# **UC Santa Cruz**

**UC Santa Cruz Previously Published Works** 

# Title

Fraction Interpolation Walking a Farey Tree

**Permalink** https://escholarship.org/uc/item/1w38m4z4

Author Garcia-Luna-Aceves, J.J.

Publication Date 2006

Peer reviewed

# Fraction interpolation walking a Farey tree

Marc Mosko<sup>a,\*</sup>, J.J. Garcia-Luna-Aceves<sup>a,b</sup>

<sup>a</sup> Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304

<sup>b</sup> Computer Engineering Dept., University of California Santa Cruz, CA 95014

#### Abstract

We present an algorithm to find a proper fraction in simplest reduced terms between two reduced proper fractions. A proper fraction is a rational number m/n with m < n and n > 1. A fraction m/n is simpler than p/q if  $m \le p$  and  $n \le q$ , with at least one inequality strict. The algorithm operates by walking a Farey tree in maximum steps down each branch. Through monte carlo simulation, we find that the present algorithm finds a simpler interpolation of two fractions than using the Euclidean-Convergent [1] walk of a Farey tree and terminating at the first fraction satisfying the bound. Analysis shows that the new algorithms, with very high probability, will find an interpolation that is simpler than at least one of the bounds, and thus take less storage space than at least one of the bounds.

Key words: algorithms, graph algorithms, Farey

## 1 Introduction

In a system based on fractional representation of real numbers [1], it may at times be necessary to interpolate two bounds. The Split Label Routing Protocol (SLRP) [2], for instance, computes the mediant of two fractions to find new node labels in a topological order. One could average the bounds or compute a mediant [3, p. 14]. Both of these techniques would result in larger magnitude numerators and denominators, which in an implementation of a communications protocol could result in field overflow.

<sup>\*</sup> Corresponding author.

*Email addresses:* marc.mosko@parc.com (Marc Mosko), jjgla@parc.com (J.J. Garcia-Luna-Aceves).

#### 2 Fraction Interpolation

The algorithms presented here focus on a method derived from the continued fraction representation of a rational number. From number theory [3, Theorem 14], it is known that there exists an exact representation of any rational number as a finite continued fraction. If two rational numbers in a fractional math system are interpolated via average or mediant, one may reduce the result by finding the continued fraction representation. The mediant of two fractions m/n and r/s is (m+r)/(n+s). Based on this observation, we find a new method with better performance (simpler results) that is not tied to continued fractions.

The problem considered is to find a proper fraction p/q between two other extended proper fractions, m/n and r/s, which may include 0/1 and 1/1. For the purposes of a system with a fixed storage size, we would like to keep the denominator magnitude small. A fraction m/n is simpler than p/q if  $n \leq q$  and  $m \leq p$ , with at least one inequality strict [4, p. 166]. The methods described could be used on an extended Farey tree [5] over all rational numbers.

A Farey series  $\mathcal{F}_n$  is the set of all reduced proper fractions with a maximum denominator of n [6, p. 118]. A Farey series may be constructed beginning with 0/1 and 1/1 and then recursively inserting the mediant of adjacent fractions when the mediant denominator is of order n or less. A Farey tree [4,5] is the organization of proper fractions with 0/1 and 1/1 in an ancestry tree. Each proper fraction has two parents, the *simplest parent* (dotted line in our figures) and the *closest parent* (solid line in our figures) [4] (Lagarias and Tresser [5] call them *old* and *young* parents, respectively). The parents combine via the mediant to form a single child. The fraction 1/2, for instance, is the mediant of 0/1 and 1/1. The simplest (old) parent is 0/1 and the closest (young) parent is 1/1. The fraction 3/7 has simplest parent 1/2 and closest parent 2/5.

A child is always adjacent to the closest parent. A reduced fraction m/n is adjacent to p/q if and only if |mq - pn| = 1 [4, p.166]. Because of the special structure of the Farey tree, one may walk up and down the tree [6, pp. 119f] [5] [4]. The walk is done using Farey triangles [4], where the vertices are the parents and the child. A simple walk on the tree for fraction m/n could have very high time complexity, on the order of O(n). There are ways to thread the tree and take short cuts to improve the average case performance. The worst case performance, however, remains O(n).

