Ph.D Thesis
박사 학위논문

A MapReduce-based method for the thorough and rapid design of high-quality primers for qPCR experiments

Hyerin Kim(김 혜 린 金 慧 麟)

Department of Information and Communication Engineering

정보통신융합전공

DGIST

2016

Ph.D Thesis
박사 학위논문

A MapReduce-based method for the thorough and rapid design of high-quality primers for qPCR experiments

Hyerin Kim(김 혜 린 金 慧 麟)

Department of Information and Communication Engineering

정보통신융합전공

DGIST

2016

A MapReduce-based method for the thorough and rapid design of high-quality primers for qPCR experiments

Advisor : Professor Min-Soo Kim

Co-advisor : Professor JaeHyung Koo

By

Hyerin Kim
Department of
Information and Communication Engineering
DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of the requirements for the degree of Doctor of Philosophy, in the Department of Information and Communication Engineering. The study was conducted in accordance with Code of Research Ethics[1]

06. 02. 2016

Approved by

Professor Min-Soo Kim (Signature)
(Advisor)

Professor JaeHyung Koo (Signature)
(Co-Advisor)

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

A MapReduce-based method for the thorough and rapid design of high-quality primers for qPCR experiments

Hyerin Kim

Accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

06. 02. 2016

Head of Committee 구 재 형 (인)
Prof. JaeHyung Koo

Committee Member 김 민 수 (인) MinSoo Kim
Prof. Min-Soo Kim

Committee Member 제 민 규 (인)
Prof. Minkyu Je

Committee Member 최 지 환 (인)
Prof. Jihwan Choi

Committee Member 황 재 윤 (인)
Prof. Jae Youn Hwang

Ph.D/IC
201142001

김 혜 린. Hyerin Kim. A MapReduce-based method for the thorough and rapid design of high-quality primers for qPCR experiments. Department of Information and Communication Engineering. 2016. 89p. Advisors Prof. Kim, Min-Soo, Prof. Co-Advisors Koo, JaeHyung.

ABSTRACT

Primer design is a fundamental technique that is widely used for polymerase chain reaction (PCR). Although many methods have been proposed for primer design, they require a great deal of manual effort to generate feasible and valid primers, including homology tests on off-target sequences using BLAST-like tools. That approach is inconvenient for many target sequences of quantitative PCR (qPCR) due to considering the same stringent and allele-invariant constraints. In this dissertation, we propose an entirely new method that overcomes these drawbacks.

In the first part of this dissertation, we propose the method called MRPrimer that can design all feasible and valid primer pairs existing in a DNA database at once, while simultaneously checking a multitude of filtering constraints and validating primer specificity. Furthermore, MRPrimer suggests the best primer pair for each target sequence, based on a ranking method. Through qPCR analysis using 343 primer pairs and the corresponding sequencing and comparative analyses, we showed that the primer pairs designed by MRPrimer are very stable and effective for qPCR. In addition, MRPrimer is computationally efficient and scalable, and therefore useful for quickly constructing an entire collection of feasible and valid primers for frequently updated databases like RefSeq. Furthermore, we suggest that MRPrimer can be utilized conveniently for experiments requiring primer design, especially real-time qPCR.

Existing web servers for primer design have major drawbacks, including requiring the use of BLAST-like tools for homology tests, lack of support for ranking of primers, TaqMan probes, and simultaneous design of primers against multiple targets. Due to the large-scale computational overhead, the few web servers supporting homology tests use heuristic approaches or perform homology tests within a limited scope. The primer pairs designed by MRPrimer are very stable and effective in qPCR experiments. However, although MRPrimer can design very high-quality primers, routine use is inconvenient because it runs on a cluster of computers and requires several hours of runtime when the filtering constraints are adjusted.

In the second part of this dissertation, we propose MRPrimerW, the online version of MRPrimer, allows users to design the best primers quickly in a web interface, without requiring a MapReduce cluster or a long computation, as in Google's search system. It performs complete homology testing, supports batch design of primers for multi-target qPCR experiments, supports design of TaqMan probes, and ranks the resulting primers to return the top-1 best primers to the user. To ensure high accuracy, we adopted the core algorithm of MRPrimer, but completely redesigned it to allow users to receive query results quickly in a web interface, without requiring a MapReduce cluster or a long computation. MRPrimerW provides primer design services and a complete set of 341,963,135 in-silico validated primers covering 99% of human and mouse genes.

In summary, we have proposed a new method for primer design that overcomes most of drawbacks of existing methods. For an entire DNA database, we have proposed MRPrimer that can design all possible feasible and valid primer pairs through simultaneously checking a multitude of filtering constraints and validating primer specificity. For user

query from web interface, we have proposed MRPrimerW that performs complete homology tests, supports batch designing for qPCR, supports TaqMan probe design, and supports ranking of primers. We believe that the proposed methods will be contribute to increasing the efficiency and specificity of experiments involving PCR.

Keywords: MapReduce, primer design, qPCR, homology test

Contents

List of tables

List of figures

I. INTRODUCTION

1.1 Background

A primer is a short, single-stranded DNA molecule that serves as a starting point for DNA synthesis. DNA primers are widely used in many biological and medical laboratory techniques that involve DNA polymerase, such as DNA sequencing and polymerase chain reaction (PCR, Figure 1.1).

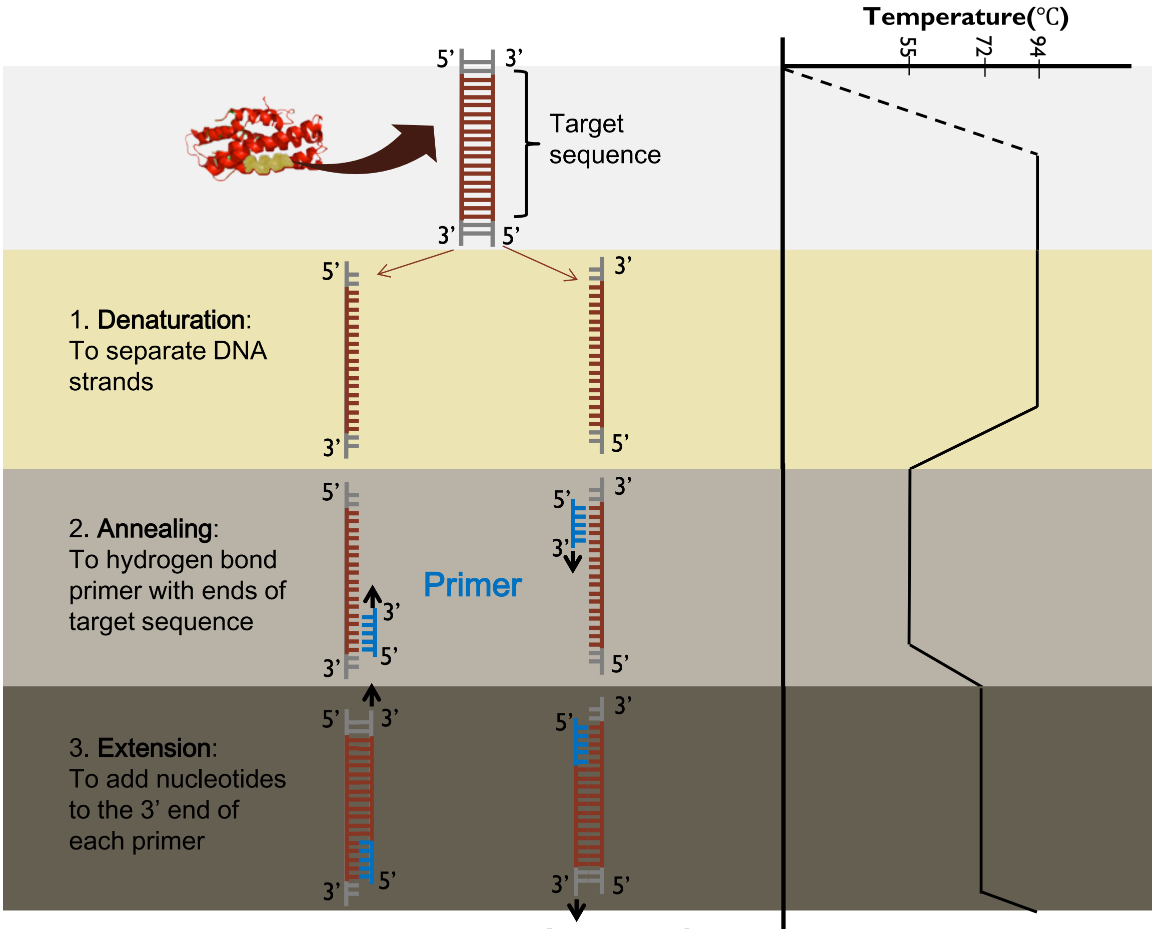


Figure 1.1. Polymerase chain reaction.

PCR was developed in 1983 by Kary Mullis, who was awarded the Nobel Prize in Chemistry in 1993. PCR amplifies a target sequence of DNA across several orders of magnitude, to generate thousands to millions of copies of the target sequences. Inputs are target sequences and a pairs of primers, will be explained later. Then output will be amplified target sequences. Figure 1.1 shows steps of PCR procedure; each cycle consists of three steps. First, heat up 94 degree to separate two strands of target sequence into one. Then cool down 55 degree to allow annealing of primers complementary to the target. Prime is a strand of nucleic acid that servers as starting point of PCR. If the primer matches with target, in the final step, primer can extend in 5' to 3' direction heating up 72 degree. Repeating the cycles, in cycle 2, get 4 molecules matched with target sequences. In the same manner, able to obtain 8 molecules in cycle 3. Eventually in cycle 35, 34 billion molecules can be observed (Figure 1.2).

