Fibonacci Search with Multiple Probes

Yaw-Ling Lin* and Shi-Chun Tsai†

Abstract

Binary search is often used to locate a query point in monotonically increasing function. When the searched target is not a monotonic function, other search strategies must be used. For example, the Fibonacci search is generally used to locate the desired minimum / maximum point on a convex function.

In this paper, we present results on parallel fibonacci search, where we allow the searching agent to use a set of k simultaneous probes on a convex function for each query.

Keywords: *Fibonacci search, convex function, nonlinear programming*

1 Introduction

A set S is called *convex* if $\lambda x + (1-\lambda)y \in S$ for any $x, y \in S$ and $0 \le \lambda \le 1$. Let f be a real valued function on a convex set S. We call f a convex function, if for any $x, y \in S$ and $0 \le \lambda \le 1$, $f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda)f(y)$. Consider a search problem to minimize a convex function $f(x)$ subject to $a \le x \le b$. The minimum of f is unknown. Our goal is to locate the minimum over $[a, b]$ within a small interval efficiently. Binary search is used to locate a query point in monotonically increasing function. There are methods, such as Fibonacci search and golden section method, used to locate the minimum over a convex function [0], where both methods starting by making two functional evaluations and one evaluation at each of the subsequent iterations. Fibonacci search is a basic tool used in nonlinear programming [0] and in some computer vision problem [0]. In this note we study the case when multiple functional evaluations are allowed in each iteration. In other words, we study the parallel version of Fibonacci search. First we have the following observations:

*Dept of Computer Science and Info. Management, Providence University, Sha-Lu, Taichung 433, Taiwan, R.O.C. Email: `yllin@pu.edu.tw`

†Dept of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, Email: `sctsai@csie.nctu.edu.tw`

Observation 1 *Let f be a convex function defined over interval $[a, b]$, and let $a < x < b \in \mathbb{R}$. By definition, we have: $f(x) \le [(b-x)f(a) + (x-a)f(b)]/(b-a)$. Further, let $f(x^*) = \min\{f(x) | x \in [a, b]\}$ be the minimum value within interval $[a, b]$. It is equally possible that $x^* \in [a, x]$ or $[x, b]$ for any $x \in [a, b]$. On the other hand, let $a < x < y < b \in \mathbb{R}$. We have: $x^* \in [a, y]$ if $f(x) \le f(y)$; otherwise, $x^* \in [x, b]$ if $f(x) \ge f(y)$.*

The observation is easily generalized to multiple consecutive segments of an interval.

Observation 2 *Let f be a convex function defined over interval $[a, b]$, and let $a = x_0 < x_1 < x_2 < \cdots < x_k < b = x_{k+1}$. Let x_m produce the minimum value among all $f(x_i)$'s; i.e., $f(x_m) = \min\{f(x_i) | 1 \le i \le k, i \in \mathbb{N}\}$. Further, let $f(x^*) = \min\{f(x) | x \in [a, b]\}$ be the minimum value within interval $[a, b]$. It follows that $x^* \in [x_{m-1}, x_{m+1}]$.*

Proof. The results follows easily by the definition of convex function. □

The observation implies that we can speed up the process of locating the minimum if multiple queries are allowed. Note that $k-2$ segments can be thrown away from the original $k+1$ segments; only two consecutive segments remains to be further examined. Without loss of generality, assume that $(x_m - x_{m-1}) \le (x_{m+1} - x_m)$, and let $\alpha = (x_m - x_{m-1})/(x_{m+1} - x_m) \le 1$ denote the *partition ratio.*

Observation 3 (Principle of Balance) *Let f be a convex function defined over interval $[a, b]$, and let $a < x < b \in \mathbb{R}$ with $\alpha = (x-a)/(b-x) \le 1$; and the function values $f(a), f(x), f(b)$ are known. The best way to place the k probes within interval $[a, b]$ is to partition the interval $[a, b]$ into $k+2$ segments of lengths $s_1, s_2, \ldots, s_{k+2}$ (also considering the given pivot point x) such that the ratio of the even numbered segment to the odd numbered segment is always α. That is, $s_i/s_{i+1} = \alpha$ if i is even; $s_i/s_{i+1} = 1/\alpha$ if i is odd.*

Proof. The neighboring ratio of the above $k+2$ segments can be illustrated as the following:

$$\underbrace{1\alpha \cdots 1\alpha \quad | \quad 1\alpha \cdots 1}_{k+2}$$

Note that k out of these $k+2$ segments will be thrown away and only 2 consecutive segments remains to be further searched. Since the optimal solution can equally appear in any of these subintervals, any partition other than this can thus produces a worse situation. That is, by the adversary argument we conclude that this is the best strategy to partition the given interval. □