Matula and Kornstrup [1, p. 102] describe the Euclidean-Convergent (EC) algorithm. For a given rational p/q, each step *i* of the EC algorithm finds the  $i^{th}$  convergent  $A_i$  and the rational approximation  $p_i/q_i$  for  $A_i$ . Algorithm 1 presents the EC algorithm, following [1]. Because p/q is a proper fraction, we

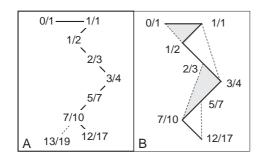
		Table 1							
Algorithm 1:		EC algorithm reducing $24/34$							
$\mathrm{EC}(p/q)$									
(1)	$\tilde{b}_{-1} = q; p_{-1} = 1; q_{-1} = 0$		i	$b_i$	$p_i$	$q_i$	$a_i$	$p_i/q_i$	
(2)	$b_0 = p; p_0 = 0; q_0 = 1$		-1	34	1	0	_	_	
(3)	i = 1								
(4)	repeat		0	24	0	1	0	0	
(5)	$a_i = \lfloor b_{i-2}/b_{i-1} \rfloor$		1	10	1	1	1	1	
(6)	$b_i = b_{i-2} - a_i \ b_{i-1}$								
(7)	$p_i = a_i \ p_{i-1} + p_{i-2}$		2	4	2	3	2	0.66667	
(8)	$q_i = a_i \ q_{i-1} + q_{i-2}$		3	2	5	7	2	0.71429	
(9)	i = i + 1								
(10)	<b>until</b> $b_i = 0$		4	0	12	17	2	0.70588	
(11)	$\mathbf{return}  p_i/q_i$								

skip the first iteration of the algorithm in [1], which computes  $a_0$ . It is always zero for a proper fraction.

Our problem statement does not require exact reduction of the mediant of m/n and r/s. Any fraction in the bound will do. For example, if we wish to find 1/3 < p/q < 23/31, one option is the mediant 24/34, which we can reduce to 12/17. Another acceptable answer is 2/3. This observation leads us to the EC Simplest middle (EC-SM) algorithm. In Alg. 1, we can replace the condition in line 10 to terminate on the first  $p_i/q_i$  that satisfies the bounds.

Our Short Cut (SC) algorithm, Alg. 2, does not find convergents, only rational approximations  $p_i/q_i$ . It does this by finding the maximum Farey triangle along a branch of the tree. We use it as the basis for the SC Simplest Middle algorithm. The SC algorithm first computes the mediant p/q, then reduces it to p'/q'. SC begins with a guess of p'/q' = 1/2. If p'/q' < p/q, it descends right down the Farey tree to the least fraction  $x/y \ge p/q$ . In the case of equality, SC returns x/y. SC then iterates with the guess p'/q' = x/y, and descends to the left. There is a similar procedure if the initial p'/q' > p/q. Compared to the behavior of EC, SC makes a better approximation of p/q at each iteration. When used in the SC-SM algorithm, it returns simpler fractions than EC-SM.

Algorithm 2 interpolates m/n and p/q given m/n < r/s. A symmetric solution exists if r/s < m/n. Fig. 2 illustrates variable labels for a descend-left scenario; a descend-right situation has a symmetric development. To find the reduction of p/q, which we take as the mediant of m/n and p/q, we begin with a guess of x/y = 1/2. Node x/y in a Farey tree has simplest parent  $x_s/y_s$  and closest parent  $x_c/y_c$ . Fig. 2(A) shows the closest parent as the upper-right parent (URP) and Fig. 2(B) shows it as the upper-left parent (ULP). If x/y = p/q, we are done. If x/y > p/q, then we must descend left from x/y, otherwise we descend right.



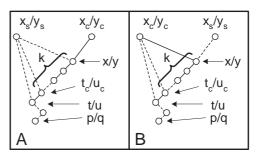


Fig. 1. EC walk for 12/17

Fig. 2. SC variables for descend-left

In the descend-left case, there is some greatest node  $t/u \leq p/q$  along the left descent from x/y. If p/q lies on the left descent, then we have the equality case. Otherwise, t/u will be strictly less than p/q. Once we find t/u, we make that our next guess for x/y and iterate the search.