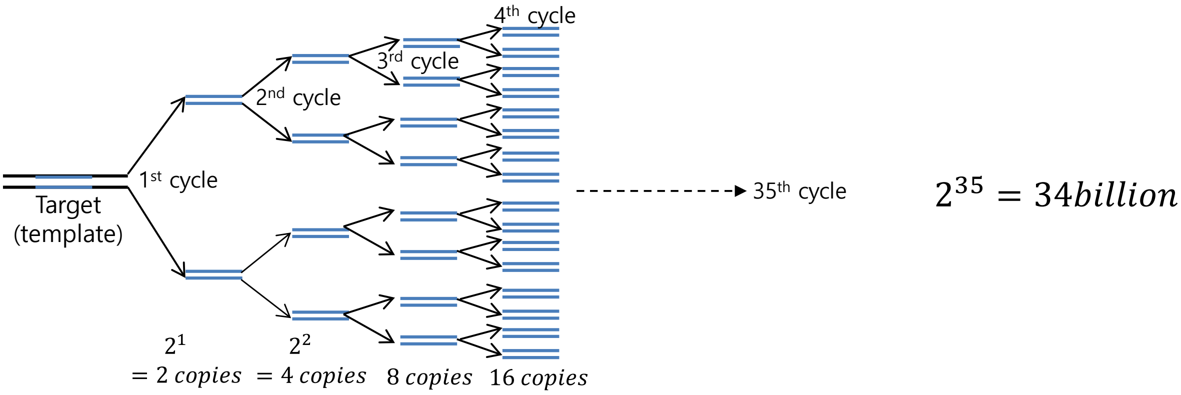


Figure 1.2. PCR amplification

As a standard laboratory technique for fast mass duplication of specific DNA sequences, PCR with suitable primers is used in a wide variety of applications, including phylogenetic analysis of DNA from different species to detect and identify unknown and distantly related gene sequences [1-3], genetic testing of DNA samples to detect the presence of disease-associated genetic mutations [4], the study of infectious diseases such as HIV and antibiotic-resistant microorganisms [4], PCR-based genetic fingerprinting and parental testing in forensics [4], DNA cloning [5], and microsatellite detection using molecular markers in population biology [6]. In addition, quantitative PCR (qPCR), also known as real-time PCR, has been widely used to confirm the results of high-throughput experiments by validating expression changes of selected genes [7]. The success of PCR-based experiments, including qPCR analysis, depends strongly on the design of suitable primers against the target sequence(s).

When designing primers, we must simultaneously consider many kinds of constraints, including primer length, melting temperature, GC content, self-complementarity, continuous residues, free-energy value, differences in length and melting temperature between members of primer pairs, product size, and pair-complementarity [8]. Table 1.1 shows the each constraints definitions and feasible range of values for general qPCR experiments. Manual design of primers is time-consuming and may easily yield incorrect results; therefore, automatic methods that can check the aforementioned single and pair filtering constraints are preferred [9]. Additional important constraints that should be evaluated are homology tests, i.e., whether the designed primers can only amplify the target sequence(s) rather than off-target sequences; such tests usually require an additional BLAST-like tool. Figure 1.3 figure shows the examples which a primer amplifies intended target or unintended target. All matches primer and intended targets is okay. Primer with 5’ end mismatch and identical rest of residues can be amplified unintended targets. Mismatch in middle of primer and target also can be amplified unintended targets. Lastly, mismatch on 3’ end cannot make annealing, so not consider. Fast automatic design of high-quality primers that satisfy both filtering constraints and homology tests remains a challenge that has not yet been completely solved.

Table 1.1 Constraints definition and feasible range of values.

	Parameter	Definitions	Value
Each primer	primer length	Primer Length	19~23 bp
	melting temperature (TM)	Primer melting temperature (nearest neighbor thermodynamic model)	60~63℃
	GC content	% of G and C	35~65%
	self-complementarity	Number of primer bases annealing to itself	< 5-mer
	3’ self-complementarity	Number of primer 3’ bases annealing to itself	< 4-mer
	Contiguous residue	Length of a mononucleotide repeat, for example AAAAAA	< 6-mer
	Gibbs free energy (ΔG)	Stability for the last five 3' bases of primer	≥ -9 *kcal/mol*
Primer pair	length difference	Length difference of pair primers	≤ 3-mer
	TM difference	Melting temperature difference of pair primers	≤ 5℃
	product size	Acceptable size of PCR product produced by reaction	100~250 bp
	pair-complementarity	Number of bases of forward primer to bind to the reverse primer, and vice versa	< 9-mer
	3’ pair-complementarity	Number of bases of the 3' end forward primer to bind to the 3' end reverse primer, and vice versa	< 4-mer

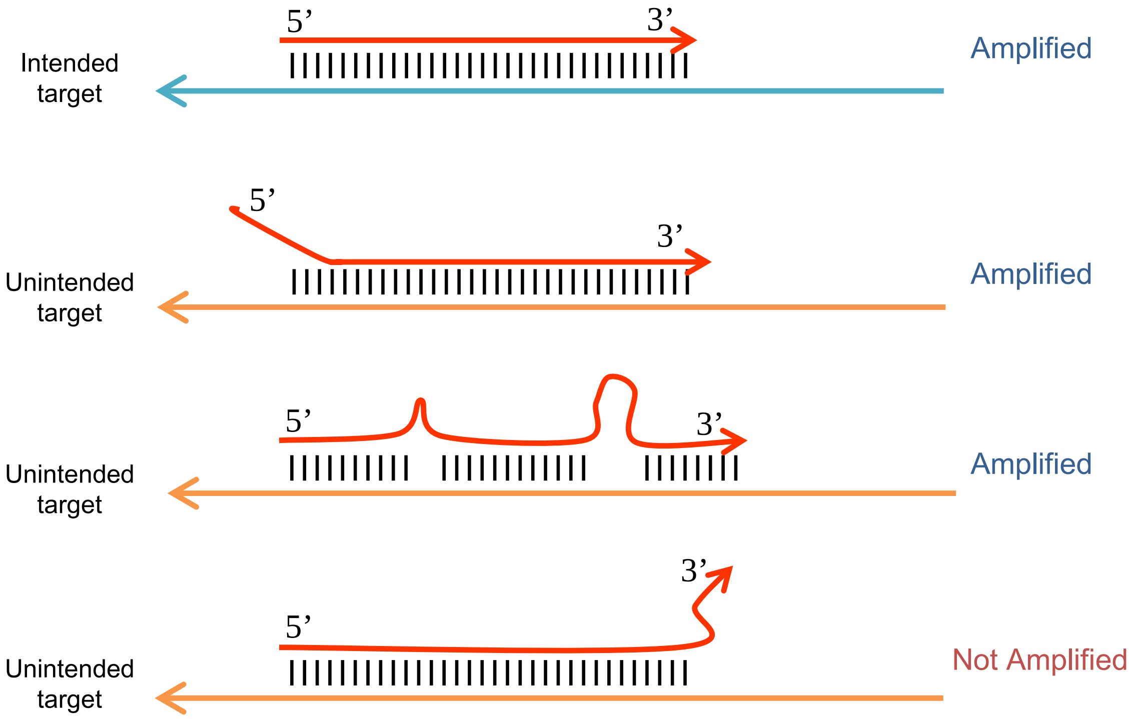


Figure 1.3. An examples which a primer amplifies intended target or unintended target.

Furthermore, if we want to design a large number of primers for qPCR in a short time that satisfy the same set of filtering constraints (e.g., similar product sizes), the problem becomes much more difficult. For qPCR experiments, in addition to the above SYBR Green primers, TaqMan probes are also commonly used to detect products, and they can significantly increase the specificity of detection; however, this requires extreme care in the design of both probes and primers to ensure they satisfy both the filtering constraints and the homology tests [10].

The existing methods have the following four fundamental problems or drawbacks.

First, the existing methods for a single target sequence do not support both specification of abundant filtering constraints and homology testing on off-target sequences. In terms of computation, it is a non-trivial problem to support both in a combined manner because this approach typically requires complex and large-scale join computation between a large number of candidate primer pairs designed from each target sequence, as well as a huge number of off-target sequences. Homology tests for every candidate primer against the entire sequence database requires 9 quintillion comparisons. Accordingly, users usually use a tool chain of both approaches with some human intervention, but such an approach cannot yield complete results.

Second, the existing methods for a single target sequence only focus on designing primers for a specific target sequence; therefore, they cannot be easily used for qPCR, which requires a large number of primer pairs to satisfy the same stringent and allele-invariant constraints (e.g., very similar product sizes) across target sequences. To alleviate this issue, PrimerBank [7, 8] was built and updated over the past several years; this database contains 248,578 primer pairs designed from 17,076 human and 18,086 mouse genes following similar constraints.

Third, existing methods cannot find all possible primers completely, especially for multiple target sequences. This deficiency is mainly due to the first step, i.e., multiple sequence alignment (MSA). The complexity of optimal MSA is inherently NP-complete [11], and so finding an optimal alignment is computationally infeasible for more than a few sequences. Most tools for MSA (e.g., CLUSTALW) [12] are heuristic; therefore, primers designed based on MSA results are also incomplete. Moreover, although we could compute the optimal MSA for a given set of sequences, it would be hard to find all possible primers only with a single fixed alignment because some primers might exist in conserved regions of non-optimal alignments. Methods not based on MSA, like HYDEN, are also heuristic, and therefore cannot find all primers. HYDEN also has the serious drawback that it cannot change primer constraints freely [13]. Overall, the existing methods tend to miss a large proportion of the feasible primers for given target sequences, even when such primers actually exist.

