken to "find" local minima and not on the points themselves.

We summarize our results here. Iterative improvement plus random restart (I^2R^2), is a homogeneous Markov Chain on Ω. Its convergence rate to a global optimizer is $\frac{1}{s}\lambda^t$, that is

$$\Pr[W(t) \notin \mathrm{opt}(C)] \sim \frac{1}{s}\lambda^t \to 0 \qquad \text{as } t \to \infty,$$

where s and λ are two scalar parameters we call *acceleration* and *retention* respectively. Acceleration is estimated as $s > 1/\lambda > 1$. The expected running time of the algorithm in order to find a basin containing a global minimizer is estimated as

$$E \approx \frac{1}{s}\frac{1}{1-\lambda}.$$

There is associated with every I^2R^2 search a *fundamental polynomial*

$$f(\xi) = r_0\xi + r_1\xi^2 + \ldots + r_d\xi^{d+1} - 1,$$

of low degree having a unique root $\eta > 1$. Acceleration and retention are given in terms of f and η according to

$$\lambda = \frac{1}{\eta} \qquad s = \frac{\eta(\eta-1)f'(\eta)}{-f(1)}.$$

The coefficients of f can be estimated during the course of a run.

Under mild conditions on the class $\mathcal{F}$, the most effective start/restart distribution μ is the uniform distribution over Ω.

Parallel execution of an I^2R^2 search using m independent and identical processes (IIP) improves the convergence rate to $(1/s^m)\lambda^{mt}$; since $s > 1$, IIP parallel using m processes converges more than m times faster than a single process. Hence improvement can be achieved by simulating m processes *in code* on a single processor.

The method and our results are illustrated on two example problems.

A recent survey of the literature on global stochastic optimization can be found in (Schoen, 1990). What we have called random restart methods are included there as the simplest instances of what Schoen calls multistart methods. (Solis and Wets, 1981) study convergence for more complicated random restart methods in which the probability distribution for choosing the next starting point can depend on the evolution of the search. (Boender and Rinnooy Kan, 1987) study stopping rules. Our studies do not address these issues. (Törn and Zelinskas, 1989) present in their survey some numerical results on parallelization for a collection of methods and end by recommending future study of Monte Carlo and geometric rather than algorithmic parallelization. Our investigation of random restarts is in one such recommended area, multiple CPU Monte Carlo parallelization, and continues the work of (Shonkwiler and Van Vleck, 1992) using the same techniques.

2. Running time estimation

When the number of improvement iterations is finite, as we have assumed, I^2R^2 search has a simple graph theoretic representation.

Forest of trees model

Let the vertices of the graph correspond to the points of the domain. And let there be a directed edge from x to y if and only if $y = g(x)$. In this representation, the points of each basin are organized as a rooted tree with the local minimizer serving as the root. By the *depth* of a tree we mean its maximum path length; the number of vertices in such a path is the depth plus 1. The graphical representation of the entire space is therefore a forest of trees, one for each basin. The settling point of at least one basin is a global minimizer. We shall refer to such a basin as a *goal basin*. By the *min-bin* M we mean the union of goal basins.

The incorporation of random restart modifies our graph model by adding an edge from every local minimizer to every point of the domain.

As noted above, the random variable $X(t)$ is a Markov chain on Ω. Letting the row vector α_t denote the distribution of $X(t)$, and P the transition probability matrix, standard Markov chain theory provides that

$$\alpha_{t+1} = \alpha_1 P^t$$

where α_1 is the starting distribution μ. Ordering the points of Ω according to M first, then basin B_1 and so on, the transition matrix P assumes the form

$$P = \begin{bmatrix} M & 0 & \dots & 0 \\ L & B_1 & \dots & L \\ \vdots & \vdots & \ddots & \vdots \\ L & L & \dots & B_n \end{bmatrix}. \tag{2}$$

By overload of notation, we also use B_i to denote the matrix corresponding to basin B_i, $i = 1, \dots, n$. The same for the sub-matrix M. By ordering the points, $x_{i,j} \in B_i$, $j = 1, \dots, |B_i|$, of each such basin beginning at the root node and working up the tree, each sub-matrix B_i assumes of the form

$$B_i = \begin{bmatrix} \mu(x_{i,1}) & \mu(x_{i,2}) & \mu(x_{i,3}) & \dots & \mu(x_{i,|B_i|}) \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix},$$

the 1's are in the lower triangle but not necessarily on the sub-diagonal. The blocks designated by L in (2), for instance the one in the kth column, are generic for the form

$$L = \begin{bmatrix} \mu(x_{k,1}) & \mu(x_{k,2}) & \dots & \mu(x_{k,|B_k|}) \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

In putting these blocks together to form P, the row corresponding to a local minimum pertains to a restart with distribution μ. Finally, M is the block corresponding to the min-bin which can be comprised of one or more goal basins. Its structure, similar to the B_i, consists of 1's in the lower triangular part on each row not corresponding to a global minimizer.

In most problems, global minima are not recognized as such when encountered. Indeed, this aspect of the search problem engenders the theory of stopping times. Of course the running minimizer $W(t)$ will remain constant henceforth when the global minimum is found, but the search will press on. If in fact global minimizers are not recognized, then the rows of P stemming from a global minimizer are identical to those corresponding to a local minimizer, namely, the distribution defined by μ. There are situations however when a global minimizer is known as such when found. These situations typically occur in inverse problems in which the precise set of inputs are sought giving some known output. In such a case the minimizer is an absorbing state for the chain since one stops searching

at that point; therefore the corresponding row is the delta function δ_{ij} equal to 1 if $i = j$ and 0 otherwise.

There is also a circumstance in which the terms of these rows don't matter, such as when computing the expected hitting time of the chain to a global minimizer; a subject we take up next.

Expected hitting time

Let H be the hitting time random variable to M, that is, the number of iterations t until $X(t) \in M$, and let E be its expectation. Knowing E is important because the min-bin will lead, deterministically, to a point of opt(C). Further E can be found by the method of (Shonkwiler and Van Vleck, 1992) because $\hat{P}$, the matrix which results from P when the rows and columns of the min-bin elements are deleted, is a primitive matrix (see next). In just the same way that $\hat{P}$ is the min-bin deleted version of P, let $\hat{\alpha}_t$ be the min-bin deleted version of α_t.