The core of the SC algorithm is to solve for k, in the descend-left case, the simultaneous integer equations

$$\frac{t}{u} = \frac{x+k x_{ulp}}{y+k y_{ulp}} \le \frac{p}{q} \tag{1}$$

In the case of descend-left, we want to find the minimum k such that  $t/u \leq p/q$ ; for descend-right, we want to find the minimum k such that  $t/u \geq p/q$ . Rearranging terms to solve for k in terms of the inequality gives

$$qx + k \ qx_{ulp} \le py + k \ py_{ulp} \tag{2}$$

$$\lfloor (px - qy)/(py_{ulp} - qx_{ulp}) \rfloor \le k.$$
(3)

In Eq. 3, we only have equality when Eq. 1 exactly equals the reduction of p/q. Otherwise, k is strictly greater than the left side of Eq. 3. In Alg. 2, we use this fact to compute the closest parent  $t_c/u_c$ . If it happens to equal p/q, we terminate with  $t/u = t_c/u_c$ . Otherwise, we compute  $t/u = (t_c+x_{ulp})/(u_c+y_{ulp})$ .

To adapt SC to find the simplest proper fraction p/q between m/n < r/s, we must change the termination computation for SC. The issue is that SC descends as far down a branch as possible to find the juncture point of the next turn. For example, in Fig. 1(A), the mediant of 1/2 and 12/17 is 13/19. The SC algorithm would begin with x/y = 1/2 and step to x/y = 3/4. The simplest proper fraction between 1/2 and 12/17 is 2/3, not 3/4.

Because each proper fraction occurs exactly once in the Farey tree, which is built on closest parent paths, there is a unique path to each fraction. This implies that there is a specific first descent where one or more fractions t/u Algorithm 2: Algorithm 3: SC-SM(m/n, r/s)SC(m/n, r/s) $x/y = 1/2; x_c/y_c = 1/1;$ Initialize x/y,  $x_c/y_c$ ,  $x_s/y_x$ , t/u, (1)(1) $x_s/y_s = 0/1$ p/qx'/y' = x/y; type = none; (2)t/u = 1/2;(2)p/q = (m+r)/(n+s)(3)while  $mu \ge nt \lor tq \ge pu$ while  $tq - up \neq 0$ (3)Save previous iteration's (4)if  $x_s/y_s < x_c/y_c$ values. (4)x'/y' = x/y.(5) $x_{ulp}/y_{ulp} = x_s/y_s;$ (5)Compute  $x_{ulp}/y_{ulp}$  as before.  $x_{urp}/y_{urp} = x_c/y_c$ (6)else if x/y > p/q(6)(7)type = left(7) $x_{ulp}/y_{ulp} = x_c/y_c;$ (8) $x'_{par}/y'_{par} = x_{ulp}/y_{ulp}$  $x_{urp}/y_{urp} = x_s/y_s$ (9)(8)if x/y > p/q(10). . . descend-left case (9)(11)else  $k = \left| \left( px - qy \right) / \left( py_s - qx_s \right) \right|$ (12)type = right(10) $t/u = (x + kx_{ulp})/(y + ky_{ulp})$  $x'_{par}/y'_{par} = x_{urp}/y_{urp}$ (11)(13) $x_c/y_c = t/u$ (12)(14)(13) $x/y = (t + x_{ulp})/(u + y_{ulp})$ (15)Compute minimum t/u along (14) $x_s/y_s = (x - x_c)/(y - y_c)$ descent. k' = 0(15)else (16)if type = left(16)descend-right case (17)k' =Same as descend-left, but (18)(17) $\left|(sx'-ry')/(ry'_{par}-sx'_{par})\right|$ use  $x_{urp}/y_{urp}$ . else if type = rightreturn t/u(19)(18)(20)k' = $\lfloor (my' - nx')/(nx'_{par} - my'_{par}) \rfloor$  $t'/u' = (x' + k'x'_{par})/(y' + k'y'_{par})$ (21)(22)return t'/u'

along that descent satisfies m/n < t/u < r/s. The simplest such t/u along that descent is the simplest rational interpolation.