Fourth, the existing methods for multiple target sequences only focus on finding degenerate primers. Degenerate primer is a set of mixture of similar, but not identical primers. With a degenerate primer, multiple target sequences can be amplified. Degenerate primers inherently have a trade-off between degeneracy and coverage. In general, we cannot increase the degeneracy of primer to a high value, since using high degenerate primers would greatly reduce the specificity of the PCR amplification [14, 15]. In addition, there have recently been some studies saying that degenerate primers are not quite effective [16-18]. The degenerate primers could introduce a level of bias into the phylogenetic study, and so the profiles using them may not accurately represent the coverage of community [16]. The non-degenerate primers also could obtain the same quality of taxonomic coverage as the previously designed degenerate primers do [17]. Using non-degenerate primer, the PCR specificity has been further increased [19]. From those previous studies, we can say that for multiple target sequences, non-degenerate primers would be better than the degenerate ones, if both have the same coverage.

1.2 Motivation and objectives

In this dissertation, we propose an entirely new method called MRPrimer (http://MRPrimer.com) that overcomes most of the drawbacks of existing methods. For a given set of filtering constraints and a given sequence database (e.g., human gene DNA sequences), the proposed method designs all feasible primers that satisfy the constraints while validating their specificity in one sitting.

It finds not only all primers that can amplify a specific single target sequence, but also all primers that can amplify specific multiple target sequences. It neither relies on MSA nor heuristics; instead, it simply finds every possible non-degenerate primer, without missing any feasible or valid primer in the given sequence database, in a single execution.

Consequently, it can design a tremendous number of feasible and valid primer pairs, e.g., about 63 million pairs from human genes and 84 million pairs from mouse genes in the consensus coding sequence (CCDS) database (http://www.ncbi.nlm.nih.gov/CCDS/CcdsBrowse.cgi) and show very high coverage ratios, 95% for human and 96% for mouse, for the same database.

For realizing the above desirable features, MRPrimer follows a fairly complicated but parsimonious flow of computation based on the MapReduce framework [20]. The overall flow of MRPrimer is composed of a total of seven steps. Here, each step is a carefully designed MapReduce algorithm.

In addition to designing all feasible and valid primer pairs, while simultaneously checking a multitude of filtering constraints and validating primer specificity, MRPrimer suggests the best primer pair for each target sequence, based on a ranking method performed in its seventh step (i.e., the final step). Consequently, users only need to use the best primer pair(s) for target sequence(s) for their experiments.

In addition, the flow of MRPrimer is highly efficient and scalable in terms of computation, and so can construct a collection of all primer pairs corresponding to genome-scale data within a few hours using only a small cluster of computers. Therefore, MRPrimer is useful for quickly constructing an entire collection of feasible and valid primers for frequently updated databases like RefSeq.

We explained the MRPrimer method in more detail and showed its results in biological experiments. Although MRPrimer can design primers for multiple target sequences, in this paper we focus on qPCR experiments using primers for single target sequences. Especially, we demonstrated the results of qPCR analysis using 343 primer pairs and the corresponding sequencing and comparative analyses for validating the stability and effectiveness of MRPrimer for qPCR.

In addition, we describe a new web-based method, MRPrimerW (http://MRPrimerW.com), for batch design of primers for qPCR experiments. This tool checks filtering constraints, performs rigorous homology testing against a whole genome database, and ranks the resultant primer pairs according to their penalty scores to pick the best one for each target sequence. MRPrimerW supports the design of not only SYBR Green primers, but also TaqMap probes.

MRPrimerW is an online processing method based on our previously proposed offline processing method MRPrimer, which returns all feasible and valid primer pairs for a DNA database at once. MRPrimer performs a fairly complex, large-scale computation based on the MapReduce framework, resulting in design of very high-quality primers. Through qPCR analysis using 343 primer pairs and corresponding sequencing and comparative analyses, we showed that the primer pairs designed by MRPrimer are very stable and effective in qPCR experiments. However, although MRPrimer can design very high-quality primers, routine use is inconvenient because it runs on a cluster of computers and requires several hours of runtime when the filtering constraints are adjusted. MRPrimerW solved this problem completely. On the MRPrimerW website, users can rapidly design primers of the same high quality without using their own computer cluster, typically within a minute, while instantly and freely adjusting filtering constraints.

To achieve this level of performance, we adopted an approach based on Google's search system. In particular, we reorganized the complex MRPrimer algorithm, which consists of seven MapReduce rounds, into two parts: offline processing and online processing. We built index structures using the results of offline processing and loaded them into the MRPrimerW web server. Using these indices, the online processing stage can quickly design high-quality primers against a user-specified target, as in a Google keyword search.

1.3 Structure of thesis

The structure of this dissertation is organized as follows. In Chapter II, we introduced related works. The batch-style methods for primer design are presented in Section 2.1. In the following Section 2.2, we discuss websites for primer design and limitations. We propose the MRPrimer method in Chapter III. The overview of MRPrimer method is characterized in Section 3.1. Then, we explain details of seven steps of MRPrimer in Section 3.2. Furthermore, biological validation and computational evaluation are presented in Section 3.3 and 3.4 respectively. In Chapter IV, we propose MRPrimerW. The overview of MRPrimerW method is discussed in Section 4.1. In the following Section 4.2 through

Section 4.4, we describe details of MRPrimerW. Section 4.5 presents the MRPrimerW input and output interface. Finally, conclusions are drawn in Chapter V.

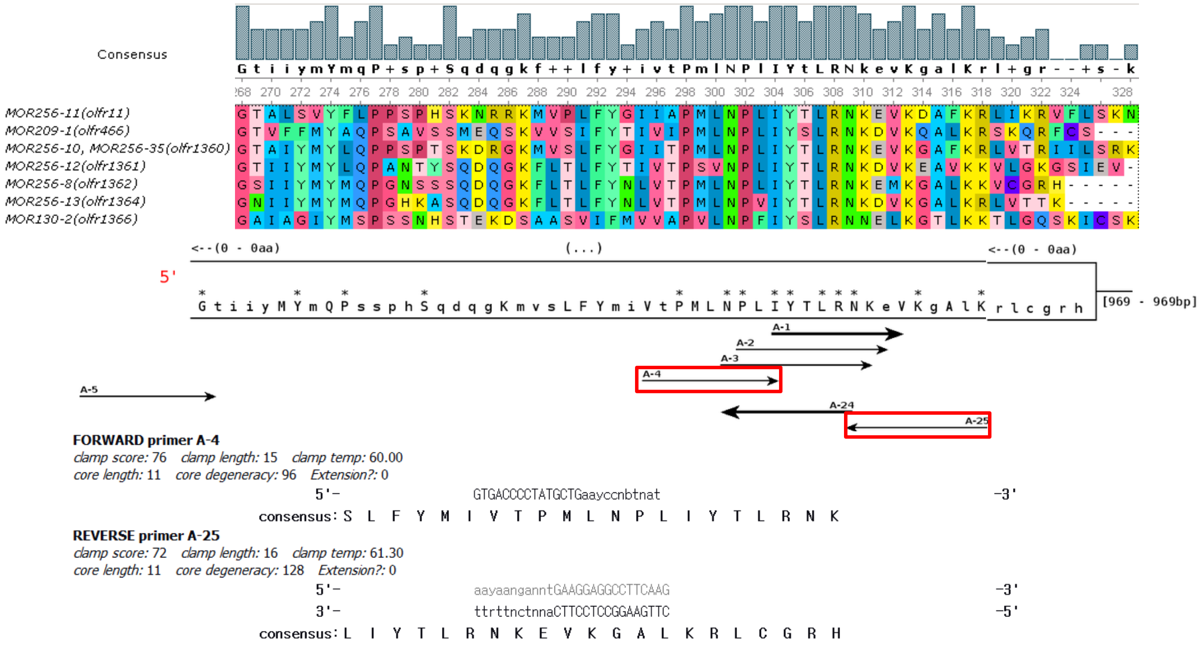
II. RELATED WORK

2.1 Batch-style primer design method

The existing methods for primer design can be categorized into two groups, depending on whether we wish to specify a single target sequence or multiple target sequences.

The major methods in the former group include Primer3Plus [21] and Primer-BLAST [22]. The core algorithm of both methods is based on Primer3 [23]. Primer3 is the most widely used tool to design primer from a single DNA sequence. Primer3Plus, a web interface of Primer3, allows users to specify a series of filtering constraints that the primers must satisfy; however, it does not perform homology tests on off-target sequences, and therefore requires the user to perform time-consuming tests with a BLAST-like tool for each candidate primer pair. Unlike Primer3Plus, Primer-BLAST performs homology tests; however, it specifies only a few filtering constraints, which makes it difficult to design primers as precisely as desired.

