# 20

# Primer3 on the WWW for General Users and for Biologist Programmers

**Steve Rozen and Helen Skaletsky**

## 1. Introduction

Designing PCR and sequencing primers are essential activities for molecular biologists around the world. This chapter assumes acquaintance with the principles and practice of PCR, as outlined in, for example, **refs. *1–4***.

*Primer3* is a computer program that suggests PCR primers for a variety of applications, for example to create STSs (sequence tagged sites) for radiation hybrid mapping ***(5)***, or to amplify sequences for single nucleotide polymorphism discovery ***(6)***. *Primer3* can also select single primers for sequencing reactions and can design oligonucleotide hybridization probes.

In selecting oligos for primers or hybridization probes, *Primer3* can consider many factors. These include oligo melting temperature, length, GC content, 3′ stability, estimated secondary structure, the likelihood of annealing to or amplifying undesirable sequences (for example interspersed repeats), the likelihood of primer–dimer formation between two copies of the same primer, and the accuracy of the source sequence. In the design of primer pairs *Primer3* can consider product size and melting temperature, the likelihood of primer–dimer formation between the two primers in the pair, the difference between primer melting temperatures, and primer location relative to particular regions of interest or to be avoided.

### *1.1.* Primer3 *Can Be Used Through its WWW Interface or as a Software Component*

Most casual users will prefer *Primer3*'s WWW interface (**Fig. 1**), which is suitable for selecting primers from a few sequences. **Subheading 2.** discusses this interface in detail.

From: *Methods in Molecular Biology, vol. 132: Bioinformatics Methods and Protocols*
Edited by: S. Misener and S. A. Krawetz © Humana Press Inc., Totowa, NJ

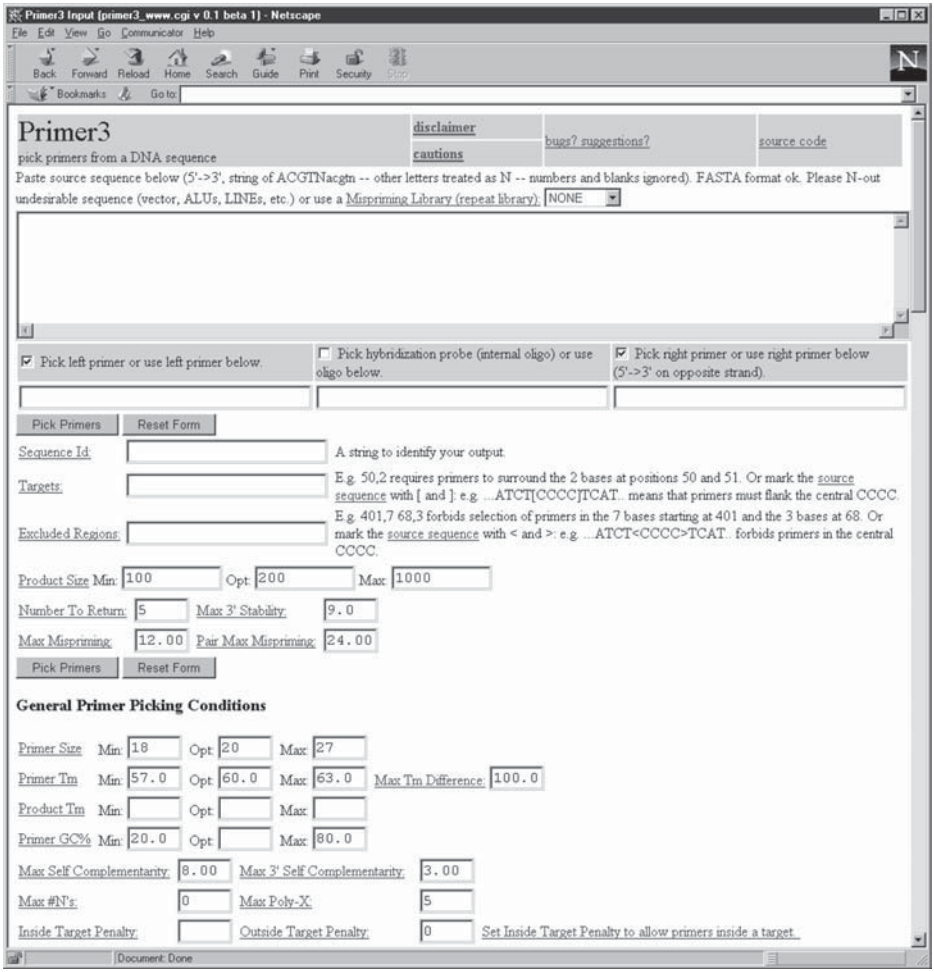



Fig. 1. Top of *Primer3*'s WWW input page without user input.

Scientists who must select primers for hundreds or thousands of sequences will prefer to use *Primer3* (specifically the *primer3_core* program) as a software component, which accepts input in a format convenient for other programs to produce and generates output in a format convenient for other programs to interpret. (We assume that no one would want to deal manually with primer picking results for hundreds or thousands of sequences.) We present examples of the use of *primer3_core* as a software component in **Subheading 3.** The underlying primer design process is identical for both the WWW interface and *primer3_core*, and in fact the WWW interface uses the WWW CGI protocol ***(9,10)*** layered on top of *primer3_core*.

In a few cases it will be preferable to modify *primer3_core* itself rather than simply use it as a software component. Therefore, for maximum portability and modifiability we wrote it in standard ANSI C ***(7)*** using standard POSIX calls ***(8)*** with simple and universally usable ASCII input and output. Furthermore, the distribution includes a thorough set of tests for *primer3_core*, which make it relatively easy to ensure that modifications do not introduce errors.

Although the WWW interface is adequately self-explanatory for many casual users, for others the background information we present here will be helpful. For potential high-volume users, customizers, and biologist programmers, this chapter introduces the use of *primer3_core* to streamline the particular primer-picking tasks at hand.

### *1.2. Where to Find* Primer3

Public WWW interfaces for use by anyone with a Web browser (for example *Netscape*) are reachable from `http://www.genome.wwi.mit.edu/cgi-bin/primer/info.cgi`. You can also download the *Primer3* programs from this location. The program *primer3_core* is available in source form only, and generating an executable program requires a C compiler; *see* **Subheading 3.1.** for details. The source code for the WWW interface is also available and can be used on computers running a Web server. The WWW interface (like *primer3_core*) can be modified to meet the needs of particular sets of end-users. It is written in *perl*, the language of choice for this sort of application ***(11)***.

### *1.3. What* Primer3 *Does not Offer*

We regret that we do not have resources to distribute *Primer3* in ready-to-run executable form, with "native" front ends (e.g., for Microsoft Windows or Mac), or on tape, diskette, or CD. Other primer selection software is available in fully supported commercial form (though possibly not as a customizable software component). Examples include *OLIGO*®, available through Molecular Biology Insights, Inc., Cascade, Colorado, `http://www.mbinsights.com/`, (*see* also **ref. *12***), *DNA*Star's *PrimerSelect* module for *LaserGene* (`http://www.dnastar.com/`), and the *Prime* module in Genetics Computer Group's Wisconsin Package. Primer selection programs available from academic institutions include *Primer 0.5* (upon which *Primer3* was based, but available as a stand-alone program and as a ready-to-run executable for Macs and PCs) ***(13)***, and *OSP–oligonucleotide selection program* ***(14)***.