Theorem 1. *Let λ be the Perron-Frobenius eigenvalue of $\hat{P}$ and λ_2 be its next largest eigenvalue (in modulus). Let χ be the corresponding normalized right eigenvector and put $s = 1/(\hat{\alpha}_1 \cdot \chi)$ (dot product). Then there is a fixed polynomial $h(t)$ such that*

$$\left| \frac{1}{\lambda^t} Pr(X(t+1) \notin M) - \frac{1}{s} \right| < h(t) \left(\frac{|\lambda_2|}{\lambda} \right)^t, \qquad t = 1, 2, \ldots. \tag{3}$$

Proof. Take M to be the goal in (Shonkwiler and Van Vleck, 1992). The results of that work imply the conclusion provided the deleted matrix $\hat{P}$ is primitive. But it is clear from the forest of trees graph (with the restart edges included) that each non-min-bin state communicates with every other, possibly through a restart. And, since every local minimizer can follow itself, the process is both irreducible and aperiodic. Hence $\hat{P}$ is a primitive matrix. ∎

Corollary. *With terms as in the Theorem, for some integer K,*

$$\left| \frac{1}{\lambda^t} Pr(W(t) \notin opt(C)) - \frac{1}{s\lambda^{d_m}} \right| = O\left(t^K \left(\frac{|\lambda_2|}{\lambda} \right)^t \right). \tag{4}$$

Proof. Since at most a finite number of additional steps after entering M lead to opt(C), say d_m (see below), we can write

$$Pr(X(t) \in opt(C)) \geq Pr(X(t - d_m) \in M).$$

Hence from the Theorem we have

$$\left| \frac{\lambda^{d_m}}{\lambda^t} Pr(W(t) \notin opt(C)) - \frac{1}{s} \right| \leq h(t - d_m) \left(\frac{\lambda}{|\lambda_2|} \right)^{d_m} \left(\frac{|\lambda_2|}{\lambda} \right)^t$$

from which the conclusion follows. ∎

Remark. *The probability vector ω acts something like a stationary distribution since*

$$\omega \hat{P} = \lambda \omega;$$

the relative probabilities among the states remain fixed, rather the overall probability of being in a non min-bin state decreases by the factor λ. Thus it is clear that retention is the asymptotic probability that the process does not find M (taken as the goal here) in a given iteration; that is the probability that the process is retained in the non min-bin basins. In almost all large problems retention is very close to (but just less than) 1; values $\lambda > .99999$ are the rule.

Remark. *Since expected hitting time is given by*

$$E = \sum_{t=1}^{\infty} Pr(X(t) \notin M),$$

from (3) an estimate of E is

$$E \approx \frac{1}{s} \frac{1}{1-\lambda}. \tag{5}$$

Definition 1. We refer to the probability $\Pr(W(t) \notin \text{opt}(C))$ as the *complementary hitting time distribution, chd*(t).

3. Fundamental Polynomial

Because of the special structure of P, both retention and acceleration can be calculated directly. In the forest of trees model, it is clear that all states which are a given number of steps from a settling point are equivalent as far as finding a global minimizer is concerned. Therefore we construct a simplified but equivalent graph model.

The equivalent graph of the iterative improvement process consists of two linear directed graphs, one for the min-bin and one for all other points of Ω; in essence, two unbranched trees. The root of the min-bin tree identifies to $\text{opt}(C)$ which we may also denote as $\text{opt}_0(C)$. The next node above this one identifies to all vertices of the min-bin connected by an edge to a global minimizer. Denote this subset of Ω by $\text{opt}_1(C)$; that is to say $\text{opt}_1(C) = \{x \in \Omega : g(x) \in \text{opt}(C)\}$. In a similar fashion, define

$$\text{opt}_{j+1}(C) = \{x \in \Omega : g(x) \in \text{opt}_j(C)\}.$$

Thus all points of $\text{opt}_j(C)$ are j steps from a global minimizer and will identify to the jth vertex above the root node of the min-bin tree. The depth, d_m of this tree is equal to the largest depth of a goal basin tree of the previous model.

Similarly the root node of the non min-bin tree identifies with the set of local minimizers which are not global minimizers. Denote this set by $\mathrm{Lopt}_0(C)$ (or just $\mathrm{Lopt}(C)$). Recursively define

$$\mathrm{Lopt}_{j+1}(C) = \{x \in \Omega : g(x) \in \mathrm{Lopt}_j(C)\} \qquad j = 0, 1, \ldots.$$

The jth vertex above the root vertex of the non min-bin tree represents the set $\mathrm{Lopt}_j(C)$. Let d denote the depth of this tree; d is equal to the largest depth among the non-goal basins of the previous model. The weight of all edges of these two, as yet, disconnected unbranched trees is 1.

Adding restart modifies the graph by adding an edge from the root vertex of the non min-bin tree to all other vertices. The weight of these new edges is easy to figure and depends on the restart measure μ. In particular let $q_j = \mu(\mathrm{opt}_j(C))$, $j = 0, 1, \ldots, d_m$ and let $r_j = \mu(\mathrm{Lopt}_j(C))$, $j = 0, 1, \ldots, d$. These are the weights applied to the restart edges. That is, the probability of restarting j steps from a global minimizer is q_j and j steps from a local minimizer is r_j.

Under the equivalency, the min-bin deleted transition probability matrix becomes

$$P' = \begin{bmatrix} r_0 & r_1 & r_2 & \ldots & r_{d-1} & r_d \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}, \tag{6}$$

(the 1's are on the sub-diagonal here).

Theorem 2. *The characteristic polynomial of P' is*

$$-\lambda^{d+1} + r_0\lambda^d + r_1\lambda^{d-1} + \ldots + r_{d-1}\lambda + r_d. \tag{7}$$

Proof. Expand $\det(P' - \lambda I)$ *by minors along the first row.* ∎

Definition 2. In what follows, we shall find it easier to work with a slightly different form of this polynomial; set $\xi = 1/\lambda$ and multiply through by ξ^{d+1}. We refer to the result, $f(\xi)$, as the *fundamental polynomial* of P,

$$f(\xi) = r_0\xi + r_1\xi^2 + \ldots + r_{d-1}\xi^d + r_d\xi^{d+1} - 1. \tag{8}$$

By construction, every root of (7) is also a root of the fundamental polynomial and conversely. Note that the degree of the fundamental polynomial, $d + 1$, is the number of distinct steps in the deepest non goal basin; usually far smaller than the dimension of P.