The major methods in the latter group include CODEHOP [1, 24], iCODEHOP [25], GeneFISHER/GeneFISHER2 [26, 27], HYDEN [14], FAS-DPD [15], DePiCt [28], Amplicon [29], and SCPrimer [30]. All of these methods design degenerate primers, which are actually mixtures of similar, but not identical primers. Most of them design primers by first aligning multiple target sequences to find conserved regions, and then designing primer pairs over those conserved regions. For example, CODEHOP and iCODEHOP align target sequences with CLUSTALW [12] and design hybrid degenerate primers that contain a short 3’ degenerate core region of about ~11–12 bp and a longer 5’ consensus clamp region of ~18–25 bp. Figure 2.1 shows the example of multiple sequence alignment results of mouse olfactory receptor genes from chromosome 13 using CLUSTAL Omega [31] and primers designed from the alignment result using iCODEHOP. It showed 7 primer candidates. Among them forward primer A-4 has 96 degeneracy, and reverse primer A-25 has 128 degeneracy. SCPrimer builds phylogenetic trees from aligned multiple sequences to identify candidate primers, and then performs a set-covering algorithm to determine the minimal set of primers required to amplify all members of the alignment. Some tools, such as HYDEN and DePiCt, do not rely on multiple sequence alignment for primer design, but still rely on heuristic techniques such as greedy hill climbing.



```
Consensus                       G t i i y m Y m q P + s p + S q d q g k f + + l f y + i v t P m l N P l I Y t L R N k e v K g a l K r l + g r - - + s - k
                                268 270 272 274 276 278 280 282 284 286 288 290 292 294 296 298 300 302 304 306 308 310 312 314 316 318 320 322 324   328
MOR256-11(olfr11)               G T A L S V Y F L P P S P H S K N R R K M V P L F Y G I I A P M L N P L I Y T L R N K E V K D A F K R L I K R V F L S K N
MOR209-1(olfr466)               G T V F F M Y A Q P S A V S S M E Q S K V V S I F Y T I V I P M L N P L I Y S L R N K D V K Q A L K R S K Q R F C S - - -
MOR256-10, MOR256-35(olfr1360)  G T A I Y M Y L Q P P S P T S K D R G K M V S L F Y G I I T P M L N P L I Y T L R N K E V K G A F K R L V T R I I L S R K
MOR256-12(olfr1361)             G T I I Y M Y L Q P A N T Y S Q D Q G K F L T L F Y T I V T P S V N P L I Y T L R N K D V K E A V K K V L G K G S I E V -
MOR256-8(olfr1362)              G S I I Y M Y M Q P G N S S S Q D Q G K F L T L F Y N L V T P M L N P L I Y T L R N K E M K G A L K K V C G R H - - - - -
MOR256-13(olfr1364)             G N I I Y M Y M Q P G H K A S Q D Q G K F L T L F Y N L V T P M L N P V I Y T L R N K D V K G A L K R L V T T K - - - - -
MOR130-2(olfr1366)              G A I A G I Y M S P S S N H S T E K D S A A S V I F M V V A P V L N P F I Y S L R N N E L K G T L K K T L G Q S K I C S K

        <--(0 - 0aa)                        (...)                                    <--(0 - 0aa)
   5'   *         *     *           *                              *     * *   * *   * * *       *       *
        G t i i y M Y m Q P s s p h S q d q g K m v s L F Y m i V t P M L N P L I Y T L R N K e V K g A l K r l c g r h   [969 - 969bp]

FORWARD primer A-4
clamp score: 76   clamp length: 15   clamp temp: 60.00
core length: 11   core degeneracy: 96   Extension?: 0
            5'-              GTGACCCCTATGCTGaayccnbtnat                       -3'
   consensus: S  L  F  Y  M  I  V  T  P  M  L  N  P  L  I  Y  T  L  R  N  K
REVERSE primer A-25
clamp score: 72   clamp length: 16   clamp temp: 61.30
core length: 11   core degeneracy: 128   Extension?: 0
            5'-              aayaangannt GAAGGAGGCCTTCAAG                     -3'
            3'-              ttrttnctnnaCTTCCTCCGGAAGTTC                      -5'
   consensus: L  I  Y  T  L  R  N  K  E  V  K  G  A  L  K  R  L  C  G  R  H
```

Figure 2.1. An example of multiple sequence alignment results using CLUSTAL Omega and primers designed from the alignment result using iCODEHOP.

2.2 Web-based primer design method

To aid in designing primers for PCR experiments, many websites have been developed, including Primer3Plus [21, 32], BatchPrimer3 [33], Primique [34], QuantPrime [35], primer-BLAST [22], and PrimerBank [7, 8]. Primer3Plus, a web interface of Primer3, is one of the most widely used tools; it allows users to specify a set of filtering constraints for a single target gene. BatchPrimer3, which adopts the Primer3 core algorithm, can design primers in batches for multiple target genes. However, neither server performs homology tests on off-target sequences, requiring users to perform time-consuming homology tests on each candidate primer pair using extrinsic alignment tools.

Primique performs homology tests using BLAST in a limited scope, i.e., only on a small secondary set of off-target sequences uploaded by the user. Due to a high computation overhead of homology testing, the maximum size of this secondary database is limited to 10 MB, much smaller than a whole genome sequence database and therefore too small for the design of high-quality primers. QuantPrime performs homology testing for primer pairs designed by Primer3 against the whole transcriptome (mRNA) and genome database using BLAST. Both Primique and QuantPrime rely on a local alignment algorithm for homology testing. However, a heuristic approach based on local alignment cannot accurately count the number of mismatches between a primer and an off-target sequence [22]; as a result, these methods could yield suboptimally specific primer pairs. On the contrary, Primer-BLAST performs homology tests with a global alignment algorithm to ensure full primer-target alignment; accordingly, Primer-BLAST tends to return more target-specific primer pairs. The core algorithm of Primer-BLAST based on Primer3 has a function that can rank primers by their penalty scores. However, Primer-

BLAST does not utilize this function to increase the chance of finding target-specific primers. Although Primer-BLAST exhibits better performance in terms of homology testing, it does not rank the designed primer pairs by their penalty scores, but ranks them by their specificity; moreover, it cannot support batch design for multi-target qPCR due to the large computational overhead required for more accurate homology tests.

Some websites, including PrimerBank [7, 8], RTPrimerDB [36-38], and qPrimerDepot [39], simply search a database of pre-designed primers, rather than designing primers in real time in response to user queries. In particular, PrimerBank is one of the largest databases of primers built and updated over the past several years. Because the specificities of the primers of PrimerBank have been experimentally validated under uniform conditions, these primers are fairly effective in real PCR experiments. However, because PrimerBank relies on the pre-designed primers, it does not allow users to adjust the filtering constraints, which might be important in the context of qPCR experiments requiring a full set of primer pairs that satisfy the same constraints. A comparison of other existing tools is summarized in Table 2.1.

Table 2.1. Comparison among websites for primer design.

Method	Batch designing	Filtering constraints	Homology test	Scoring (Ranking)	TaqMan probes
Primer3Plus	X	O	X	O	O
BatchPrimer3	O	O	X	O	O
Primique	O	O	△	O	X
QuantPrime	O	O	△	O	X
Primer-BLAST	X	O	O	△[a]	O
PrimerBank	X	X	O	X	X

O: fully supported

△: partially supported

X: not supported

[a] Primer-BLAST ranks the designed primer pairs not by penalty scores, but by specificity

III. MRPRIMER: Batch-style primer design method

3.1 Overview

MapReduce

MRPrimer is based on the MapReduce framework, whose dataflow is shown in Figure 3.1. One round of MapReduce consists of two user-defined functions, Map and Reduce. Here, Reduce is optional, and so can be omitted. The input data for a round is distributed over the disks of a cluster of computers. The partial data in each computer is processed by Map functions, shuffled via network, processed by Reduce functions, and then stored in disks, which is again fed to the Map functions of the next MapReduce round. The input/output format of data, i.e. signatures of Map and Reduce functions are formally defined as follows.

$$\text{Map: } \langle k1, v1\rangle \rightarrow \text{list}(\langle k2, v2\rangle)$$

$$\text{Reduce: } \langle k2, \text{list}(v2)\rangle \rightarrow \langle k3, v3\rangle$$

The Map function takes a pair of key and value, ⟨k1, v1⟩, as input (e.g. k1 is a sequence ID, and v1 is the sequence itself), and then returns a list of pairs ⟨k2, v2⟩ in a different domain, as output. The Reduce function takes a pair of key and list of values, i.e. ⟨k2, list(v2)⟩, as input, and then returns a pair of key and value, i.e. ⟨k3, v3⟩ in a different domain, as output. Here, we note that v2 in the output of Map becomes list(v2) in the input of Reduce, since the shuffle process gathers all v2s having the same k2, which are scattered over computers, into a single list of v2 on a single computer. MRPrimer largely relies on that feature of MapReduce for performing large-scale and complicated computation efficiently.

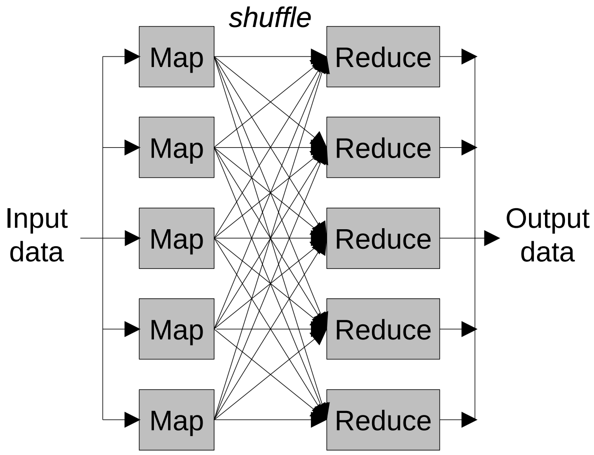


Figure 3.1. Dataflow of the MapReduce framework.