The following tasks are *not* built in to *Primer3*:

- Automatically adding standard 5′ tails to each primer.
- Selecting nested primer pairs.

- Selecting primer pairs for multiplex amplification.
- Designing a tiling of amplicons across a sequence.
- Picking primers from a reverse-translated amino acid sequence.

(However, we have used *primer3_core* as a software component in conjunction with other codes to accomplish each of these tasks but the last.) The packages mentioned above perform some of these tasks.

### 1.4. PCR and Primer Design Applications Are Diverse

Primer design is really many different problems. Sometimes one wishes to design primers for a large number of sequences, and if for some reason it is difficult to find good primers for a particular source sequence one simply discards that source sequence. An example would be high-throughput whole genome mapping (the application for which *Primer3* and its predecessors were originally designed). In this case one designs STSs from tens of thousands of sequences and then uses these STSs in hundreds of amplifications. In this application no one sequence is particularly valuable compared to the cost of primers and subsequent amplifications, so it is not worth proceeding with a sequence for which there are only dubious primer pairs.

In other applications one wishes to design primers to amplify a *particular* sequence if at all possible; if there are no obvious good primers one will choose several possibilities in the hope that at least one will work. Examples include designing primers to distinguish two very similar sequences or to amplify a particular exon flanked by a CpG island in which it is hard to find a good primer. In this situation the precious resource is the particular sequence to amplify, and the scientist is willing to spend considerable effort getting a clean amplicon.

There are many other variables in primer-design goals. Sometimes one wants large amplicons (for example to amplify as much of a cDNA as possible), and sometimes one wants very short amplicons (for example to flank a single nucleotide polymorphism as closely as possible). Sometimes the amplification template is complex (for example a mammalian genome), and sometimes it is simple (for example a single bacterial artificial chromosome). Some *Taq* formulations are less likely than others to produce primer dimers or self-priming hairpins.

Because primer design is really many different problems *Primer3* gives users numerous options to specify which primers are acceptable and which primers are better than others. The number of these options can be overwhelming to new and experienced users alike, but typically for any particular application only a few need changing from default values.

This chapter cannot discuss all of *Primer3*'s options, but it covers those you are most likely to change. The WWW interface and the **README** distributed with the program document the more esoteric options.

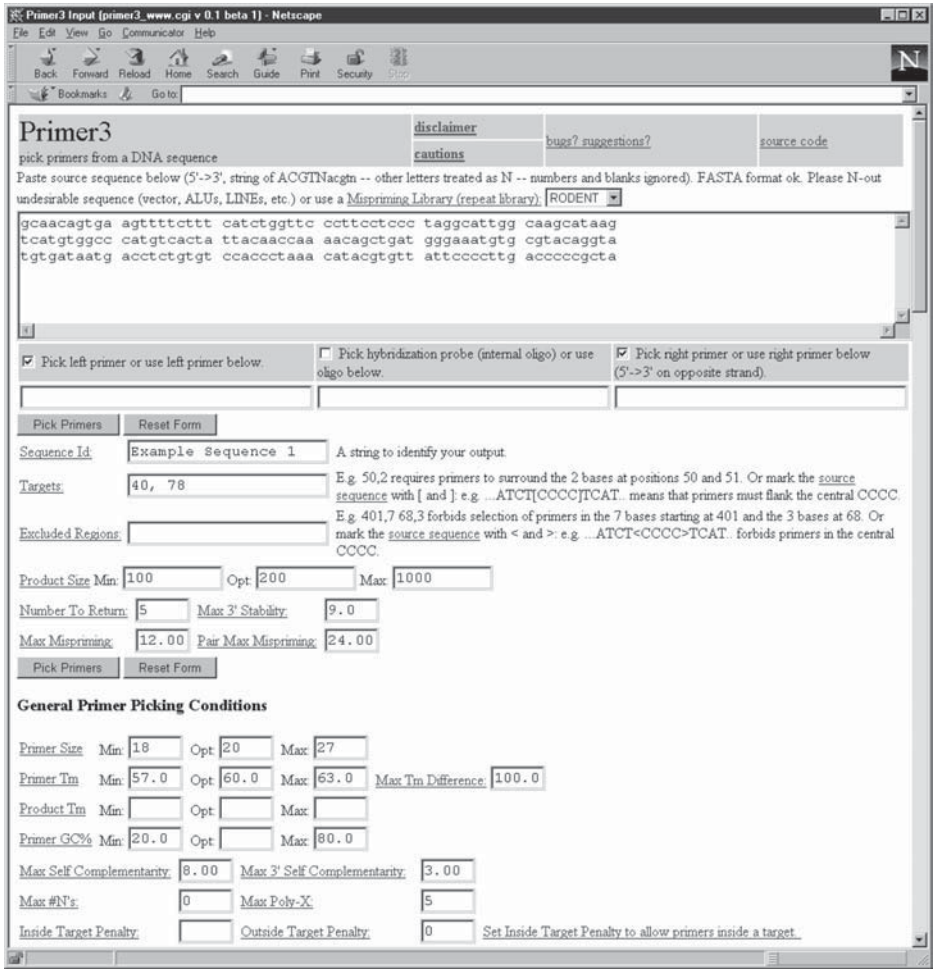



Fig. 2. *Primer3*'s WWW input page after the user has entered sequence, a "Sequence Id," and a "Target."

## 2. *Primer3* from the End-User Perspective

This section refers primarily to the WWW interface, which calls upon *primer3_core* to perform almost all of the work of selecting primers.

*Primer3* takes as input a sequence and selects single primers or PCR primer pairs. **Figure 2** shows example input in the WWW interface. The user has pasted the source sequence into the large data-entry field near the top of the page, selected the **RODENT Mispriming Library** and entered a **Sequence Id** ("Example Sequence 1") and a **Target** ("40, 78").

Below the field for the source sequence are three check boxes labeled **Pick left or use left primer below**, **Pick hybridization probe (internal oligo) or use oligo below**, and **Pick right primer or use primer below. . . .** These check boxes govern whether *Primer3* tries to design a primer pair, a primer pair plus hybridization probe, or an individual primer (e.g., for sequencing) or hybridization probe. In **Fig. 2** the **Pick left...** and **Pick right...** boxes are checked, so *Primer3* will select PCR primer pairs.

Below each of these three check boxes is an input box. Placing an oligo in one of these boxes instructs *Primer3* to evaluate that oligo and choose matching oligos (depending on the check boxes).

After the user clicks on any of the **Pick Primers** buttons in the input page (**Fig. 2**) *Primer3* returns suggested primers as shown in **Fig. 3**. **Subheading 2.2.** discusses the interpretation of this output in detail; **Subheading 2.3.** suggests some strategies for proceeding when *Primer3* is unable to find any acceptable primers or primer pairs.