The classical Fibonacci search and golden ratio method consider the most simple case when $k=1$. Again we assume that $(x-a) \leq (b-x)$ as the initial condition. Further, let $\alpha = (x-a)/(b-x) \leq 1$. By Observation 3 we know that $\alpha = (\alpha : 1) = (1-\alpha : \alpha) = (1-\alpha)/\alpha$. It follows that $\alpha^2 + \alpha - 1 = 0$. That is, we have $\alpha = (\sqrt{5}-1)/2 = 0.618$. It is related to the *golden ratio*, defined by $\phi = \lim_{n\to\infty}(F_{n+1}/F_n) = (1+\sqrt{5})/2 = 1.618$; here F_n denotes the n-th Fibonacci number. It is interesting to note that $\alpha = 1/\phi = \phi - 1$. It is the reason why this searching method is also known as Fibonacci search.

The characteristics of the convex function and the principle of balance (Observation 3) dictate that if we were to locate that minimum value (or searching for a given value) then the best possible way is to partition the searching range $[a,b]$ in such a manner that we will have $(x-a) : (b-x) = 1/\phi$ and then place the next query point $y > x$ so that $(b-y) = 1/\phi$. Note that either range $[a,x]$ or range $[y,b]$ will be thrown away so that the searching range will be reduced from $1/\phi + 1 = \phi$ to 1. Namely the *shrinking factor* is exactly $1/\phi$.

Situations change if we can probe k queries at each run simultaneously. Here we describe how the k-Fibonacci search works. Again, by the characteristics of the convex function, the possible searchable range will be just *exactly two* intervals, say $[a,x]$ and $[x,b]$. Without loss of generality we claim that $(x-a) \leq (b-x)$ and let $\alpha = (x-a)/(b-x) \leq 1$. Suppose that we can issue k independent *probes* at each query iteration then we might able to further reduce the shrinking factor so that the final optimal solution can be found in fewer iterations.

2 Main Results

Let ρ_k denote the *shrinking factor* of a k-Fibonacci search. Note that $k=1$ is exactly the case of the classical Fibonacci search and we have $\rho_1 = (\sqrt{5}-1)/2 = 0.618$.

For $k > 1$, we have two cases. That is, either k is an even number or k is odd. The case that $k = 2\ell$ is easy. It is not hard to verify that

Theorem 1 (Even Fibonacci search) *For an even number k, The shrinking factor ρ_k of the k-Fibonacci search is:* $1/(k/2+1)$

Proof. Let $k = 2\ell$, $\ell \in \mathbb{N}$. Divide both intervals $[a,x]$ and $[x,b]$ into $\ell+1$ *equal-sized* intervals. That is, we shrink the original sized 2 intervals into 2 intervals each sized $1/(1+\ell)$. That is, the shrinking factor is $\frac{2}{1+\ell}/2 = 1/(\ell+1) = 1/(k/2+1)$. □

The case for odd number k is not as straight forward. We have two cases of odd numbers. Either $k = 4\ell - 3$ or $k = 4\ell - 1$. Here we have:

Theorem 2 (Odd Fibonacci search) *For an odd number k, let $\ell = \lceil k/4 \rceil$, and $\alpha = (\sqrt{4\ell^2+1} - 1)/2\ell$. The shrinking factor ρ_k of the k-Fibonacci search is exactly:*

$$\rho_k = \frac{1+\alpha}{\lceil k/2 \rceil (1+\alpha) + 1}$$

Proof. We have two cases of odd numbers. Recall that $\ell = \lceil k/4 \rceil$, i.e., either $k = 4\ell - 3$, or $k = 4\ell - 1$. By Observation 3, assume that the optimal partition ratio $\alpha = (x-a)/(b-x) \leq 1$. We first show that for either cases, we both have $\alpha = (\sqrt{4\ell^2+1} - 1)/2\ell$.

For $k = 4\ell - 3$, interval $[a,x]$ is queried by $2\ell - 2$ (even) probes and partitioned into $2\ell - 1$ segments and interval $[x,b]$ queried by $2\ell - 1$ probes and partitioned into 2ℓ segments. Interval $[a,x]$ has ℓ segments of size 1 and $\ell - 1$ segments of size α; interval $[x,b]$ has ℓ segments of size 1 and ℓ segments of size α. In all, by Observation 3, we have $(\ell + (\ell-1)\alpha)/\ell(1+\alpha) = \alpha$. It follows that we are solving equation:

$$\ell\alpha^2 + \alpha - \ell = 0$$

Thus we have $\alpha = (\sqrt{4\ell^2+1} \pm 1)/2\ell$; since $\alpha \leq 1$, we have: $\alpha = (\sqrt{4\ell^2+1} - 1)/2\ell$.