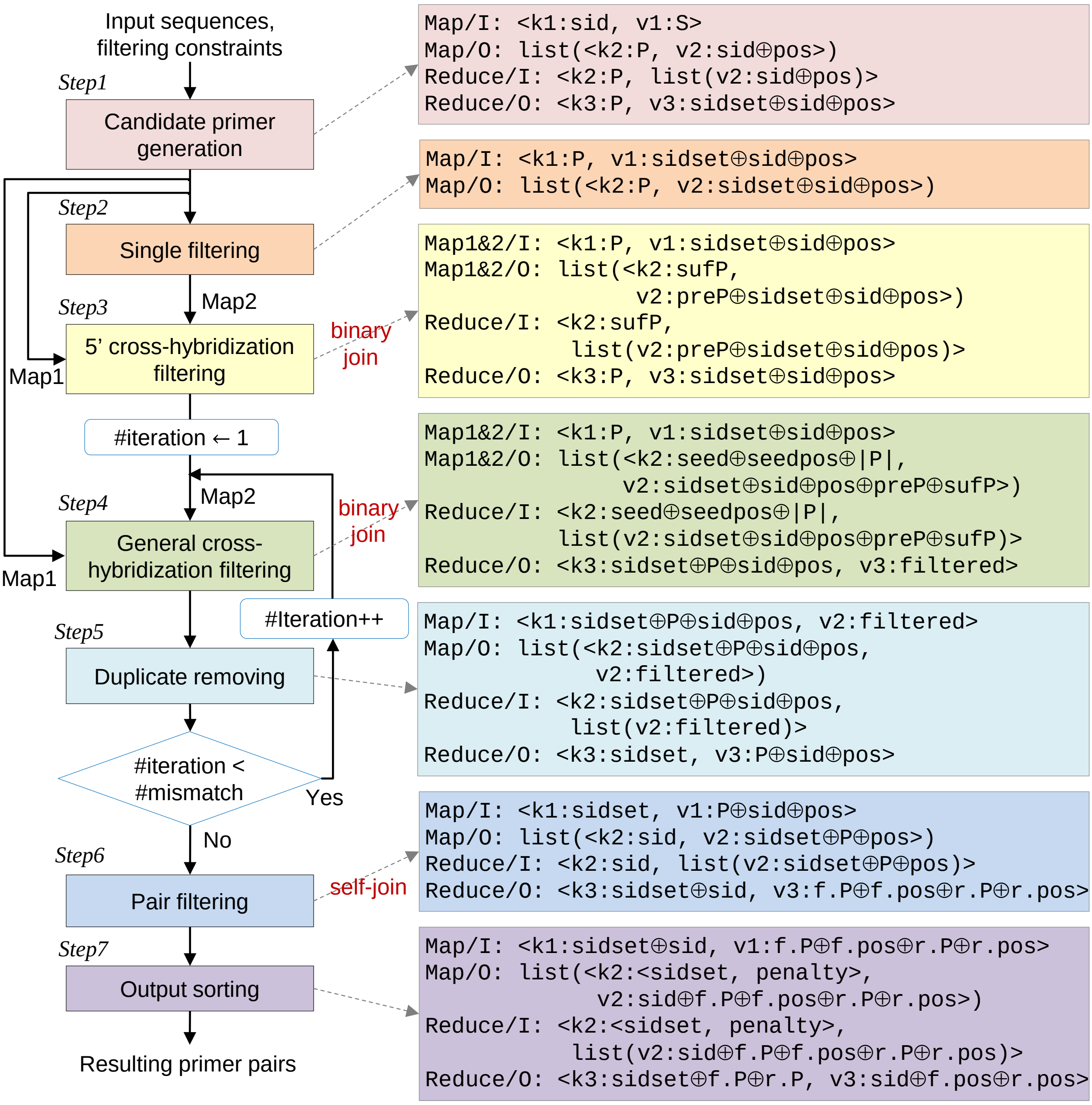
MRPrimer method

The flow of MRPrimer consists of seven steps (Figure 3.2). Each step corresponds to a single MapReduce round, which again consists of Map and Reduce. The output of Reduce of each round becomes the input of Map of its next round. MRPrimer takes a DNA sequence database and a set of filtering constraints, as input, where the set of filtering constraints include at least seven constraints for single primer, five constraints for primer pair, and two kinds of validation constraints. Then, through seven steps, it returns all feasible and valid primer pairs existing in the database and satisfying all constraints set by users.

The two kinds of validation constraints are 5’ cross-hybridization filtering constraint and general cross-hybridization filtering constraint. The former means the maximum number of mismatched residues at the 5’ end of a candidate primer that might cross-hybridize off-target sequences and so should be filtered out. For instance, we assume that a candidate primer is the same with any subsequence of an off-target sequence at the 3’ end and has only two mismatches at the 5’ end. Then, that candidate primer might cross-hybridize the off-target sequence due to the high similarity between both, especially at the 3’ end, and thus we filter out the candidate primer. The default value for this constraint is four, i.e. the candidate primers of up to four mismatches at the 5’ end (and all matches at the 3’ end) are filtered out. That value can be changed by users. The latter refers to the maximum number of mismatched residues spread over a candidate primer. It is sufficient to use a smaller value for this constraint (e.g. two) than for the former constraint. Hereafter, we denote the latter value as #mismatch.

Among seven steps of MRPrimer, an important ones are single primer filtering (Step 2), homology test (Step 3-5), and pair primer filtering (Step 7). Different from other steps, Step 3 (5’ cross-hybridization filtering round) and Step 4 (general cross-hybridization filtering round) take two kinds inputs, Map1 and Map2, for binary join computation between both. Here, Map1 indicates a set of all possible subsequences from off-target sequences, and Map2 indicates a set of candidate primers passed from the previous step. The series of Steps 4 and 5 is repeatedly performed until checking the general cross-hybridization filtering constraint is finished. For instance, if #mismatch = 2, the series of steps is performed twice, i.e. at #mismatch=1 and at #mismatch=2. Although Step 6 takes a single input, it splits the input into two sets, a set of forward primers and a set of reverse primers, and then performs self-join computation between both. In Figure 3.2, the boxes in the right part show the input/output formats of Map and Reduce of each MapReduce round. We explain the method used in each step in more detail.

.



```
Map/I: <k1:sid, v1:S>
Map/O: list(<k2:P, v2:sid⊕pos>)
Reduce/I: <k2:P, list(v2:sid⊕pos)>
Reduce/O: <k3:P, v3:sidset⊕sid⊕pos>
```

```
Map/I: <k1:P, v1:sidset⊕sid⊕pos>
Map/O: list(<k2:P, v2:sidset⊕sid⊕pos>)
```

```
Map1&2/I: <k1:P, v1:sidset⊕sid⊕pos>
Map1&2/O: list(<k2:sufP,
                v2:preP⊕sidset⊕sid⊕pos>)
Reduce/I: <k2:sufP,
           list(v2:preP⊕sidset⊕sid⊕pos)>
Reduce/O: <k3:P, v3:sidset⊕sid⊕pos>
```

```
Map1&2/I: <k1:P, v1:sidset⊕sid⊕pos>
Map1&2/O: list(<k2:seed⊕seedpos⊕|P|,
               v2:sidset⊕sid⊕pos⊕preP⊕sufP>)
Reduce/I: <k2:seed⊕seedpos⊕|P|,
          list(v2:sidset⊕sid⊕pos⊕preP⊕sufP)>
Reduce/O: <k3:sidset⊕P⊕sid⊕pos, v3:filtered>
```

```
Map/I: <k1:sidset⊕P⊕sid⊕pos, v2:filtered>
Map/O: list(<k2:sidset⊕P⊕sid⊕pos,
             v2:filtered>)
Reduce/I: <k2:sidset⊕P⊕sid⊕pos,
           list(v2:filtered)>
Reduce/O: <k3:sidset, v3:P⊕sid⊕pos>
```

```
Map/I: <k1:sidset, v1:P⊕sid⊕pos>
Map/O: list(<k2:sid, v2:sidset⊕P⊕pos>)
Reduce/I: <k2:sid, list(v2:sidset⊕P⊕pos)>
Reduce/O: <k3:sidset⊕sid, v3:f.P⊕f.pos⊕r.P⊕r.pos>
```

```
Map/I: <k1:sidset⊕sid, v1:f.P⊕f.pos⊕r.P⊕r.pos>
Map/O: list(<k2:<sidset, penalty>,
             v2:sid⊕f.P⊕f.pos⊕r.P⊕r.pos>)
Reduce/I: <k2:<sidset, penalty>,
           list(v2:sid⊕f.P⊕f.pos⊕r.P⊕r.pos)>
Reduce/O: <k3:sidset⊕f.P⊕r.P, v3:sid⊕f.pos⊕r.pos>
```

Figure 3.2. Overall flow of the seven-step MRPrimer method.

3.2 MRPrimer algorithm

3.2.1 Step 1: Candidate primer generation round

The Map of this step takes a DNA database, which is a set of pairs of sequence ID, sid, and sequence data, S, ⟨k1:sid, v1:S⟩, and extracts all possible subsequences of the lengths between the minimum length, minL, and the maximum length, maxL. The lengths minL and maxL are specified by users as one of the single-primer filtering constraints. The Map also extracts their reverse complementary primers while tagging them with “reverse primers”. All outputs of Maps of all computers are shuffled and fed into each Reduce, which again transforms its input to the output format ⟨k3:P , v3:sidset⊕sid⊕pos⟩. Here, P is a candidate primer, sidset the set of sequence IDs where P occurs, sid a specific sequence ID where P occurs, and pos the position where P occurs within the sequence of sid. Three values of sidset, sid, and pos are concatenated with each other using a character operator ⊕, and thus v3 becomes a single text value. For the operator ⊕, we can use any character that does not occur in sequence data (e.g. ‘-’). Figure 3.3 presents the Map and Reduce algorithms for Step 1.

Algorithm 1. Step1-Map

```
Input:   <k1:sid, v1:S> // sid is the ID of a sequence S
Output:  list(<k2:P, v2:sid⊕pos>)
for pos = 0 to |S|-minLen
   for len = minLen to maxLen
      if (pos + len ≤ |S|) then
         P = S[pos:pos+len];
         emit(P, sid⊕pos); // concatenation ⊕
         rP = reversePrimerGen(P); // reverse primer
         P = reverseTag(rP); // reverse primer tag
         emit(P, sid⊕pos+len); // concatenation ⊕
         end if
      end if
   end for
end for
```

Algorithm 2. Step1-Reduce

```
Input:   <k2:P, list(v2:sid⊕pos)>
Output:  <k3:P, v3:sidset⊕sid⊕pos> // sidset is a
                      concatenation ⊗ of sids covered by P
sidset ← ø;
foreach v in list(v2:sid⊕pos)
   sidset ← sidset⊗v.sid;  // concatenation ⊗
end for
foreach v in list(v2:sid⊕pos)
   emit(P, sidset⊕v.sid⊕v.pos);
end for
```

Figure 3.3. The Map and Reduce algorithms for Step 1.