The label for each input option is a link to documentation on the meaning of the option and how *Primer3* uses it. For example, clicking on **Max End Stability** takes one to the following documentation:

**Max End Stability**

The maximum stability for the five 3' bases of a left or right primer. Bigger numbers mean more stable 3' ends. The value is the maximum delta G for duplex disruption for the five 3' bases as calculated using the nearest neighbor parameters published in Breslauer, Frank, Bloeker and Marky, Proc. Natl. Acad. Sci. USA, vol 83, pp 3746-3750. Rychlik recommends a maximum value of 9 (Wojciech Rychlik, "Selection of Primers for Polymerase Chain Reaction" in BA White, Ed., "Methods in Molecular Biology, Vol. 15: PCR Protocols: Current Methods and Applications", 1993, pp 31-40, Humana Press, Totowa NJ).

The **Max Mispriming** and **Pair Max Mispriming** input fields are important in many situations because the source sequence might contain one of the interspersed repeats (ALUs, LINEs, and others) that make up more than 35% of the human genome ***(15)***. The user should either replace these sequences by Ns before picking primers or should select a **Mispriming library**. (Not all mispriming is strictly speaking caused by repeats; it could be any sequence that one does not wish to inadvertently amplify.) However, the WWW interface at the `www.genome.wi.mit.edu` web site only offers repeat libraries for human and for mouse and rat (**RODENT**).

The maximum primer length is restricted because the nearest neighbor melting temperature model agrees well with reality only for relatively short sequences ***(16,17)***.

### *2.1. How* Primer3 *Picks Primers*

*Primer3* accepts many options that specify which primers are acceptable and which primers are better than others. In the WWW interface the user selects

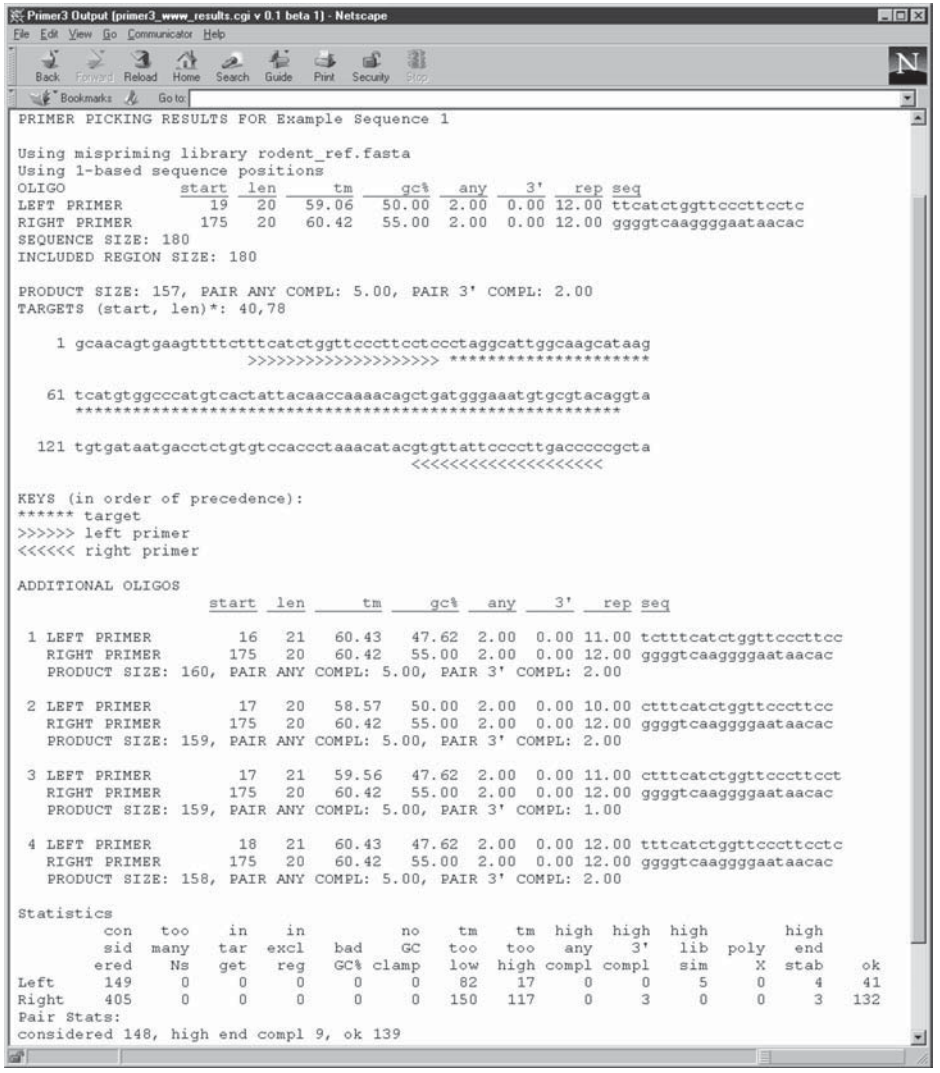



Fig. 3. Output from *Primer3's* WWW interface when primers have been found.

these options through text boxes, check boxes, and pull-down menus. For example in **Fig. 2** these include the **Mispriming Library** pull down (above the sequence input field), **Product Size Min**, **Opt**, and **Max** input fields, and all the input fields beneath the **General Primer Picking Conditions** heading.

Some options specify which primers are *acceptable*. For example to the left of **Product Length** the **Min** and **Max** options set lower and upper bounds on the length of products. Such options are called *constraints* because they

constrain the set of acceptable primer pairs. Other options that specify constraints include **Primer Tm Min** and **Max**, **Max End Stability**, and **Max Mispriming**. (Tm is an abbreviation for "melting temperature".)

Other options specify characteristics of the optimum output primers or primer pairs (beyond specifying those that are simply acceptable). Examples include the **Product Length Opt** (Optimum) and the **Primer Tm Opt** inputs. Roughly speaking, if **Product Length Opt** is specified *Primer3* tries to pick a pair of primers that produce an amplicon of approximately the specified length. For some options the user need not specify the optimum value because *Primer3* can take it for granted; for example there is no input field for the optimum mispriming library similarity, which *Primer3* assumes to be 0.

*Primer3* examines all primer pairs that satisfy the constraints and finds pairs that are closest to the optimum. How does *Primer3* calculate how close a primer pair is to the optimum? By default the WWW interface tries to balance equally primer length, primer melting temperature, and product length. (For compatibility with earlier versions *primer3_core* by default uses only primer length and primer melting temperature.)

However, to accommodate the diversity of primer picking applications *Primer3* is flexible in the formula it uses to calculate how close a primer or primer pair is to the optimum. The technical term for this formula is *objective function*. Thus, suppose you deem the difference in melting temperature between the two primers to be more important than their lengths, melting temperatures, and the product size. Then you can use the **Objective Function Weights...** sections of the input page (as partially shown in **Fig. 4**) to tell *Primer3* to use these considerations in calculating optimality. In **Fig. 5** this effect is accomplished by the values in the fields labeled **Product Size Lt** and **Gt**, **Tm Difference**, and **Primer Penalty Weight**. (**Primer Penalty Weight** is an adjustment factor for the entire per-oligo contribution to the objective function. More details are available in the online documentation.)

