

# Re: Surrogate factoring, corrected algorithm

**Source:** <http://sci.tech-archive.net/Archive/sci.math/2005-02/6411.html>

---

**From:** Bryan Olson ([nameless\\_at\\_nowhere.org](mailto:nameless_at_nowhere.org))

**Date:** 02/17/05

Date: Thu, 17 Feb 2005 04:52:09 GMT

Tim Peters wrote:

> Sorry, it didn't work for me. I started this time with my running  
example,  
>  $M = 112554401 * 221667653$  (I like that one because it's small enough so that  
> the  $T$ 's are easy to factor quickly by other simple methods, but its smallest  
> factor is large enough that luck has scant chance of succeeding quickly).  
>  
> The two previous versions of this algorithm didn't factor that  $M$  at any  $j$  in  
> 1 thru 623, but succeeded at  $j=624$ .  
>  
> This version didn't factor that  $M$  at any  $j$  in 1 thru 16, but did succeed  
> once at  $j=17$ . In all, 62,320,128 gcds were tried through  $j=17$ , with 1  
> success.  
>  
>  $j=17$  was painfully expensive. Then  $T^3 j^2 =$   
>  
>  $2^9 3^6 5^3 7^3 11^3 17^2 191^3 7841^3 53633^3 56197^3$   
>  $335417^3 14832053^3$   
>  
> and so there are  
>  
>  $10 * 7 * 4 * 4 * 4 * 3 * 4 * 4 * 4 * 4 * 4 * 4 / 2 = 27,525,120$   
>  
> distinct ways to split  $T^3 j^2$  as the product  $f_1 f_2$  with  $f_1$  and  $f_2$  both  
> integers  $\geq 1$ , and without regard to order; and so twice as many as that =  
> 55,050,240 gcds tried at  $j=17$  alone. In comparison, all of  $j=1$  through  $j=16$   
> computed a grand total of 7,269,888 gcds (with no success).

With the "corrected" algorithm's suggested  $j = \text{floor}(M/2) (+1 \text{ if even})$ ,  
I get a mere 24960 gcd trials. None of them work.

> Of course the most troublesome bit is that factors still aren't found at  
> some  $j$ , so at least one of {proof, algorithm, implementation} is wrong.

I never saw a proof of the claim that one of the GCD's must be a proper  
factor of  $M$ .

sci.math: Re: Surrogate factoring, corrected algorithm

- > *Implementation note: it's easiest to generate all  $\langle f_1, f_2 \rangle$  splittings*
- > *without trying to weed out reversals. For example, if  $\langle 3, 5 \rangle$  is generated,*
- > *also generate  $\langle 5, 3 \rangle$ . But then trying to weed out reversals later, by*
- > *keeping track of the ones already seen, can require an enormous amount of*
- > *memory,*

How about requiring  $f_1 \leq f_2$ ? Just build  $f_1$ 's and toss any that get larger than the square-root of the product.

--  
--Bryan