3.2.2 Step 2: Single filtering round

This step applies six filtering constraints for a single primer to the candidate primers passed from Step 1. The constraints include melting temperature, GC content, self-complementarity, 3'-end self-complementarity, contiguous residue, and Gibbs free energy. Checking of a single filtering constraint can be defined as a binary function. The function length() checks if a primer satisfies the length constraint. It can be defined as

$$length(p, minL, maxL) = \begin{cases} true, & if\ minL\ \leq |p| \leq maxL \\ false, & otherwise \end{cases}$$

, where p is a primer, minL is a minimum length, and maxL is a maximum length. The primer length was already checked in Step 1. The function temp() checks if a primer satisfies the melting temperature constraint, which can be defined as

$$temp(p, minT, maxT) = \begin{cases} true, & if\ minT\ \leq Tm(p) \leq maxT \\ false, & otherwise \end{cases}$$

, where Tm() is a function to calculate a melting temperature, minT is a minimum temperature, and maxT is a maximum temperature. There are various Tm() functions. To date the most accurate formula, the nearest neighbor method, which returns as below, can be used.

$$Tm(p) = \frac{\Delta H^{\circ} \times 1000}{\Delta S^{\circ} + R \times \ln\left(\frac{C_T}{4}\right)} - 273.15$$

, where ΔH° is the enthalpy change, ΔS° is the entropy change, R is the gas constant 1.9872cal/K-mol, and C_T is the total molar strand concentration. For calculating the melting temperature and the value of free energy, we adopted the nearest-neighbor thermodynamic model [40], which is an improved model of the previous one [41] used in

Primer3Plus [21]. The function GCcontent() checks if a primer satisfies the GC content constraint. It can be defined as

$$GCcontent(p, minR, maxR) = \begin{cases} true, & if\ minR\ \leq GC(p) \leq maxR \\ false, & otherwise \end{cases}$$

, where GC(p) is the ratio of base-nucleic acid codes G and C in the primer p, minR is the minimum ratio, and maxR is the maximum ratio. In order to prevent hairpin formation, the function selfComplementary() checks if any sequences of a primer bind to anywhere its complementary sequence. It can be defined as

$$selfComplementary(p, maxC) = \begin{cases} true, & if\ |self - complement\ p| < maxC \\ false, & otherwise. \end{cases}$$

The function repeatSeq() checks whether a primer containing contiguous residues, which can be defined as

$$repeatSeq(p, maxRS) = \begin{cases} true, & if\ |repeat(p)| \leq maxRS \\ false, & otherwise \end{cases}$$

, where repeat(p) is the length of contiguous sequences in primer p, and maxRS is the maximum length of allowed repetitive sequences. The function freeEnergy() evaluates the free energy (ΔG) for annealing stability of the 3'end of primer to minimize nonspecific amplification. The function can be defined as

$$freeEnergy(p, bindLen\ , minG) = \begin{cases} true, & if\ \ deltaG(p, bindLen\ ) \geq minFE \\ false, & otherwise \end{cases}$$

, where deltaG(p, numBase) is the value of free energy determined by the most accurate nearest neighbor thermodynamics method [40], and bindLen is the specific length of the binding nucleic acid bases at the 3'-end of the primer p. All of these constraints can be specified by users when starting the program. There is no Reduce function in this step. Figure 3.4 presents the Map and Reduce algorithms for Step 2.

```
Algorithm 3. Step2-Map
Input:    <k1:P, v1:sidset⊕sid⊕pos>
Output:   list(<k2:P, v2:sidset⊕sid⊕pos>)
if (isReverse(P))
   P = reverseUntag(P);
end if
if (singleFiltering(P)) then
   emit (P, sidset⊕sid⊕pos); // concatenation ⊕
end if
```

```
Algorithm 4. Step2-Reduce
Input:    <k2:P, list(v2:sidset⊕sid⊕pos)>
Output:   <k3:sidset, v3:P⊕sid⊕pos> // sidset is a
                    concatenation ⊗ of sids covered by P
foreach v in list(v2:sidset⊕sid⊕pos)
   emit (sidset, P⊕v.sid⊕v.pos);
end for
```

Figure 3.4. The Map and Reduce algorithms for Step 2.

3.2.3 Step 3: 5’ cross-hybridization filtering round

This step takes two inputs, Map1, which is the output of Step 1, i.e. all possible subsequences, and Map2, which is the output of Step 2, i.e. a set of candidate primers that passed all single filtering constraints. The purpose of this step is eliminating a candidate primer that violates 5’ cross-hybridization filtering constraint. While performing all pair join between two sets, if a primer from Map1 and a primer from Map2 are identical with each other except at the 5’ end, the primer from Map2 is filtered out. Figure 3.5 shows an example of this step, where the primer (b) is identical with the primer (a) except at the 5’ end, and so is filtered out. The primer (c) is similar with the primer (a), and so should be removed. However, it does not violate 5' cross-hybridization constraint, which will be filtered out in the next step. Figure 3.6 illustrates the Map and Reduce algorithms for Step 3.

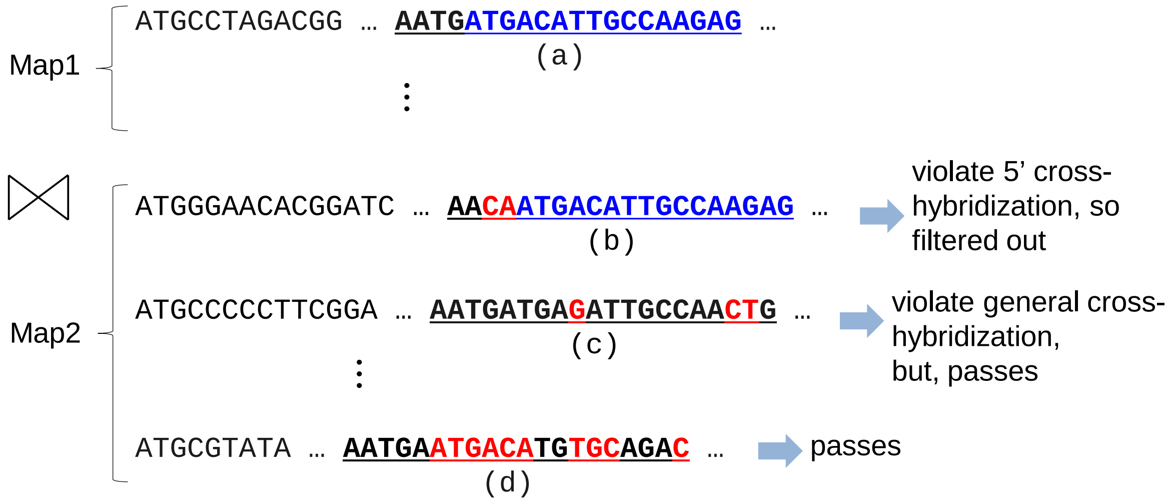

Map1
ATGCCTAGACGG … AATGATGACATTGCCAAGAG …
(a)
⋮
⋈
Map2
ATGGGAACACGGATC … AACAATGACATTGCCAAGAG …
(b)
violate 5' cross-hybridization, so filtered out
ATGCCCCCTTCGGA … AATGATGAGATTGCCAACTG …
(c)
violate general cross-hybridization, but, passes
⋮
ATGCGTATA … AATGAATGACATGTGCAGAC …
(d)
passes

Figure 3.5. An example of the 5' cross-hybridization filtering step.

Algorithm 5-1. Step3-Map1

```
Input:     <k1:P, v1:sidset⊕sid⊕pos>  // Step1 output
Output:    list(<k2:sufP, v2:preP⊕sidset⊕sid⊕pos>)
Variable:  prefixLen    // length of 5' subprimer
    preP = round1Tag(preP);
    preP += P[0:prefixLen-1];
    subP = P[prefixLen:|P|];
    emit(sufP, preP⊕sidset⊕sid⊕pos); // concatenation ⊕
```

Algorithm 5-2. Step3-Map2

```
Input:     <k1:sidset, v1:P⊕sid⊕pos>  // Step2 output
Output:    list(<k2:sufP, v2:preP⊕sidset⊕sid⊕pos>)
Variable:  prefixLen    // length of 5' subprimer
    subP = P[prefixLen:|P|];
    prefix = P[0:prefixLen-1];
    emit(sufP, preP⊕sidset⊕sid⊕pos); // concatenation ⊕
```

Algorithm 6. Step3-Reduce

```
Input:   <k2:sufP, list(v2:preP⊕sidset⊕sid⊕pos)>
Output:  <k3:P, v3:sidset⊕sid⊕pos>
R1, R2← ø;
foreach v in list(v2:preP⊕sidset⊕sid⊕pos)
   preP⊕sidset⊕sid⊕pos ← v   // decomposition
   if (!isRound1Tagged(prefix)
      R2  ← preP⊕sidset⊕sid⊕pos;
   else
      prefix = round1Untag(prefix);
      R1 ← preP⊕sidset⊕sid⊕pos;
   end if
foreach r2 in R2
   isFilter = false;
   foreach r1 in R1
      if (r2.sidset != r1.sidset) then
         isFilter = true;
         break;
      end if
   end for
   if (!isFilter)
      P = r2.prefix + subP
      emit (P, r2.sidset⊕r2.sid⊕r2.pos)
   end if
end for
```

Figure 3.6. The Map and Reduce algorithms of Step 3.

3.2.4 Step 4: General cross-hybridization filtering round

This step also takes two inputs, Map1, which is the output of Step 1, i.e. all possible subsequences, and Map2, which is the output of Step 3, i.e. a set of candidate primers that passed both the single filtering constraints and 5' cross-hybridization filtering constraint. While performing all pair join between two sets, if a primer from Map1 and a primer from Map2 are identical except a given number of mismatches, i.e. #mismatch, the primer from Map2 is filtered out.