Theorem 3. *All non-zero eigenvalues of the original matrix $\hat{P}$ are eigenvalues of P'. Therefore, the reciprocal of every eigenvalue of P' is a root of (8), and conversely, the reciprocal of every root of (8) is an eigenvalue of P'.*

Proof. We show that two nodes may be merged. Since the complete merging of nodes can be achieved by successively merging two at a time, this will prove the theorem. Hence suppose nodes x_s and x_t lead to x_m. Then both rows s and t of $\hat{P}$ have the form $(\delta_m) = (0 \quad 0 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 0)$ with the 1 in the mth position and 0's elsewhere. Under the equivalence, assume that node t is merged with node s whose new chance of being selected for restart is therefore the sum $\mu_s + \mu_t$. Row s of P' will be (δ_m) while column s of P' will be the sum of columns s and t of $\hat{P}$, namely $\mu_s + \mu_t$ for rows corresponding to local minima and 0 for the other rows.

To simplify, we may (and do) re-index both matrices. For $\hat{P}$ write x_s first and x_t next followed by the other nodes in their same order. Then $\hat{P} - \lambda I$ will be

$$\begin{bmatrix} -\lambda & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & -\lambda & \dots & 0 & 1 & 0 & \dots & 0 \\ p_{31} & p_{32} & \cdots & p_{3,m-1} & p_{3m} & p_{3,m+1} & \cdots & p_{3n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \dots & p_{n,m-1} & p_{nm} & p_{n,m+1} & \dots & p_{nn} - \lambda \end{bmatrix}$$

Expanding $\det(\hat{P} - \lambda I)$ by the first row in minors gives

$$(-\lambda) \det \begin{bmatrix} -\lambda & \dots & 0 & 1 & 0 & \dots & 0 \\ p_{32} & \cdots & p_{3,m-1} & p_{3m} & p_{3,m+1} & \cdots & p_{3n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n2} & \dots & p_{n,m-1} & p_{nm} & p_{n,m+1} & \dots & p_{nn} - \lambda \end{bmatrix} +$$

$$(-1)^{m+1} det \begin{bmatrix} 0 & -\lambda & \dots & 0 & 0 & \dots & 0 \\ p_{31} & p_{32} & \cdots & p_{3,m-1} & p_{3,m+1} & \cdots & p_{3n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \dots & p_{n,m-1} & p_{n,m+1} & \dots & p_{nn} - \lambda \end{bmatrix}.$$

Now expanding each sub-determinant by its first row in minors gives

$$(-\lambda) \Bigg((-\lambda) \det \begin{bmatrix} p_{33} - \lambda & \dots & p_{3n} \\ \vdots & \ddots & \vdots \\ p_{n3} & \dots & p_{nn} - \lambda \end{bmatrix} +$$

$$(-1)^m \det \begin{bmatrix} p_{32} & p_{33} - \lambda & \dots & p_{3,m-1} & p_{3,m+1} & \dots & p_{3n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{n2} & p_{n3} & \dots & p_{n,m-1} & p_{n,m+1} & \dots & p_{nn} - \lambda \end{bmatrix} \Bigg) -$$

$$(-1)^{m+1}(-\lambda)\det\begin{bmatrix} p_{31} & p_{33}-\lambda & \cdots & p_{3,m-1} & p_{3,m+1} & \cdots & p_{3n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n3} & \cdots & p_{n,m-1} & p_{n,m+1} & \cdots & p_{nn}-\lambda \end{bmatrix}.$$

Factoring $-\lambda$ from all members and adding the last two determinants, since all but their first columns are identical, we get

$$(-\lambda)\det\begin{bmatrix} p_{33}-\lambda & \cdots & p_{3n} \\ \vdots & \ddots & \vdots \\ p_{n3} & \cdots & p_{nn}-\lambda \end{bmatrix} +$$

$$(-1)^{m}\det\begin{bmatrix} p_{31}+p_{32} & p_{33}-\lambda & \cdots & p_{3,m-1} & p_{3,m+1} & \cdots & p_{3n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{n1}+p_{n2} & p_{n3} & \cdots & p_{n,m-1} & p_{n,m+1} & \cdots & p_{nn}-\lambda \end{bmatrix}.$$

But this final result is exactly the expansion of $\det(P'-\lambda I)$ for the reduced matrix P' proving the theorem. In reducing the degree of the characteristic polynomial by one, only the zero root $-\lambda = 0$ has been lost; it occurred in the factor step above. ∎

Definition 3. Let θ_0 be the probability of landing in the min-bin,

$$\theta_0 = \sum_{j=0}^{d_m} \mu(\mathrm{opt}_j(C)).$$

Theorem 4. *For the fundamental polynomial,*

$$f(1) = -\theta_0; \tag{9}$$

in particular, its value at 1 is negative. Moreover, f has a unique root $\eta > 1$. Thus the Perron-Frobenius eigenvalue λ, that is retention, equals reciprocal η

$$\lambda = \frac{1}{\eta}.$$

Proof. Since a restart must select some node,

$$\theta_0 + r_0 + r_1 + \ldots + r_d = 1$$

from which the first conclusion follows. Since $f \to \infty$ as $\xi \to \infty$ and

$$f'(\xi) = r_0 + 2r_1\xi + 3r_2\xi^2 + \ldots + dr_{d-1}\xi^{d-1} + (d+1)r_d\xi^d, \tag{10}$$

is positive for all $\xi \geq 0$, the second conclusion also follows. The Perron-Frobenius eigenvalue of $\hat{P}$ (and also of P') is its largest eigenvalue in modulus, is real and less than 1; hence it must be reciprocal η. ∎

Definition 4. We refer to η as the *principal* root of the fundamental polynomial.