For $k = 4\ell - 1$, interval $[a,x]$ is queried by $2\ell - 1$ (odd) probes and partitioned into 2ℓ segments and interval $[x,b]$ queried by 2ℓ probes and partitioned into $2\ell + 1$ segments. Interval $[a,x]$ has ℓ segments of size 1 and ℓ segments of size α; interval $[x,b]$

k	1	2	3	4	5
ρ_k	0.618	0.5	0.382	0.333	0.281
k	6	7	8		
ρ_k	0.25	0.219	0.2		

Figure 1: First 8 shrinkg factors.

has $\ell+1$ segments of size 1 and ℓ segments of size α. Again, by Observation 3, we have $(\ell+\ell\alpha)/[1+\ell(1+\alpha)] = \alpha$. Interestingly it follows that we are solving *the same* equation:

$$\ell\alpha^2 + \alpha - \ell = 0$$

Thus again we have: $\alpha = (\sqrt{4\ell^2+1}-1)/2\ell$.

After showing that the partition ratio for either cases are the same. Now we analyze the shrinking factors for both cases. For odd number $k = 4\ell-3$, the length of intervals $[a, b]$ is shrunk from 2ℓ segments of size 1 and $2\ell-1$ segments of size α into an interval of size $\alpha+1$. That is, the shrinking factor

$$\rho_k = \rho_{4\ell-3} = \frac{1+\alpha}{(2\ell-1)(1+\alpha)+1}$$

For odd number $k = 4\ell-1$, the length of intervals $[a, b]$ is shrunk from $2\ell+1$ segments of size 1 and 2ℓ segments of size α into an interval of size $\alpha+1$. That is, the shrinking factor

$$\rho_k = \rho_{4\ell-1} = \frac{1+\alpha}{2\ell(1+\alpha)+1}$$

However these two equations are can be written as the following single condensed equation

$$\rho_k = \frac{1+\alpha}{\lceil k/2\rceil(1+\alpha)+1}$$

This completes the proof. □

3 Conclusion and remarks

In this paper, we show interesting results about parallel Fibonacci search. It is interesting to note that, when $k = 2$, the shrinking factor $\rho_2 = 1/2$ which makes the 2-Fibonacci search essentially a binary search. What's more interesting is that, when $k = 3$, the shrinking factor $\rho_3 = \phi/(1+2\phi) = 1-\rho_1 = 1-1/\phi$, which again is related to the golden ratio.

Here we list a table of first 8 shrinking factors in Figure . Observe that the shrinking ratio decreases as k gets larger, i.e., we discard more interval. Actually, we can show that ρ_k or $\rho(k)$ is a *strictly decreasing* function as k increases.

Further, our analysis assumes that, for any given k, the initial state of the interval $[a, b]$ is always partitioned into $[a, x]$ and $[x, b]$ such that $(x-a)/(b-x) = \alpha$. In reality, most certainly only the endpoints $[a, b]$ is given. In such case, the best strategy will be to partition the interval into $k+1$ *equal-sized* segments by the principle of balance. For even k, that will immediately goes into the recursive construction of the even case. However, for odd $k = 2\ell-1$, the next phase will then to partition the two equal sized intervals, one with equal-sized ℓ segments and the other with $\ell+1$ segments; note that these segments are equal-sized because of Observation 3 with the case $\alpha = 1$. The situation remains the same until it finds that the optimal value lies between two consecutive segments which are not equal-sized (in which case, $\alpha = \ell/(\ell+1)$) we can then apply Observation 3 before goes into the next phase.

In summary, the partition ratio α shown in Theorem 2 can be considered as the approaching value as the above phases repeated many times. That is, the shrinking ratio derived in Theorem 2 and illustrated in Figure can be viewed as the *upper bound*; in reality, the solution space might shrink faster than the value estimated in Theorem 2.

Further, we can show that $\rho_k = 2/k + o(1/k^2)$ or $\lim_{k\to\infty} \rho_k/(2/k) = 1$.

References

[1] M. Bazaraa and H. Sherali and C. Setty. Nonlinear programming, theory and algorithms, 2nd ed, John Wiley & Sons, 1993.

[2] E. Krotkov. Focusing, International Journal of Computer Vision, Vol. 1, pp. 223-237, 1987.