### 2.2. Interpreting the Output when Primers are Found

Please refer to **Fig. 3**. The top of the output displays the sequence id (**Example Sequence 1**) and a number of informational notes. The next part of the output displays the best left and right primers, and their characteristics (starting position, length, melting temperature, and so forth). Then the output displays information specific to the input sequence and the selected pair.

The next information is a quasi-graphical representation of the location of the left (>>>>> ...) and right (<<<<< ...) primers in the source sequence, as well as any important features of the source sequence, in this example only the position of the target (marked by asterisks***** ...). Following the sequence is information for some number of additional primer pairs. (The user can control

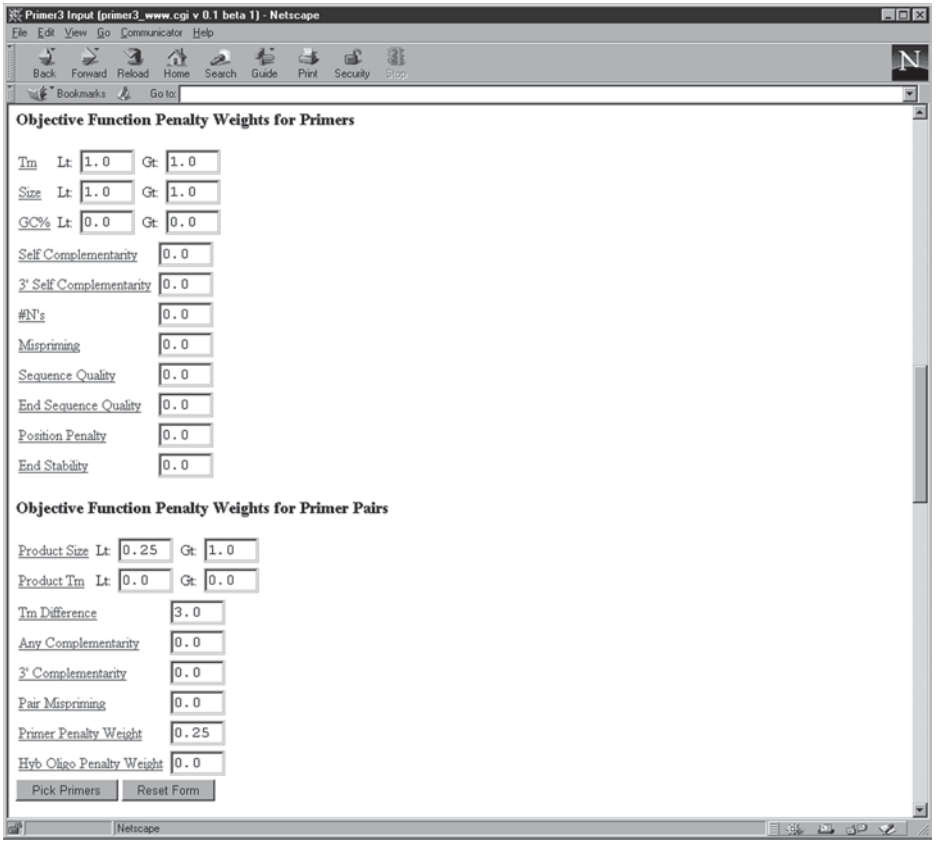



Fig. 4. The part of *Primer3*'s WWW interface that allows modification of the objective function. In this example "Product Size Lt" and "Gt," "Tm Difference," and "Primer Penatly Weight" have been changed from the defaults.

the number returned by entering a different value in the **Number to Return** input field.)

Finally the output contains a section headed **Statistics**, which we discuss in detail below.

### 2.3. What if There Are no Acceptable Primers?

Recall from **Subheading 1.3.** that in some situations one wants only good primers for uniform conditions and would rather discard some source sequence than deal with dubious primers. Most of *Primer3*'s default option values are tuned to suit these situations: constraints are strict. Given strict constraints the specifics of the objective function are not critical because any acceptable

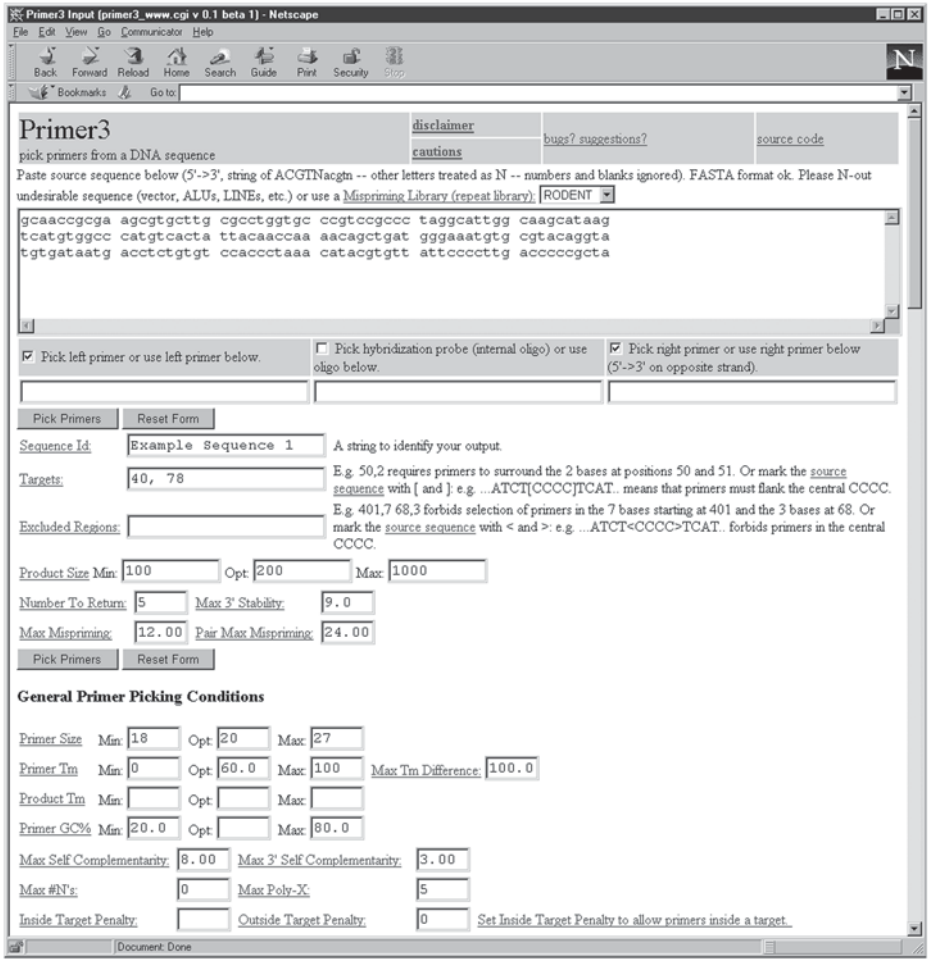



Fig. 5. *Primer3*'s WWW interface in which the "Primer Tm Min" and "Primer Tm Max" constraints have been made completely liberal.

primer or primer pair will be reasonably good, but potentially usable primers might be rejected as unacceptable.