The right Perron-Frobenius eigenvector, χ, of $\hat{P}$ is easily calculated. From (6) we get the recursion equations

$$\begin{aligned}
\chi_0 &= \lambda\chi_1 \\
\chi_1 &= \lambda\chi_2 \\
\vdots &= \vdots \\
\chi_{n-1} &= \lambda\chi_n .
\end{aligned}$$

From this we find

$$\chi_k = \eta^k \chi_0, \qquad k = 1, \ldots, n.$$

Similarly, from (6) we get the following recursion equations for the components of the left Perron-Frobenius eigenvector, ω,

$$\begin{aligned}
\omega_0 r_0 + \omega_1 &= \lambda\omega_0 \\
\omega_0 r_1 + \omega_2 &= \lambda\omega_1 \\
\vdots + \vdots &= \vdots \\
\omega_0 r_{n-1} + \omega_n &= \lambda\omega_{n-1} \\
\omega_0 r_n &= \lambda\omega_n .
\end{aligned} \tag{11}$$

From these we get equations for the components in terms of ω_0;

$$\begin{aligned}
\omega_n &= \omega_0 \eta r_n \\
\omega_{n-1} &= \omega_0(\eta r_{n-1} + \eta^2 r_n) \\
\vdots &= \vdots \\
\omega_1 &= \omega_0(\eta r_1 + \eta^2 r_2 + \ldots + \eta^n r_n).
\end{aligned} \tag{12}$$

The sum of the system (11) in conjunction with the normalization, $\sum \omega_i = 1$, gives

$$\begin{aligned}
\omega_0 \sum r_i + \sum_i^n \omega_i &= 1/\eta \\
\omega_0(1-\theta_0) + 1 - \omega_0 &= 1/\eta .
\end{aligned}$$

Solve for ω_0 to get

$$\omega_0 = \frac{\eta - 1}{\eta\theta_0}. \tag{13}$$

Further recall that under the normalization, $\sum \omega_i \chi_i = 1$; hence, using (12) and $f(\eta) = 0$,

$$\begin{aligned}
\frac{1}{\omega_0 \chi_0} &= 1 + \eta^2 r_1 + 2\eta^3 r_2 + \ldots + n\eta^{n+1} r_n \\
\frac{1}{\omega_0 \chi_0} &= \eta r_0 + 2\eta^2 r_1 + 3\eta^3 r_2 + \ldots + (n+1)\eta^{n+1} r_n \\
\frac{1}{\omega_0 \chi_0 \eta} &= r_0 + 2\eta r_1 + 3\eta^2 r_2 + \ldots + (n+1)\eta^n r_n \\
\frac{1}{\omega_0 \chi_0 \eta} &= f'(\eta).
\end{aligned}$$

From this we get that

$$\chi_0 = \frac{1}{\omega_0 \eta f'(\eta)}.$$

Finally we calculate $s = 1/(\chi \cdot \alpha_1')$ where α_1' is the non min-bin partition vector of the starting distribution (for the equivalent process); thus

$$\alpha_1' = (\, r_0 \quad r_1 \quad \ldots \quad r_d \,).$$

Therefore

$$\begin{aligned}
\frac{1}{s} &= r_0 \chi_0 + r_1 \chi_1 + \ldots + r_d \chi_d \\
&= r_0 \chi_0 + r_1 \eta \chi_0 + r_2 \eta^2 \chi_0 + \ldots + r_d \eta^d \chi_0 \\
&= \frac{\chi_0}{\eta} = \frac{1}{\eta^2 \omega_0 f'(\eta)}
\end{aligned}$$

So

$$s = \frac{\eta(\eta - 1) f'(\eta)}{-f(1)}. \tag{14}$$

The foregoing prove the following theorem.

Theorem 5. *Acceleration s is given by equation (14) in terms of the fundamental polynomial and its principal root.*

Theorem 6. *For iterative improvement with restart, $s \geq \lambda^{-1} > 1$.*

Proof. The secant line joining the points $(1, f(1))$ and $(\eta, 0)$ has slope $-f(1)/(\eta - 1)$. By the Mean Value Theorem, for some point $1 < \xi < \eta$,

$$f'(\xi) = \frac{-f(1)}{\eta - 1}.$$

But since $f'' > 0$ on $[1, \infty)$, it follows that

$$f'(\eta) > f'(\xi).$$

Hence

$$\frac{s}{\eta} = \frac{f'(\eta)}{(-f(1)/(\eta-1)} > 1.$$

∎

Run time estimation of retention, acceleration and hitting time

Remark. Returning to the fundamental polynomial, we notice that its coefficients are the various probabilities for restarting a given distance from a local minimum. Thus the linear coefficient is the probability of restarting on a local minimum, the quadratic coefficient is the probability of restarting one iteration from a local minimum and so on.

As a result, it is possible to estimate the fundamental polynomial during a run by keeping track of the number of iterations spent in the downhill processes. Using the estimate of the fundamental polynomial, estimates of retention and acceleration and hence also expected hitting time can be affected. As a run proceeds, the coefficient estimates converge to their right values and so does the estimate of E, see §7.

4. Hitting time to a global minimizer.

Let $\mathcal{E}$ denote the expected hitting time directly to $\text{opt}(C)$ calculated from a start up. By combining the hitting time E to the min-bin together with the conditional distribution of landing in the min-bin, $\mathcal{E}$ can be calculated. Thus

$$\mathcal{E} = E + \sum_{j=1}^{d_m} j\mu(\text{opt}_j(C)). \tag{15}$$

Alternatively, $\mathcal{E}$ can be computed directly from the equivalent matrix description. The full equivalent matrix can be written

$$\begin{bmatrix} A & Z \\ Q & P' \end{bmatrix}$$

where Z is the $(d_m+1)\times(d+1)$ matrix of 0's, P' is as above, A is the matrix with 1's on the sub-diagonal (beginning with the second row, $a_{11}=1$) and Q is the $(d+1)\times(d_m+1)$ matrix

$$Q = \begin{bmatrix} q_0 & q_1 & q_2 & \cdots & q_{d_m-1} & q_d \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

Hence the $\text{opt}(C)$ deleted matrix is

$$\bar{P} = \begin{bmatrix} A' & Z' \\ Q' & P' \end{bmatrix}$$

where Z' is Z with the first row deleted, Q' is Q with the first column deleted and

$$A' = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

Theorem 7. *The matrix $\bar{P}$ has a unique eigenvalue, $\bar{\lambda}$, largest in modulus; it is equal to λ. The corresponding normalized left eigenvector is*

$$\begin{pmatrix} a_1 & a_2 & \dots & a_{d_m-1} & a_{d_m} & \frac{1}{z}\omega \end{pmatrix}$$

where ω is the left eigenvector of P' from above and

$$a_j = \omega_0(\eta q_j + \eta^2 q_{j+1} + \ldots + \eta^{d_m-j+1} q_{d_m}) \qquad z = \sum_{j=1}^{d_m-1} a_j.$$

The normalized right eigenvector is

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ z\chi \end{bmatrix}$$

where χ is the right eigenvector of P' as above.

Proof. Due to the block matrix Z' of 0's, we have

$$\det(\bar{P} - \lambda I) = \det(A' - \lambda I)\det(P' - \lambda I).$$

But the first of these works out to be $(-\lambda)^{d_m}$ contributing d_m 0's to the eigenvalues of $\bar{P}$. Hence the non zero eigenvalues of $\bar{P}$ are the same as those of P'.