The SC-SM algorithm proceeds as Alg. 2, except we change the condition of line 3 to be  $\neg(m/n < t/u < p/q)$  and make an additional computation to find the simplest t/u along the descent that terminated the while loop. Alg. 3 shows the changes to Alg. 2.

The final computation of SC-SM directly solves m/n < p/q < r/s. As in the SC algorithm, consider a descent-left case as shown in Fig. 2. SC-SM finds the greatest t/u along the left descent that satisfies m/n < t/u < r/s.

$$\frac{m}{n} < \frac{x + kx_{ulp}}{y + ky_{ulp}} < \frac{r}{s} \tag{4}$$

Re-arranging terms yields the pair of inequalities

$$m(y + ky_{ulp}) < n(x + kx_{ulp}) \tag{5}$$

$$s(x + kx_{ulp}) < r(y + ky_{ulp}).$$
(6)

For the descend-left case, we use the bound from Eq. 6 (Alg. 3, l. 18). For a descend-right case, we use the bound from Eq. 5 with the URP (Alg. 3, l. 20).

An important question is the denominator magnitude for p/q found between m/n and r/s. For our case of proper fractions, the numerator is always less than the denominator. Formulations such as the mediant will always increase the magnitude of the denominator because q = n + s. Methods such as EC and SC, which find an exact reduction of the mediant p/q, may still grow the denominator magnitude when p/q is irreducible. From number theory [7, Theorem 18.6], we know that the probability that two integers are coprime is  $6/\pi^2 \approx 61\%$  in the limit for large numbers. Thus, it is common that interpolation by reduction of the mediant results in larger magnitude fractions because no such reduction is possible. For Simplest Middle methods (EC-SM and SC-SM), one is no longer constrained by finding an exact reduction. The only time the magnitude of p/q may increase from its bounds is when m/n and r/s share no common ancestor.

Not sharing a common ancestor is a necessary condition to increase the denominator magnitude in Simplest Middle methods. A sufficient condition is that m/n and r/s are adjacent in the Farey tree. If m/n and r/s share a common ancestor, then the simplest p/q between m/n and r/s is that ancestor, which is simpler than both m/n and p/q. If m/n and r/s do not have a common ancestor (i.e. one is an ancestor of the other), but they are non-adjacent, then the simplest p/q between m/n and r/s will be one of those Farey tree elements that lie between m/n and r/s. p/q will be simpler than one of m/nor r/s, but not simpler than both of them. Only in the case when m/n and r/s are adjacent will the simplest fraction p/q between them not be simpler than either. In such a case p/q must be the mediant.

The probability that m/n and r/s are adjacent is much less than the probability that p and q are coprime. In a full binary tree of height k, there are  $v = 2^{k+1} - 1$  nodes and v - 1 edges (the root does not have an ancestor edge). Each edge is equivalent to an adjacent pair of nodes. The total number of unique node pairings is  $\binom{v}{2} = (v)(v-1)/2$ . The probability that a uniform choosing of two distinct nodes results in a pair of adjacent nodes is  $2 \cdot (v-1)/(v)(v-1)) = 2/(2^{k+1}-1)$ . This approaches zero for large k. The simplest middle method, therefore, with high probability will find a p/qsimpler than at least one of the bounds.