In other situations, however, one must design primers for a given sequence at almost any cost. Suppose you are faced with such a situation, and *Primer3* cannot find acceptable primers given the default constraints. In this case *Primer3* will return a screen similar to that shown in **Fig. 6**.

Then what? The most intuitive course is to relax the constraints that you think are least important in your particular situation and that are most likely preventing any primers or primer pairs from being acceptable. The **Statistics**

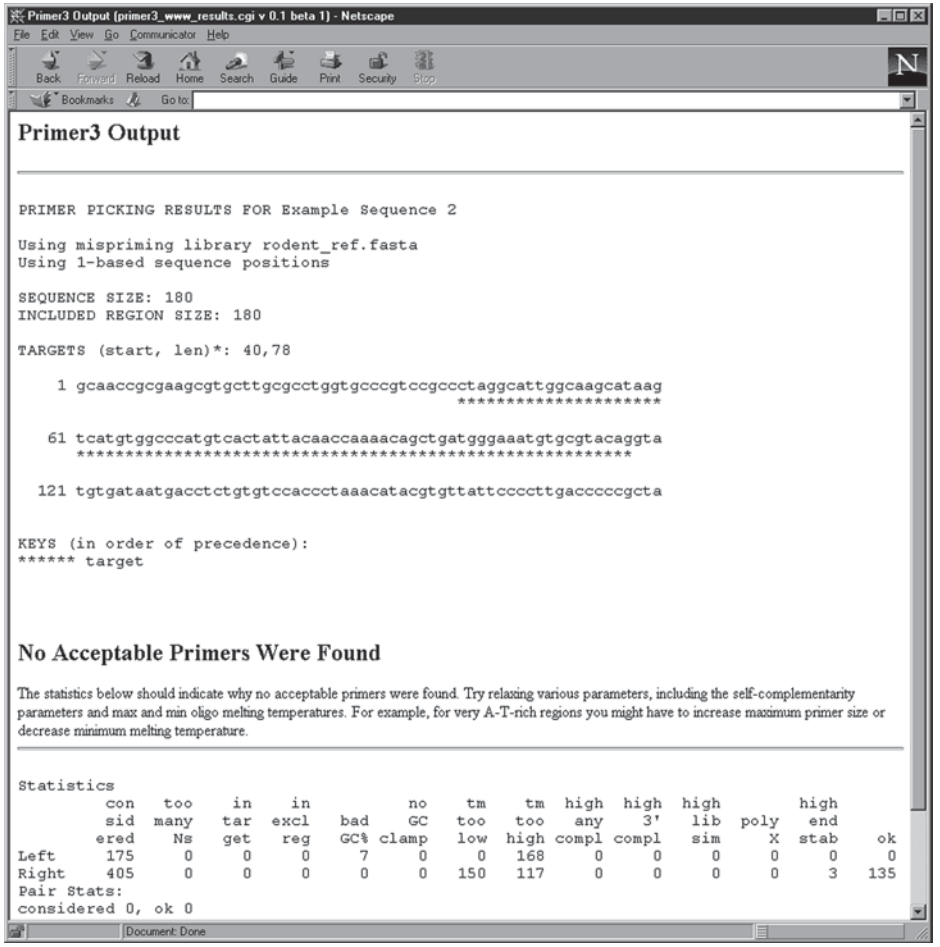



Fig. 6. Output from *Primer3*'s WWW interface when no primers were found.

section at the bottom of the output indicates reasons that individual primers are unacceptable. In the example above it is clear that the main problem is that all acceptable left primers have too high a melting temperature (as indicated by the column headed **tm too high**).

A word of caution: *Primer3* never considers a primer that is unacceptable because of its position. Thus, if a primer is outside of the included region or can never be acceptable given the length of the sequence, the position of any specified "included region" and targets and the range of allowable product sizes then it is *not* counted in the **considered** column. If it seems as though very few primers are even being considered, you might want to modify your maximum

and minimum product size options, or expand the included region. An example of such a situation is the following, in which no left primers are considered:

```
Statistics
         con   too    in    in             no    tm    tm  high  high        high
         sid  many   tar  excl   bad    GC   too   too   any    3'  poly   end
        ered    Ns   get   reg   GC% clamp   low  high compl compl    X  stab    ok
Left       0     0     0     0     0     0     0     0     0     0     0     0     0
Right   1401    25     0     0     9     0   884   191     0     0     0     0   292
```

The **Pair Stats:** section indicates reasons that pairs of primers (as opposed to single primers or oligos) are rejected. For example, there might only be a few acceptable primers, all of which when paired would create a product with too low or too high a melting temperature.

Examining the **Statistics** (and less commonly) the **Pair Stats:** sections should suggest constraints that if relaxed would allow primers to be chosen.

In some cases (especially when relaxing several constraints at once) it might be desirable to also modify the objective function to reflect specific primer design objectives. In the sequence in **Fig. 6**, the temperatures of all possible left primers are too high. One way to proceed is to incrementally relax what seem to be the limiting constraints, for example increasing the **Primer Tm Max** option until an acceptable left primer is found. Alternatively, it can be more expeditious to simply relax the limiting constraint totally, as in **Fig. 5**, in which the **Primer Tm Min** and **Max** are set to 0 and 100° C, respectively. The primer pair selected with these relaxed constraints is:

```
OLIGO            start  len      tm      gc%   any    3'    rep seq
LEFT PRIMER         10   19   68.72    63.16  6.00  1.00 10.00 aagcgtgcttgcgcctggt
RIGHT PRIMER       175   20   60.42    55.00  2.00  0.00 12.00 ggggtcaaggggaataacac
```

And the Statistics are

```
         con   too    in    in             no    tm    tm  high  high  high        high
         sid  many   tar  excl   bad    GC   too   too   any    3'   lib  poly   end
        ered    Ns   get   reg   GC% clamp   low  high compl compl   sim     X  stab    ok
Left      53     0     0     0     7     0     0     0     0     9     0     0    16    21
Right    387     0     0     0     0     0     0     0     0     4     0     0     5   378
```

Now there are acceptable primer pairs but a large difference in melting temperatures between the left and right primers. To reduce this difference one can include it as part of the objective function, as shown in **Fig. 4**. After this adjustment, *Primer3* selects the following primer pair:

```
OLIGO            start  len      tm      gc%   any    3'    rep seq
LEFT PRIMER         11   18   67.83    66.67  4.00  1.00 10.00 agcgtgcttgcgcctggt
RIGHT PRIMER       180   20   66.95    60.00  2.00  2.00 10.00 tagcgggggtcaaggggaat
```

## 3. *Primer3* for Biologist Programmers

### *3.1. Installation Instructions*

The source distribution is available as a UNIX "tar" archive, which can be managed by the UNIX tar utility, by the Windows / Windows NT *WinZip* utility (Nico Mak Computing; `http://www.winzip.com/winzip.htm`) or by the Mac *DropStuff with Expander Enhancer* utility `www.aladdinsys.com`. To run *primer3_core* you will first need to compile it using an ANSI C compiler with *POSIX* libraries and run the tests supplied with the distribution as documented in the **README**.