The remaining conclusions are verified by direct calculation. ∎

The convergence toward $\bar{\omega}$ occurs provided the process starts in the non min-bin states; such a starting vector α has 0's for its first d_m elements. Hence the dot product $\bar{\alpha}\bar{\chi} = \hat{\alpha}\chi = 1/s$. So acceleration and retention for P' is the same as that for $\hat{P}$. Further,

$$\Pr(X(t) \notin \text{opt}(C)) = \bar{\alpha}_t \mathbf{1} \ = \lambda^t (1/s)\bar{\omega}\mathbf{1} \ = \frac{\lambda^t}{s}$$

as before.

5. Choice of a Probability Measure for Restart

In this section arguments in favor of using the uniform measure on the state space for restarting are presented. The sense in which this is a good measure is this: if a random restart method is to be used to find a global minimum of a function f, and if this function is unknown (pointwise evaluations can be made of f and possibly some of its derivatives or other functionals, but the general behavior of f on Ω is not known) and could be any one from a sufficiently large collection of functions, then by using any other measure for restarting, the worst case is worse than using the uniform measure.

Choosing the restart measure is a game theoretic problem. We start by assuming there is a way to rank the performance of the search process when applied to a specific cost function. This performance, Π, might be taken as reciprocal expected hitting time (so smaller times correspond to higher performances) or possibly as reciprocal median hitting time. We assume Ω, $\mathcal{F}$, and g to be given leaving $\Pi = \Pi(\mu, C)$ to depend on the choice of restart distribution and cost. Certainly the more the objective class is restricted, the more the restart distribution can be crafted.

Since the calculation of expected hitting time depends in a complicated way on g, and therefore on specific cases, we will simplify matters by taking $\Pi(\mu, C)$ to be $\theta_0 = \mu(M_C)$, the probability of restarting in the min-bin for cost C, which is an estimate of reciprocal expected hitting time.

Next, there must be a criterion for interpreting performance differences. One possibility, and the one we will use, is *mini-max*; that is we select the restart distribution which maximizes the performance of the worst performing objective function. This is the right criterion against an intelligent adversary who would always choose the worst cost function for a given restart distribuition.

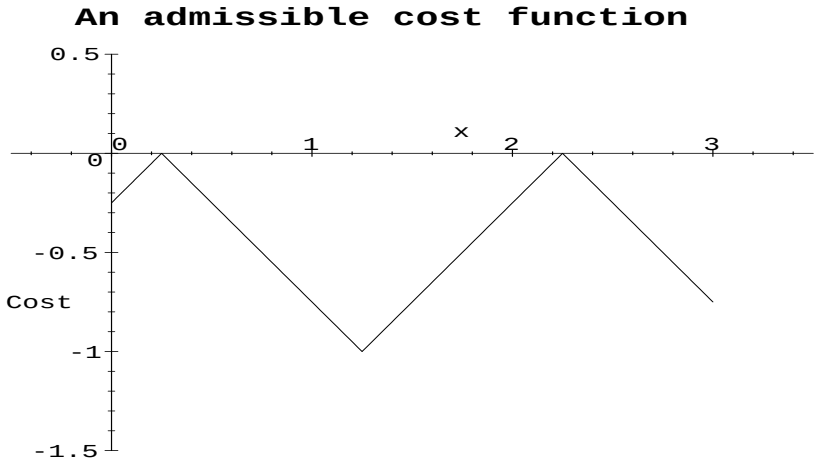


Fig. 1

Example.

Let Ω be the interval $[0,3] \subset \mathbb{R}$ and take $\mathcal{F}$ to be all continuous, piecewise linear functions on Ω such that: (1) all local maxima lie on the x-axis strictly outside $[1,2]$, (2) all line segments have slope equal to ± 1 except the initial slope may be less than -1 and the final slope may exceed $+1$, and (3) at least one basin has depth at least 1. This class is sufficiently large that every point $x_0 \in \Omega$ is the unique global minimizer of some cost $C \in \mathcal{F}$, see Fig. 1. Let g be gradient descent plus line search, then exactly one iteration is required to find the settling point of each basin. Here a mini-max restart distribution is any whose support is the subinterval $[1,2]$.

Definition 5. If Ω is a finite space, let the *volume* V of a set refer to its cardinality; if Ω is a subset of Euclidean space then take V to be Euclidean volume.

Theorem 8. *Suppose Ω is either finite or a compact subset of $\mathbb{R}^n$. If the objective class $\mathcal{F}$ is such that whenever some set $S \subset \Omega$ of a given volume V is the min-bin of some cost, then every subset of Ω of volume V is also the min-bin of some cost, then uniform restart is mini-max optimal.*

Proof. Let $V_0 = \inf\{V : \text{some cost has min-bin of size } V\}$ and let M_{V_0} denote a min-bin of size V_0. If $V_0 = 0$ then the conclusion is true with a zero mini-max value. Otherwise, for the uniform measure μ and any other measure ν,

$$\inf_C \nu(M_C) \leq \nu(M_{V_0}) \leq \mu(M_{V_0}) = \mu(V_0).$$

Hence

$$\sup_\nu \inf_C \nu(M_C) \leq \inf_C \mu(M_C) = \mu(V_0)$$

showing that μ is the mini-max solution with value $\mu(V_0)$. ∎

6. IIP parallelization

A simple method for parallelizing stochastic search referred to as IIP parallel, meaning independent and identical processes, is studied in (Shonkwiler and Van Vleck, 1992). Specifically, assume m identical copies of an algorithm such as I^2R^2 are run independently of each other on m processors with only their random number seeds differing. The m process expected hitting time, E_m, is the expectation of the first process to find the min-bin. That is, E_m is the expectation of the random variable

$$H = \min\{H_1, H_2, \ldots, H_m\}$$

where H_i is the min-bin hitting time for the ith process. The expectation $\mathcal{E}_m$ is similarly defined relative to a global minimizer.

Theorem 9. *Pr(multi-process not in M by time t)* $\sim \frac{1}{s^m}\lambda^{mt}$ *as* $t \to \infty$.