In order to compute this step efficiently, Map of this step splits each primer into multiple seeds and transforms the input ⟨k1:P, v1:sidset⊕sid⊕pos⟩ to the output list(⟨k2:seed, v2: sidset⊕sid⊕pos⊕preP⊕sufP⟩). According to the theorem, a sequence of length m with at most k mismatches must contain a seed exactly matched of at least $\lfloor m/(k+1) \rfloor$ residues [42-44]. In the output format, preP means the left part of seed in a primer, and sufP means the right part of seed in the primer. So, the concatenation of preP, seed, and sufP is equal to the original P. All outputs of Maps are shuffled, and then all primers from Map1 and Map2 having the same seed are collected in the input of a specific Reduce. Thus, each Reduce can check general cross-hybridization filtering constraint on each set of primers having a common seed.

Figure 3.7 shows an example of that checking. Compared to the primer (a) from Map1, the primers (c) and (d) from Map2 have two and ten mismatches, respectively. When checking a single mismatch, i.e. #mismatch=1 in Figure 3.7B(i), the seed size becomes nine, and there is no common seed among (a), (c), and (d). Thus, the primers (c) and (d) are not collected together with the primer (a), and do not get filtered out. However, in the next iteration, i.e. #mismatch=2 in Figure 3.7B(ii), the seed size becomes six, and there is the common seed between (a) and (c), and between (a) and (d). The primers (a) and (c) are collected in a specific Reduce, the number of mismatches in the preP and sufP parts in both primers is checked as two, and so the primer (c) is filtered out. Since the number of mismatches between the primers (a) and (d) is so high, the primer (d) passes. If a primer passes successfully, v3:filtered is set to true in the output of Reduce. Otherwise, it is set to false. Figure 3.8 shows the Map and Reduce algorithms for Step 4.

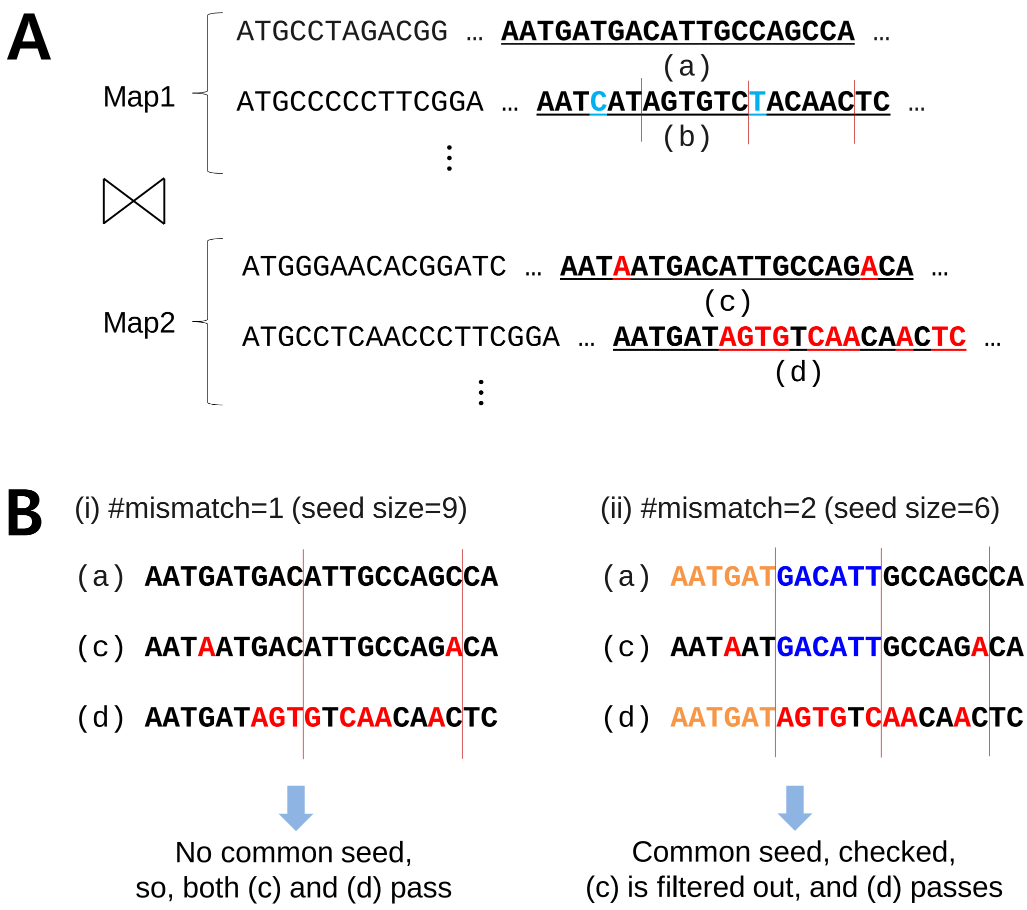

A
Map1
ATGCCTAGACGG … AATGATGACATTGCCAGCCA …
(a)
ATGCCCCCTTCGGA … AATCATAGTGTCTACAACTC …
(b)
⋮
Map2
ATGGGAACACGGATC … AATAATGACATTGCCAGACA …
(c)
ATGCCTCAACCCTTCGGA … AATGATAGTGTCAACAACTC …
(d)
⋮
B
(i) #mismatch=1 (seed size=9)
(a) AATGATGACATTGCCAGCCA
(c) AATAATGACATTGCCAGACA
(d) AATGATAGTGTCAACAACTC
No common seed,
so, both (c) and (d) pass
(ii) #mismatch=2 (seed size=6)
(a) AATGATGACATTGCCAGCCA
(c) AATAATGACATTGCCAGACA
(d) AATGATAGTGTCAACAACTC
Common seed, checked,
(c) is filtered out, and (d) passes

Figure 3.7. An example of the general cross-hybridization filtering step.

Algorithm 7-1. Step4-Map1

```
Input:    <k1:P, v1:sidset⊕sid⊕pos>  // Step1 output
Output:   list(<k2:seed, v2:sidset⊕sid⊕pos⊕preP⊕sufP>)
Variable: seedLen   // length of non-overlapping subsequenes

sidset = round1Tag(sidset);
for index = 0 to |P|-seedLen+1
   seed = P[index:index+seedLen-1];
   if (isReverse(P))
      seed = reverseTag(seed);
   end if
   preP = P[0:index-1];
   sufP = P[index+seedLen:|P|];
   emit (seed, sidset⊕sid⊕pos⊕preP⊕sufP);
   index += seedLen;
end for
```

Algorithm 7-2. Step4-Map2

```
Input:    <k1:P, v1:sidset⊕sid⊕pos>  // Step3 output
Output:   list(<k2:seed, v2:sidset⊕sid⊕pos⊕preP⊕sufP>)
Variable: seedLen   // length of non-overlapping subsequenes

for index = 0 to |P|-seedLen+1
   seed = P[index:index+seedLen-1];
   if (isReverse(P))
      seed = reverseTag(seed);
   end if
   preP = P[0:index-1];
   sufP = P[index+seedLen:|P|];
   emit (seed, sidset⊕sid⊕pos⊕preP⊕sufP);
   index += seedLen;
end for
```

Algorithm 8. Step4-Reduce

```
Input:     <k2:seed, list(v2:sidset⊕sid⊕pos⊕preP⊕sufP)>
Output:    <k3:sidset⊕P⊕sid⊕pos, v3:filtered>
Variable:  filterHashSet// HashSet storing false primer candidate
           numMismatch  // minimum number of mismatch bases

R1, R3 ← ø;
foreach v in list(v2:sidset⊕sid⊕pos⊕preP⊕sufP)
   sidset⊕sid⊕pos⊕preP⊕sufP ← v   // decomposition
   if (isRound1Tagged(sidset))
      sidset = round1Untag(sidset);
      R1 ← sidset⊕sid⊕pos⊕preP⊕sufP;
   else
      R3 ← sidset⊕sid⊕pos⊕preP⊕sufP;
   end if
end for
foreach r3 in R3
   P = r3.preP + r3.seed + r3.sufP;
   if (isReverse(r3.seed))
      P = reverseTag(P);
   end if
   if (!filterHashSet.contains(P))
      filtered = false;
      foreach r1 in R1
         if (r1.sid ∉ r3.sidset) then
            if (countMismatch(r3.preP, r1.preP,
                        r3.sufP, r1.sufP) < numMismatch)
               filtered = true;
               break;
            end if
         end if
      end for
      emit (r3.sidset⊕P⊕sid⊕r3.pos, filter);
      if (filtered)
        filterHashSet ← P;
      end if
   end if
end for
```

Figure 3.8. The Map and Reduce algorithms of Step4

3.2.5 Step 5: Duplicate removing round

After Step 4, there still might be false-positive primers violating the general cross-hybridization filtering constraint. For instance, in Figure 3.7, primer (d) passes when it is checked against primer (a). However, it should still be filtered out because it is very similar to another primer (b) in Map1. Because there are three seeds in primer (d) at the iteration of #mismatch = 2, a total of three output pairs, ⟨k3, v3⟩, are produced for (d) in the output of Reduce in Step 4. Among them, filtered of primer (d) checked with primer (a) is true, whereas filtered of primer (d) checked with primer (b) is false. They have the same sidset⊕P⊕sid⊕pos, and so are collected in Reduce in Step 5. If filtered of any of them is false, Reduce of Step 5 does not return the corresponding primer (e.g., primer (d)) as output, i.e., the primer is filtered out. The series of Steps 4 and 5 is performed repeatedly until checking of the general cross-hybridization filtering constraint is finished. Figure 3.9 presents the Map and Reduce algorithms for Step 5.