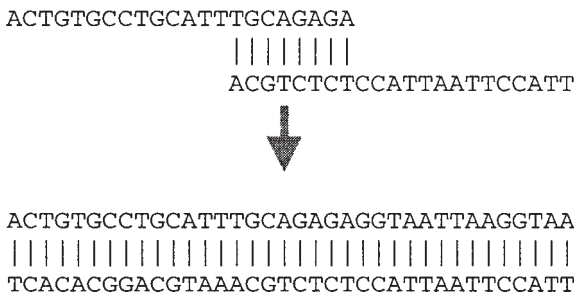
### *3.2. Examples of How to Use* primer3_core *as a Software Component*

In this section we present two examples of using *primer3_core* as a software component. The code for these examples is available in the *Primer3* distribution.

#### *3.2.1. Using* primer3_core *with UNIX Pipes*

The first example is a relatively lightweight application of the kind that requires only a minimum of *perl* scripting experience.

This example shows how to postprocess *primer3_core*'s output to complete an oligo design task. The example task is the specification of "overgos" (John D. McPherson, pers. comm.), in which a 36-mer double-stranded hybridization probe is constructed from annealing overlapping 22-mers and filling in the singled stranded tails:




Specifically, we will show code that takes an existing primer pair and then designs an overgo that will hybridize to the site amplified by that primer pair. Here is the UNIX command line one would use:

```
prompt> ./primer3_core < input | ./overgo.pl
```

In this command line *primer3_core* runs first, taking its input from the file "input", whereas its output is sent directly to the *perl* program *overgo.pl* via a

UNIX pipe (specified by the vertical bar, '|', on the command line). The input could be prepared by hand in a text editor or (more likely) produced by another program. It has the form of *tag*=value pairs, a format dubbed *Boulder-IO* ***(18)***:

```
PRIMER_SEQUENCE_ID=Overgo Example
PRIMER_PICK_INTERNAL_OLIGO=1
PRIMER_INTERNAL_OLIGO_MAX_MISHYB=36
PRIMER_INTERNAL_OLIGO_MIN_SIZE=36
PRIMER_INTERNAL_OLIGO_MAX_SIZE=36
PRIMER_INTERNAL_OLIGO_OPT_SIZE=36
PRIMER_INTERNAL_OLIGO_MIN_TM=10
PRIMER_INTERNAL_OLIGO_MAX_TM=90
PRIMER_INTERNAL_OLIGO_OPT_TM=70
PRIMER_INTERNAL_OLIGO_SELF_ANY=30
PRIMER_INTERNAL_OLIGO_SELF_END=30
PRIMER_INTERNAL_OLIGO_MISHYB_LIBRARY=humrep
PRIMER_PRODUCT_SIZE_RANGE=70-1000
PRIMER_EXPLAIN_FLAG=1
PRIMER_PAIR_WT_IO_QUALITY=1
PRIMER_PAIR_WT_PR_QUALITY=0
PRIMER_IO_WT_REP_SIM=1
PRIMER_IO_WT_TM_GT=0
PRIMER_IO_WT_TM_LT=0
PRIMER_IO_WT_SIZE_GT=0
PRIMER_IO_WT_SIZE_LT=0
PRIMER_NUM_RETURN=1
PRIMER_LEFT_INPUT=GAAATGTGTCCTTCCCCAGA
PRIMER_RIGHT_INPUT=GAGTTCACCCATACGACCTCA
SEQUENCE=GGATCACAACGTTTTTTGACACACCCTATAATGATGTATT . . .
=
```

*Boulder-IO* is a format for moving semistructured data between programs. *Primer3* receives its input and (by default) produces its output in a simple subset of *Boulder-IO*. The **README** in the *Primer3* distribution describes the meanings of all these *tag*=value pairs in the input, as well as those in the output. The output from *primer3_core* given the input above is:

```
PRIMER_SEQUENCE_ID=Overgo Example
PRIMER_PICK_INTERNAL_OLIGO=1
PRIMER_INTERNAL_OLIGO_MAX_MISHYB=36
```

```
PRIMER_INTERNAL_OLIGO_MIN_SIZE=36
PRIMER_INTERNAL_OLIGO_MAX_SIZE=36
PRIMER_INTERNAL_OLIGO_OPT_SIZE=36
PRIMER_INTERNAL_OLIGO_MIN_TM=10
PRIMER_INTERNAL_OLIGO_MAX_TM=90
PRIMER_INTERNAL_OLIGO_OPT_TM=70
PRIMER_INTERNAL_OLIGO_SELF_ANY=30
PRIMER_INTERNAL_OLIGO_SELF_END=30
PRIMER_INTERNAL_OLIGO_MISHYB_LIBRARY=humrep
PRIMER_PRODUCT_SIZE_RANGE=70-1000
PRIMER_EXPLAIN_FLAG=1
PRIMER_PAIR_WT_IO_QUALITY=1
PRIMER_PAIR_WT_PR_QUALITY=0
PRIMER_IO_WT_REP_SIM=1
PRIMER_IO_WT_TM_GT=0
PRIMER_IO_WT_TM_LT=0
PRIMER_IO_WT_SIZE_GT=0
PRIMER_IO_WT_SIZE_LT=0
PRIMER_NUM_RETURN=1
PRIMER_LEFT_INPUT=GAAATGTGTCCTTCCCCAGA
PRIMER_RIGHT_INPUT=GAGTTCACCCATACGACCTCA
SEQUENCE=GGATCACAACGTTTTTTGACACACCCTATAATGATGTATT . . .
PRIMER_LEFT_EXPLAIN=considered 1, ok 1
PRIMER_RIGHT_EXPLAIN=considered 1, ok 1
PRIMER_INTERNAL_OLIGO_EXPLAIN=considered 224,
long poly-x seq 12, ok 212
PRIMER_PAIR_EXPLAIN=considered 1, ok 1
PRIMER_PAIR_QUALITY=15.0000
PRIMER_LEFT_SEQUENCE=GAAATGTGTCCTTCCCCAGA
PRIMER_RIGHT_SEQUENCE=GAGTTCACCCATACGACCTCA
PRIMER_INTERNAL_OLIGO_SEQUENCE=ACTGTGCCTGCATTTGCA . . .
PRIMER_LEFT=99,20
PRIMER_RIGHT=198,21
PRIMER_INTERNAL_OLIGO=140,36
PRIMER_LEFT_TM=59.903
PRIMER_RIGHT_TM=59.981
PRIMER_INTERNAL_OLIGO_TM=72.885
PRIMER_LEFT_SELF_ANY=3.00
PRIMER_RIGHT_SELF_ANY=4.00
PRIMER_INTERNAL_OLIGO_SELF_ANY=8.00
```

```
PRIMER_LEFT_SELF_END=0.00
PRIMER_RIGHT_SELF_END=1.00
PRIMER_INTERNAL_OLIGO_SELF_END=3.00
PRIMER_INTERNAL_OLIGO_MISHYB_SCORE=15.00, MLT1b
(MLT1b subfamily) - consensus sequence
PRIMER_LEFT_END_STABILITY=8.2000
PRIMER_RIGHT_END_STABILITY=8.2000
PRIMER_PAIR_COMPL_ANY=4.00
PRIMER_PAIR_COMPL_END=1.00
PRIMER_PRODUCT_SIZE=100
=
```