Proof. Since the event $H \geq t$ is equivalent to the event

$$H_1 \geq t \quad \text{and} \quad H_2 \geq t \quad \text{and} \cdots \text{and} \quad H_m \geq t,$$

by independence we have the following,

$$\Pr(H \geq t) = (\Pr(H_1 \geq t))^m, \quad t = 1, 2, \ldots.$$

Therefore, from (3),

$$\Pr(\text{multi-process not in } M \text{ by time } t) = \left[\frac{1}{s}\lambda^t + h(t)\left(\frac{|\lambda_2|}{\lambda}\right)^t\right]^m$$
$$= \frac{1}{s^m}\lambda^{mt} + O\left(t^K\left(\frac{|\lambda_2|}{\lambda}\right)^t\right)$$

for some integer K. ∎

Remark. *As before, since*

$$E_m = \sum_{t=0}^{\infty}(Pr(H_1 \geq t))^m,$$

the above leads to the following approximation for the expected hitting time for the IIP process

$$E_m \approx \frac{1}{s^m}\sum_{t=0}^{\infty}\lambda^{mt} = \frac{1}{s^m}\frac{1}{1-\lambda^m}.$$

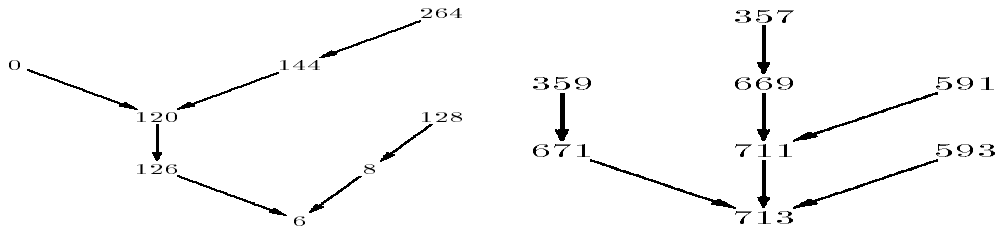
Remark. *It is proved in (Shonkwiler and Van Vleck, 1992) that m processor IIP speed-up SU_m, defined as the ratio E_1/E_m, satisfies*

$$SU_m = s^{m-1}\frac{1-\lambda^m}{1-\lambda} + O(1-\lambda^m).$$

Recall from Theorem 6 that for I^2R^2, $s > 1$; hence one sees that IIP is an especially effective technique for I^2R^2.

7. Numerical Results

In this section the foregoing results are demonstrated on two example problems, one discrete and the other continuous. The first is the Traveling Salesman Problem (TSP) on



Basin for Tour (1,2,4,3,5,6) Basin for Tour (6,5,3,4,2,1)

Fig. 2

7 cities (keeping the size small enough so that exact calculations will be possible) and the other is the minimization of a one variable multi-modal function.

For the TSP on 7 cities there are 6! possible tours so that $N = 720$ is the number of points in Ω. (Pairing every tour with its reverse leads to 360 distinct potential solutions but we ignore these pairings). The cities, placed in a 10×10 square, have locations 0:(2,2), 1:(7,3), 2:(4,5), 3:(8,7), 4:(1,6), 5:(6,9), 6:(3,8). The space Ω of tours is 0 (0,1,2,3,4,5,6,0); 1 (0,1,2,3,4,6,5,0); 2 (0,1,2,3,5,4,6,0); ...; 719 (0,6,5,4,3,2,1,0). The minimum tour length is 24.27 achieved by Tour 123 (0,2,1,3,5,6,4,0) and its reverse Tour 478. Tours always begin and end with city 0 so we omit it in the description henceforth.

The improvement algorithm g is taken as "successive city swap" defined by: given a Tour, then (step 1) start at the leftmost tour position $i = 1$, and (step 2) swap the cities in positions i and $i + 1$, (step 3) if the tour length has not increased, replace Tour by this modification and go to step 1, otherwise increment i and (step 4) if $i = 6$ return Tour, otherwise go to step 2.

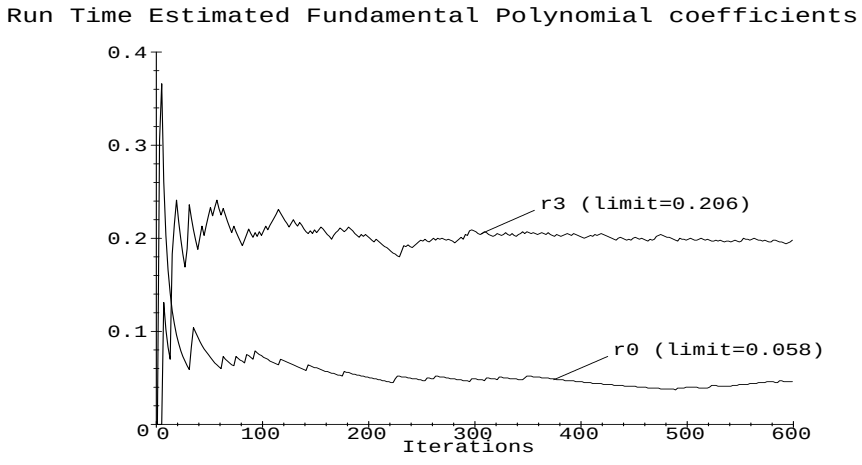


Fig. 3

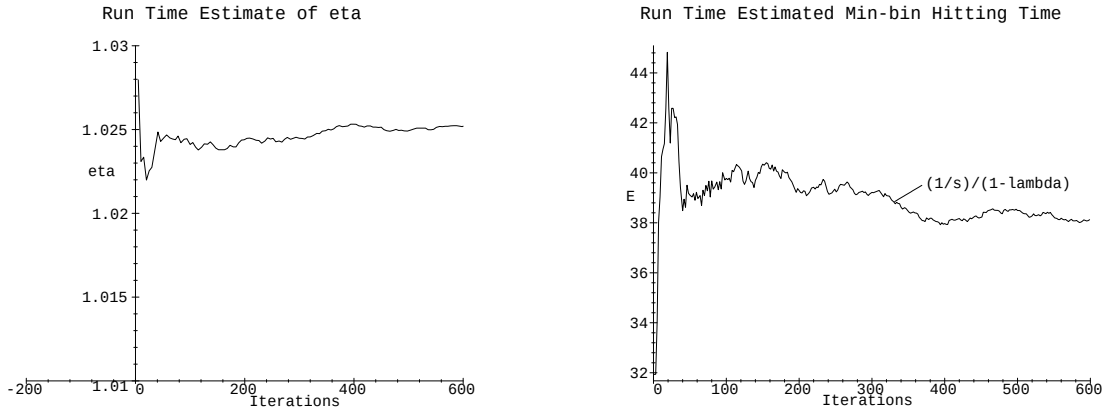
Among the basins defined by g is the depth 4 basin shown in Fig. 2(a) ending with Tour

$6 = (1, 2, 4, 3, 5, 6)$ which is a local minimizer. Successive city swap breaks the symmetry between Tours and their reverse; the basin for Tour 713, which is the reverse of Tour 6, has depth 3 and is shown in Fig.2(b).