Table 2 summarizes the comparison of the algorithms via Monte Carlo simulation of 100,000 pairs of unsigned 16-bit fractions and unsigned 31-bit fractions. We use 31-bit integers because the algorithms require a signed operation. Ran-

	avg bits	max bits	ties	wins	avg steps	max steps
input	14.8	16	_	_	_	_
Mediant	17.33	18	0	0	1	1
EC	16.35	18	2	0	10.0	22
SC	16.35	18	2	0	8.8	10
EC-SM	2.17	12	63042	0	1.7	8
SC-SM	1.79	10	63042	36958	1.2	8
input	28.5	30	_	_	_	_
Mediant	30.3	31	0	0	1	1
EC	29.5	31	0	0	18.7	32
SC	29.5	31	0	0	18.2	32
EC-SM	2.18	12	62770	0	2.74	12
SC-SM	1.79	12	62770	37230	1.74	11

Table 2Interpolation algorithm performance

dom generation of the input set only resulted in at most 30-bit numbers. The *input* data set is the set of random fractions used for input to the algorithms. The *Mediant* data set computes the mediant of each fraction pair without any reductions. The *EC*, *SC*, *EC-SM* and *SC-SM* data sets represent each of the four algorithms. The *avgbits* column shows the average number of bits required to store the denominator of the algorithm output. The *maxbits* column is the maximum number of storage bits over all denominators. The *ties* column is the number of times the algorithm tied for best performance (simplest fraction output). The *wins* column is the number of times that the algorithm output the best (simplest) result. The column *avg steps* is the average number of iterations of each algorithm. The Mediant algorithm does not iterate, so we denote it as exactly 1.

For 16-bit integer fractions, the SC-SM algorithm has the best average and maximum bit performance. 63% of the time it outputs the same solution as EC-SM, but the remaining 37% of the time it outputs a simpler solution. EC-SM and SC-SM output the same results 63% of the time. The EC and SC algorithms also tied for best twice, when the reduced fraction equals 1/2. SC-SM output a simpler result the remaining 37% of the time. In terms of the number of steps required by each algorithm, the Mediant, which does not iterate, always finds a solution in one step. Of the iterative algorithms, SC-SM requires the fewest iterations. SC-SM and EC-SM have very low average iterations because for randomly chosen fractions, it is common for one to be below 1/2 and one to be above 1/2, in which case SC-SM and EC-SM terminate immediately with the answer 1/2 in one step. Analyzing the input file, about 1/3 of the cases have both fractions below 1/2, about 1/3 of the

cases have both fractions above 1/2, and the remaining 1/3 of cases have mixed above and below 1/2 (which only take 1 iteration to solve in simplest middle schemes).

For the 30-bit input file, the results are largely the same. The average storage size did not change. The distribution of above and below 1/2 fractions remained at 1/3. The maximum storage size for SC-SM increased a few bits, but its size is still under 40% the size of the mediant.

## 3 Conclusion

We present several algorithms to find an interpolation of two fractions. The algorithm Short Cut - Simplest Middle (SC-SM) has the best performance, in terms of keeping the denominator magnitude small. Monte Carlo simulation shows that SC-SM requires less storage size and takes fewer steps (except for the mediant) to interpolate fractions than other methods studied. Analysis shows that methods that rely on an exact reduction of the mediant p/q increase the storage size around 61% of the time when p and q are coprime. Methods that use the Simplest Middle method will, with very high probability, reduce the storage size compared to at least one of the bound m/n or r/s because the only case when p/q is not simpler is when m/n and r/s are adjacent.

#### References

- D. W. Matula, P. Kornerup, Foundations of finite precision rational arithmetic, Computing Supplementum 2 (1980) 85–111.
- [2] M. Mosko, J. J. Garcia-Luna-Aceves, Loop-free routing using a dense label set in wireless networks, in: Proc. ICDCS, 2004.
- [3] A. Y. Khinchin, Continued Fractions, 3rd Edition, Dover Publications, Mineola, NY, 1997.
- [4] D. W. Matula, L. D. McFearin, A  $p \times p$  bit fractional model of binary floating point divison and extremal rouding cases, Theoretical Computer Science 291 (2) (2003) 159–82.
- [5] J. C. Lagarias, C. P. Tresser, A walk along the branches of the extended farey tree, IBM Journal of Research and Development 39 (3) (1995) 281–368.
- [6] R. L. Graham, D. E. Knuth, O. Patashnik, Concrete Mathematics, 2nd Edition, Addison-Wesley, Reading, MA, 1989.
- [7] G. H. Hardy, E. M. Wright, An Introduction to the Theory of Numbers, 5th Edition, Oxford University Press, Oxford, England, 1979.