The second program, *overgo.pl*, takes the sequence of the 36-mer hybridization probe from this output and produces the overlapping 22-mers that constitute the overgo:

```
#!/usr/local/bin/perl5 -w

$/ = "\n=\n"; # Set the record terminator.

while (<>) {
  %rec = split /[=\n]/;  # A DANGEROUS approach
                         # to parsing the sequence.
  for (keys %rec) { $rec{$_} =~ s/\n// }
  $seq = $rec{'PRIMER_INTERNAL_OLIGO_SEQUENCE'};
  next unless $seq;
  print "MARKER\t\t$rec{'PRIMER_SEQUENCE_ID'}\n";
  $left  = substr($seq,0,22);          # Get left oligo.
  $r     = substr($seq, 14);           # Get the right
                                       # oligo,
  $right = reverse($r);                # reverse it, and
  $right =~ tr/GATC/CTAG/;             # complement it.
  print "LEFT_MID_OLIGO\t$left\n";
  print "RIGHT_MID_OLIGO\t$right\n";
  print "MAX_SCORE\t
  $rec{'PRIMER_INTERNAL_OLIGO_MISHYB_SCORE'}\n";
  $gc    = ($seq =~ tr/GC/GC/);
  printf "GC_content\t%d%%\n\n", $gc * 100 / 36;
}
```

The statement `$/ = "\n=\n";` tells *perl* that each record is terminated by an "=" sign on a line by itself (which is the standard record terminator for

*Boulder-IO*). The statement `%rec = split /[=\n]/;` parses *Boulder-IO* record into the perl hash `%rec`. This method of parsing the output requires that we know that "=" will not appear in the *value* part of any *Boulder-IO tag*=value pair. For situations in which more robustness is required, use Lincoln Stein's *perl Boulder* module (available at `http://www.genome.wi.mit.edu/genome_software/other/boulder.html`). Using this module *overgo.pl* would be rewritten as:

```
#!/usr/local/bin/perl5 -w

use Boulder::Stream;
$in = new Boulder::Stream;
while ($rec = $in->read_record()) {
  $seq
    = $rec->get('PRIMER_INTERNAL_OLIGO_SEQUENCE');
  next unless $seq;
  print "MARKER\t\t",
  $rec->get('PRIMER_SEQUENCE_ID'), "\n";
  $left  = substr($seq,0,22);      # Get the left
                                   # oligo.
  $r     = substr($seq, 14);       # Get the right
                                   # oligo,
  $right = reverse($r);            # reverse it, and
  $right =~ tr/GATC/CTAG/;         # complement it.
  print "LEFT_MID_OLIGO\t$left\n";
  print "RIGHT_MID_OLIGO\t$right\n";
  print "MAX_SCORE\t",
  $rec->get('PRIMER_INTERNAL_OLIGO_MISHYB_SCORE'),
  "\n";
  $gc    = ($seq =~ tr/GC/GC/);
  printf "GC_content\t%d%%\n\n", $gc * 100 / 36;
}
```

Using the *Boulder* module is preferable because it is more robust. It will run correctly even if someone puts an "=" in, for example, the value for **PRIMER_SEQUENCE_ID**. The only disadvantage is that you need to get the *Boulder* module before you can try it. The output for the input above is

```
MARKER              Overgo Example
LEFT_MID_OLIGO      ACTGTGCCTGCATTTGCAGAGA
RIGHT_MID_OLIGO     TTACCTTAATTACCTCTCTGCA
MAX_SCORE  15.00, MLT1b (MLT1b subfamily) . . .
  consensus sequence
```

### 3.2.2 *Calling* primer3_core *from* perl

The second example is *Primer3's* WWW interface itself. This code fragment is adapted from the CGI script, *primer3_www_results.cgi*, which implements part of that interface. (The CGI module is available from `http://www.genome.wi.mit.edu/ftp/distribution/software/WWW/`.) *Primer3_www.cgi* calls *primer3_core*, with a flag requesting formatted output (-format_output), and then grabs *primer3_core*'s output and tweaks it a bit:

```
#!/usr/local/bin/perl5 -w
...
use FileHandle; # Standard part of perl distribution
use IPC::Open3; # Standard part of perl distribution
use CGI;
    ...
    $query = new CGI;
                     # $query now contains the parameters
                     # to the cgi script
    ...
    my @names = $query->param;
    ...
    for (@names) {
       next if ... # Some cgi parameters do not get
                   # sent to primer3_core
       ...
       $line = "$_=$v\n";
       push @input, $line;  # Save a Boulder-IO line for
                            # primer3_core's eventual consumption.
    }
    my $cmd = "./primer3_core -format_output -strict_tags"
    my $primer3_pid;
    my ($childin, $childout) = (FileHandle->new, FileHandle->new);
    {
       local $^W = 0;
       $primer3_pid = open3($childin, $childout, $childout, $cmd);
    }
```

```
if (!$primer3_pid) {
       print "Cannot excecure $cmd:<br>$!\n$wrapup\n";
      exit;
    }
    print "<pre>\n";
    print $childin @input;
    $childin->close;
    my $cline;
    while ($cline = $childout->getline) {
      if ($cline =~ /(.*)start  len    tm   gc%  any   3\' seq/)
      {
        # Grap a particular line and
        # add hyperlinks to it:
        $cline = $1
               . "<a href=\"$DOC_URL#PRIMER_START\">start</a>"
               . "<a href=\"$DOC_URL#PRIMER_LEN\">len</a>"
               . "<a href=\"$DOC_URL#PRIMER_TM\">tm</a>"
               . "gc%   any    3\' seq\n"
      }
      print $cline;
    }
    print "</pre>\n";
    waitpid $primer3_pid, 0;
    if ($? != 0 && $? != 64512) { # 64512 == -4
        ... # primer3_core exited with
            # an error code; alert the browser.
    }
```

Of course the -formated_output flag in $cmd is not an essential part of the paradigm at work in this example. The script could have parsed *Boulder-IO* output and then formatted or processed the information in some other way.

### *3.2.3 Other Uses of* primer3_core *as a Software Component*

The two examples above show how to use *primer3_core* as a component in a lightweight Unix pipeline (the overgo design example) and how to use *perl*'s **open3** command to start an execution of *primer3_core* and then grab its output for further processing. An intermediate approach that is simpler to program

than using **open3** is to simply use the *perl* **open** command and then return *primer3_core*'s output unmodified, e.g.

```
if (!open(PRIMER,"| $cmd")) {
  print "Cannot execute <pre>$cmd\n</pre>\n$wrapup";
  return;
}
print PRIMER @input;
close PRIMER;   # primer3_core's output is the same
                # as this script's output.
if ($? != 0 && $? != 64512) { # 64512 == -4
    . . . # $cmd exited with an error code.
}
```

At the Whitehead institute we have used *primer3_core* as part of an industrial-strength primer design pipeline that includes vector clipping (identification and electronic "removal" of vector arms), microsatellite repeat identification, and automatic screening for vector contaminants. We have also used it in pipelines that add constant 5' tails to each primer and in pipelines that find a tiling of amplicons across a sequence. For this last application we set **PRIMER_FILE_FLAG=1** in *primer3_core*'s input, which directs *primer3_core* to create files containing all acceptable left and right primers. A different program then selects primers from these lists to produce the tiling.