The topology induced by g consists of 44 basins of which 2 are goal basins. The goal basins comprise 62 points so that $\theta_0 = 62/720 = 0.0861$. The length of the longest min-bin path is $d_m = 6$. The longest non min-bin path has length $d = 9$ so the structure polynomial is of degree 10. The structure polynomial can be exactly calculated for this problem and is

$$f(\xi) = \frac{1}{720}(42\xi + 130\xi^2 + 174\xi^3 + 148\xi^4 + 93\xi^5 + 44\xi^6 + 18\xi^7 + 6\xi^8 + 2\xi^9 + \xi^{10}) - 1.$$

Its principal root is $\eta = 1.0254$, hence it follows from Theorem 4 that retention $\lambda = \eta^{-1} = 0.9753$ and, from equation (14), that acceleration $s = 1.067$. As predicted by Theorem 6, $s > \eta$. The estimated min-bin hitting time from equation (5) is 37.94.



Runtime Estimate of η Runtime Estimate of E

Fig. 4

Empirical data were acquired by executing the I^2R^2 algorithm on this problem until a global optimizer was found 200 times. Keeping track of the depth data for each (non min-bin) restart, it is possible to make run time estimates of the coefficients of the fundamental polynomial as discussed in §3. Some of these estimates are shown in Fig. 3. From the resulting estimated fundamental polynomial, run time estimates of the principal root η can be calculated; these are shown in Fig. 4(a). Then using (5), min-bin hitting time can be predicted as shown in Fig. 4(b).

Finally, by the independence of the IIP method, the data were used to simulate parallel processing; the speedup results are shown in Fig. 5. Linear speedup is also shown in the figure for comparison.

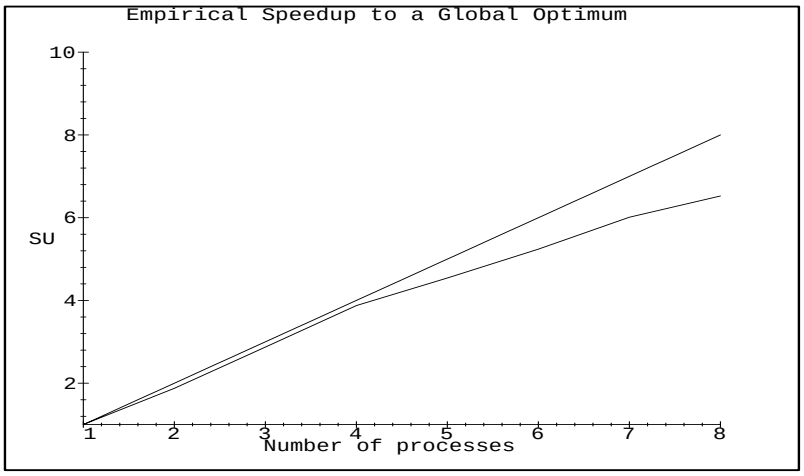


Fig. 5

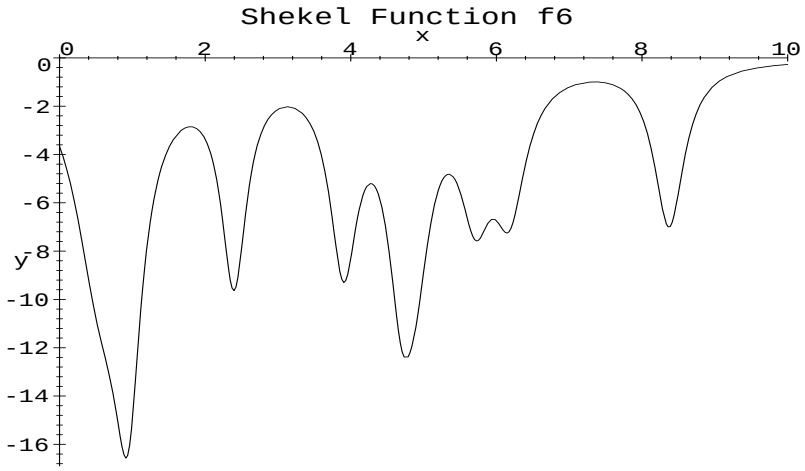


Fig. 6

Shekel function example

Our second problem, shown in Fig. 6, is an example of a Shekel function and is the function f_6 in (Törn and Zelinskas, 1989, p177). The function is given by

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(k_i(x - a_i))^2 + c_i}, \qquad 0 \leq x \leq 10,$$

where the parameters a_i, c_i, and k_i are as in Table 1.

Shekel Function Parameters

a	k	c
4.696	2.871	0.149
4.885	2.328	0.166
0.800	1.111	0.175
.4986	1.263	0.183
3.901	2.399	0.128
2.395	2.629	0.117
0.945	2.853	0.115
8.371	2.344	0.148
6.181	2.592	0.188
5.713	2.929	0.198

Table 1

In this problem the global minimum is -16.6 and occurs at $x = 0.925$. The min-bin is comprised of the range $0 < x < 1.8$.

For the improvement algorithm g in this problem we used Newton's Method for Unconstrained Minimization (see (Dennis and Schnabel, 1983)) with the modification that no Newton's step was allowed to exceed a pre-calculated maximum, MAXSTEP. This value was based on the reciprocal of the maximum curvature of the graph and assured that every downhill sequence remained in its restart bin.

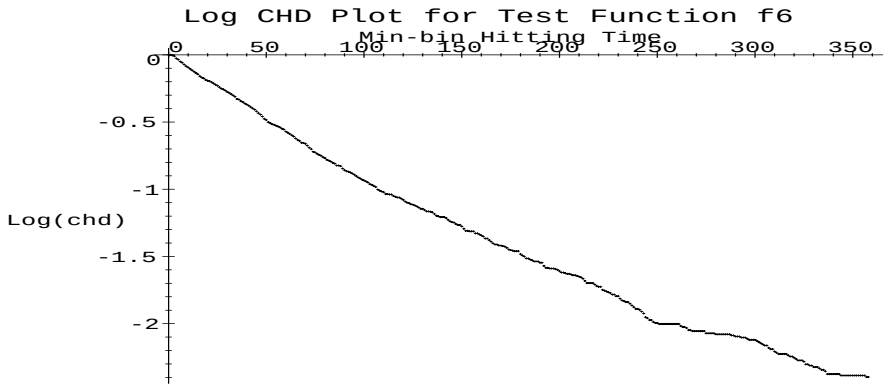


Fig. 7

For each trial, the number of iterations required to find both the min-bin and the global minimum (within 0.6) was observed. In order to do run time estimates, numbers of iterations in non goal basins was also observed. From the raw data consisting of 1000

such trials, the average value of min-bin hitting time $E = 85$ and global minimum hitting time $\mathcal{E} = 103$.