### 3.3. Efficiency Considerations

The running time of *Primer3* is seldom an issue for users of the WWW interface. However, users of *primer3_core* for high volume applications should be aware of the factors that determine running time. The most expensive operation in selecting individual primers is a check against a mispriming or mishyb library (the actual time needed for each oligo is a linear function of the size of the library). The next most expensive operations are checks for oligo self-complementarity, and, if *Primer3* examines a large number of primer pairs, checking oligo pairs for self-complementarity.

*Primer3*'s running time depends also on the size of the sequence in which to select primers. Selecting a single primer pair anywhere within a 10-kb sequence will take approx 10 times as long as selecting a single primer pair anywhere within a 1-kb sequence (all other options being equal).

The following are additional determinants of *Primer3*'s running time:

- Strict as opposed to liberal constraints on oligos. *Primer3* excludes primers based on cheap computations (e.g., oligo melting temperature) before examining more

expensive-to-compute characteristics (e.g., similarity to mispriming library entries) so relaxing cheap-to-compute constraints entails evaluation of a larger number of expensive-to-compute constraints.

- Acceptable locations for primers (considering also constraints on product size). This item is similar to the preceding one. *Primer3* does not perform expensive operations to characterize primers which, because of their location, can never be part of an acceptable primer pair.
- The **PRIMER_FILE** input flag. This flag causes *Primer3* to compute every characteristic, including mispriming similarity and self-complementarity, of every possibly acceptable primer.
- Cost of computing the objective function. There are two subcases.
  - The objective function depends on expensive-to-compute characteristics of oligos or primers, such as similarity to mispriming or mishyb libraries or complementarity between primers in a pair. In this case *Primer3* must perform these expensive computations on essentially all acceptable primers.
  - The objective function depends on characteristics of primer pairs *per se*, such as product melting temperature or product size. In this case *Primer3* must calculate whether each individual primer is acceptable, which usually requires some expensive computation to determine acceptability.

(When the objective function depends neither on expensive characteristics of individual primers nor on characteristics of primer pairs then *Primer3* organizes its search so that it only checks expensive constraints on the best primers.)

## Acknowledgments

The development of *Primer3* and the *Primer3* WWW interface were funded by Howard Hughes Medical Institute and by the National Institutes of Health, National Human Genome Research Institute, under grants R01-HG00257 (to David C. Page) and P50-HG00098 (to Eric S. Lander).

We gratefully acknowledge the support of Digital Equipment Corporation, which provided the Alphas that we used for much of the development of *Primer3*, and of Centerline Software, Inc., whose TestCenter memory error, memory leak, and test coverage checker helped us discover and correct a number of otherwise latent errors in *Primer3*.

*Primer3* is the most recent of a number of primer-picking programs implemented at Whitehead Institute, starting with primer 0.5 ***(13)***. *Primer3* started as a reimplementation of *Primer 0.5* as software component; the design of *Primer3* draws heavily on the design of *Primer 0.5* and *Primer v2* (Richard Resnick) and the WWW interface designed by Richard Resnick for *Primer v2*.

Thanks to Alex Bortvin, Mark Daly, Nathan Siemers, and William J. Van Etten for reviewing drafts of this chapter.

## References


1. Dieffenbach, C. W. and Dveksler, G. S. (1995) *PCR Primer A Laboratory Manual*. Cold Spring Harbor Laboratory Press, Cold spring Harbor, NY.
2. Innis, M. A., Gelfand, D. H., Sninsky, J. J., and White, T. J., eds. (1990) *PCR Protocols A Guide to Methods and Applications*. Academic Press, San Diego, CA.
3. Rychlik, W. (1993) Selection of primers for polymerase chain reaction, in *Methods in Molecular Biology, vol. 15: PCR Protocols: Current Methods and Applications* (White, B. A., ed.) Humana, Totowa, NJ, pp. 31–40.
4. Wetmur, J. G. (1991) DNA probes: applications of the principles of nucleic acid hybridization. *Crit. Rev. Biochem. Mol. Biol.* **26,** 227–259.
5. Schuler, G. D. et al. (1996) A gene map of the human genome. *Science* **274,** 540–546.
6. Wang, D. G. et al. (1998) Large-scale identification, mapping and genotyping of single-nucleotide polymorphisms in human genome. *Science* **280,** 1077–1082.
7. Harbison, S. and Steele, G. (1995) *C A Reference Manual*, 4th ed. Prentice Hall, Englewood Cliffs, NJ.
8. Dougherty, D. (1991) *POSIX Programmer's Guide*. O'Reilly, Cambridge, MA.
9. Gundavaram, S. (1997) *CGI Programming with Perl*. O'Reilly, Cambridge MA.
10. Stein, L. D. (1997) *How to Set Up and Maintain a Web Site*, 2nd ed. Addison-Wesley, Reading, MA.
11. Wall, L., Christiansen, T., and Schwartz, R. L. (1996) *Programming Perl*, 2nd ed. O'Reilly, Cambridge, MA.
12. Rychlik, W. and Rhoads, R. E. (1989) A computer program for choosing optimal oligonucleotides for filter hybridization, sequencing and in vitro amplification of DNA. *Nucleic Acids Res.* **17,** 8543–8551.
13. Daly, M. J., Lincoln S. E., and Lander E. S. (1991). "*PRIMER*", unpublished software, Whitehead Institute/MIT Center for Genome Research. Available at `http://www.genome.wi.mit.edu/ftp/pub/software/primer.0.5`, and via anonymous ftp to `genome.wi.mit.edu, directory /pub/software/primer.0.5`.
14. Hillier, L. and Green, P. (1991) OSP: an oligonucleotide selection program. *PCR Meth. Appl.* **1,** 124–128. Documentation available at `http://genome.wustl.edu/gsc/manual/protocols/ospdocs.html`. OSP is available from the author on request.
15. Smit, A. F. A. (1996) The origin of interspersed repeats in the human genome. *Curr. Opin. Genet. Devel.* **6,** 743–748.
16. Breslauer, K. J., Frank, R., Bloeker, H., and Marky L. A. (1986) Predicting DNA duplex stability from the base sequence. *Proc. Natl. Acad. Sci. USA* **83,** 3746–3750
17. Rychlik, W., Spencer, W. J., and Rhoads, R. E. (1990) Optimization of the annealing temperature for DNA amplification in vitro. *Nucleic Acids Res.* **18,** 6409–6412.
18. Stein, L. (1997) How perl saved the human genome project. *Dr Dobb's Journal* Spring 1997 Special Report on Software Careers. Available at `http://www.ddj.com/ddj/1997/careers1/stei.htm`.