In this example we used the data to determine the complementary hitting time distribution (chd). Thus for each k, $chd(k)$ is the fraction of runs requiring k or more iterations. Theoretically, the chd is asymptotically geometric,

$$chd(k) \sim \frac{1}{s}\lambda^{k-1}.$$

Therefore a $\log(chd)$ vs $k-1$ plot should tend to a straight line with slope $\log(\lambda)$. This plot, Fig. 7, was indeed observed to be approximately affine in its central region and least squares was used to calculate its slope. By this method, $\lambda = 0.992$.

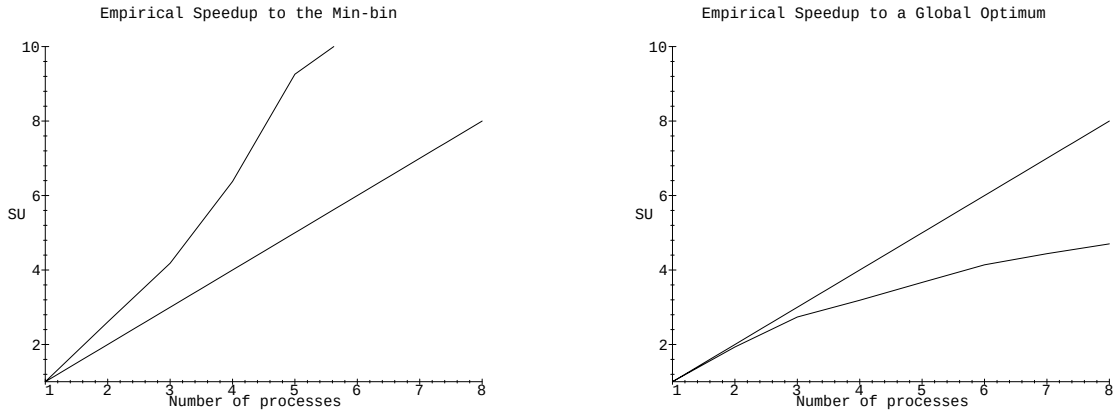


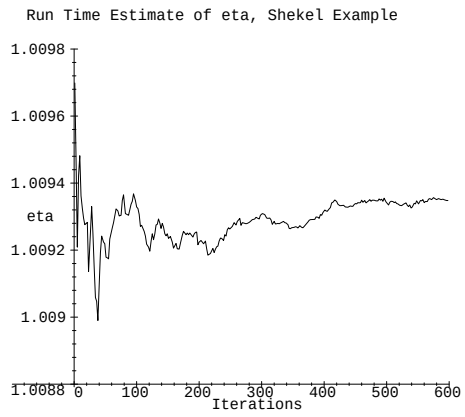
Fig. 8

The chd plot is not suitable for determining the acceleration factor s however as its intercept is too sensitive to the choice of affine region selected. Instead the acceleration is estimated from speedup data to the min-bin. Speedup data for m parallel processes were obtained by the *in-code parallel* technique. That is, m separate trial solutions are maintained in each iteration of a single processor algorithm. The first of these to reach the min–bin flags a stop to the others. The hitting time is noted, and a new run initiated. The speedup plot is given in Fig. 8; linear speedup is shown for comparison. Since $SU \approx ms^{m-1}$, $\log(s)$ is the slope of the plot of $\log(SU/m)$ versus m, This worked out to be $s = 1.094$. Since $1/\lambda = 1.008$, as predicted, $s > 1/\lambda$.

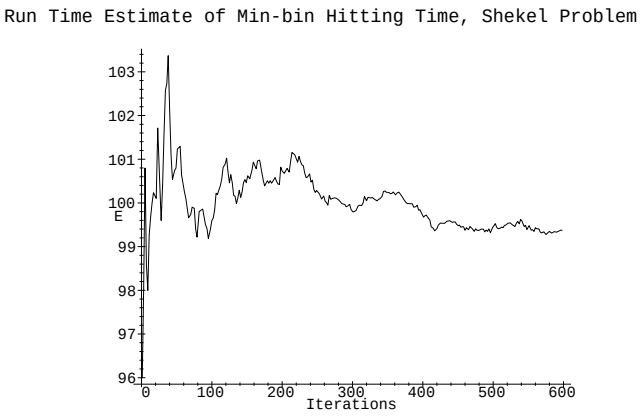
As in the TSP problem, run time estimations of the fundamental polynomial and its principal root were made. The result is shown in Fig. 9(a). From that, min-bin hitting time is estimated as shown in Fig. 9.(b)

References

[1] Dimitri Bertsekas, John Tsitsiklis,*Parallel and Distributed Computation*, Prentice Hall, Englewood Cliffs (1989)



Runtime Estimate of *eta*



Runtime Estimate of E

Fig. 9

[2] G. Boender, and A. Rinnooy Kan, *Bayesian Stopping Rules for Multistart Optimization Methods*, Math. Programming, 37(1987), pp .59–80.

[3] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, (1983)

[4] J. Galambos, *The Asymptotic Theory of Extreme Order Statistics*, Wiley, New York, (1978)

[5] J. Pickands, *Moment Convergence of Sample Extremes*, Ann. Math. Statist., 39(1968), pp. 881-889.

[6] F. Schoen, *Stochastic Techniques for Global Optimization: A Survey of Recent Advances*, J. of Global Optimization, 1(1991), pp. 207–228.

[7] R. Shonkwiler, and E. Van Vleck, *Parallel speed-up of Monte Carlo methods for global optimization*, to appear in the Journal of Complexity.

[8] F. Solis, and R. Wets, *Minimization by Random Search Techniques*, Math. of Operations Research, 6(1981), pp. 19–30.

[9] A. Törn, and A. Zelinskas, *Global Optimization*, Springer-Verlag, New York, (1989)

[10] R. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ (1963).