Algorithm 9. Step5-Map

Input: `<k1:sidset⊕P⊕sid⊕pos, filtered>`

Output: `list(<k2:sidset⊕P⊕sid⊕pos, v2:filtered>)`

```
emit(sidset⊕P⊕sid⊕pos, filtered);
```

Algorithm 10. Step5-Reduce

Input: `<k2:sidset⊕P⊕sid⊕pos, list(v2:filtered)>`

Output: `<k3:sidset, v3:P⊕sid⊕pos>`

```
falseCount = 0;
foreach v in list(v2:filtered)
  if (!v)
    falseCount ++;
  end if
end for
if (falseCount == 0)
   emit (sidset, P⊕sid⊕pos);
end if
```

Figure 3.9. The Map and Reduce algorithms of Step 5.

3.2.6 Step 6: Pair filtering round

In this step, Map first transforms the output of Step 5 into the format ⟨k2:sid, list(v2:sideset⊕P⊕pos) ⟩ such that all candidate primers belonging to the same sequence are collected in a specific Reduce. Then, Reduce splits the candidate primer of each group into two sets, a set of forward primers and a set of reverse primers, using tags addressed in Step 1, and performs self-join computation between them. In self-join computation, Reduce applies five filtering constraints to each candidate primer pair. These constraints include length difference, melting temperature difference, product size, pair-complementarity, and 3’-end pair-complementarity. They also can be defined as a binary function as follows.

The function lengthDiff() checks the difference between lengths of a forward primer fp and a reverse primer rp, which can be defined as

$$lengthDiff(fp, rp, lenD) = \begin{cases} true, & if\ abs(|fp| - |rp|) \leq lenD \\ false, & otherwise. \end{cases}$$

Similarly, the function TmDiff() checks the difference between Tm values of a forward primer fp and a reverse primer rp, which can be defined as

$$TmDiff(fp, rp, tmD) = \begin{cases} true, & if\ abs(Tm(fp) - Tm(rp)) \leq tmD \\ false, & otherwise. \end{cases}$$

The function productSize() checks if the product(or amplicon) size is within a certain range, which can be defined as

$$productSize(fp, rp, minS, maxS)$$
$$= \begin{cases} true, & if\ minS < pos(rp) - pos(fp) + |rp| < maxS \\ false, & otherwise \end{cases}$$

, where pos(p) is the nucleotide position of the primer p in a sequence. The function complementary() checks if a forward primer is not the complement of a reverse primer, or vice versa, which can be defined as

$$complementary(fp, rp, maxC) = \begin{cases} true, & if\ |complement\ at\ the\ 3'end\ of\ fp\ and\ rp| < maxC \\ false, & otherwise. \end{cases}$$

They all can be specified by users when starting the program. In the output of Reduce, f.P means a forward primer, r.P means the corresponding reverse primer, f.pos means the position of f.P, and r.pos means the position of r.P (Figure 3.2). Figure 3.10 shows the Map and Reduce algorithms for Step 6.

Algorithm 11. Step6-Map

Input: <k1:sidset, v1:P⊕sid⊕pos>

Output: list(<k2:sid, v2:sidset⊕P⊕pos>)

```
emit(sid, sidset⊕P⊕pos);
```

Algorithm 12. Step6-Reduce

Input: <k2:sid, list(v2:sidset⊕P⊕pos)>

Output: <k3:sidset⊕sid, v3:f.P⊕f.pos⊕r.P⊕r.pos>
// f is a forward primer and
   r is a reverse primer

```
fp, rp ← ø;
foreach v in list(v2:sidset⊕P⊕pos)
  sidset⊕P⊕pos ← v   // decomposition
  if (!isRvsPrimer(P))
    fp  ← sidset⊕P⊕pos;
  else
    P = reverseUntag(P);
    rp ← sidset⊕P⊕pos;
  end if
foreach f in fp
   foreach r in rp
      if (f.pos < r.pos and
         pairFiltering(f,r)) then
         sidset ← f.sidset∩r.sidset;
         emit (sidset⊕sid, f.P⊕f.pos⊕r.P⊕r.pos)
      end if
   end for
end for
```

Figure 3.10. The Map and Reduce algorithms of Step 6.

3.2.7 Step 7: Ranking round

The output of Step 6 is not convenient for users because it is unordered. Among millions of primer pairs, users might have difficulty in choosing a few. The primer pairs passed to Step 6 might not be equally effective even if they satisfy all the given constraints. Thus, the final step of MRPrimer, i.e., Step 7, determines their ranking by calculating a penalty score for each primer pair. The ranking is determined within a specific target sequence(s), and so users can easily pick the top-1 primer pair for each target sequence. The calculation of penalty scores follows the method of Primer3Plus [21], which adds penalty scores of seven constraints for single primers and five constraints for primer pairs. In general, for the constraints having a range (e.g., melting temperature), the median value has the lowest penalty. For the other constraints (e.g., self-complementarity), the smallest value, typically zero, has the lowest penalty. Each penalty score for each constraint is normalized between 0 and 1. Primer pairs with low scores have high rank for the corresponding target sequence. After calculating a penalty score for each primer pair, Map of Step 7 emits ⟨k2: sidset⊕penalty, v2: sid⊕f.P⊕f.pos⊕r.P⊕r.pos⟩. Then, Reduce takes the pairs grouped by sidset, and at the same time ordered by penalty, which is possible through so-called secondary sorting provided by MapReduce. Finally, Reduce transforms those ordered primer pairs into the format ⟨k3: sidset⊕f.P⊕r.P, v3: sid⊕f.pos⊕r.pos⊕penalty⟩, which means that a primer pair, ⟨f.P, r.P⟩, for amplifying a set of sequences of sidset occurs at ⟨f.pos, r.pos⟩ in sid. Figure 3.11 describes the Map and Reduce algorithms for Step 7.

Algorithm 13. Step7-Map

```
Input:     <k1:sidset⊕sid, v1:f.P⊕f.pos⊕r.P⊕r.pos>
Output:    list(<k2:<sidset, penalty>, v2:sid⊕f.P⊕f.pos⊕r.P⊕r.pos>)
Variable:  PrimerPenaltyWeight
penalty = PrimerPenaltyWeight *
          (SinglePanalty (f.p) + SinglePanalty (r.p));
emit(<sidset, penalty>, sid⊕f.P⊕f.pos⊕r.P⊕r.pos);
```

Algorithm 14. Step7-Partitioner

```
Input:     <k2:<sidset, penalty>, v2:sid⊕f.P⊕f.pos⊕r.P⊕r.pos>
Output:    PartitionID
Variable:  numReduceTasks
return(sidset[0] % numReduceTasks);
```

Algorithm 15. Step7-KeyComparator

```
Input:     <K1: <sidset, penalty>, k2: <sidset, penalty>>
Output:    comparisonResult
comparisonResult = compareTo(k1.sidset, k2.sidset);
if (comparisonResult == 0)
   return compareTo(k1.penalty, k2.penalty);
return comparisonResult;
```

Algorithm 16. Step7-Reduce

```
Input:     <k2:<sidset, penalty>, list(v2:sid⊕f.P⊕f.pos⊕r.P⊕r.pos)>
Output:    <k3:sidset⊕f.P⊕r.P, v3:sid⊕f.pos⊕r.pos>
foreach v in list(v2:sid⊕f.P⊕f.pos⊕r.P⊕r.pos)
   emit (sidset⊕f.P⊕r.P, sid⊕f.pos⊕r.pos);
end for
```

Figure 3.11. The Map and Reduce algorithms of Step 7, Ranking.

3.2 Experiments for biological validation

3.3.1 Data and methods

Validating the completeness and ranking method of MRPrimer

To show the completeness and superiority of MRPrimer in terms of the number of primer pairs designed, we compare the results of MRPrimer with PrimerBank, which is one of the largest databases of primers that has been built and updated over the past several years [7, 8, 45, 46]. PrimerBank uses the human and mouse genes databases of the NCBI RefSeq project [47-49]. There are multiple versions of the RefSeq database, specified by their release dates. Unfortunately, the version of the RefSeq database used for PrimerBank is out of date, and is therefore no longer available. Thus, we use the oldest version available for comparison because it is the version most similar to that used for PrimerBank. That version (released on 07/11/2007) contains a total of 22,942 human mRNA sequences and a total of 27,305 mouse mRNA sequences. For a fair comparison, we use the exact same set of filtering constraints as were used to construct PrimerBank [7]; these constraints are summarized in Table 3.1.

Table 3.1. The Summary of the constraints for filtering used in the PrimerBank.

	Parameter	Value
Each primer	primer length	19~23 bp
	melting temperature (TM)	60~63℃
	GC content	35~65%
	self-complementarity	< 5-mer
	3’ self-complementarity	< 4-mer
	Contiguous residue	< 6-mer
	Gibbs free energy (ΔG)	≥ -9 *kcal/mol*
Primer pair	length difference	≤ 3-mer
	TM difference	≤ 5℃
	product size	100~250 bp
	pair-complementarity	< 9-mer
	3’ pair-complementarity	< 4-mer

In addition, to show the effectiveness of ranking method of MRPrimer, we extracted the validated primer pairs that specifically cover mouse olfactory receptor (OR) sequences from PrimerBank and analyzed them using the ranking method of MRPrimer. Among 27,305 mouse mRNA sequences of the RefSeq database, there are 990 mouse OR genes. We searched for the NCBI Gene IDs of those genes in the PrimerBank (http://pga.mgh.harvard.edu/primerbank/index.html) and collected 778 validated primer pairs covering 768 mouse OR genes. MRPrimer can also find 772 out of 778 primer pairs, and so we can rank those 772 primer pairs, which are common between PrimerBank and MRPrimer, according to the ranking method of MRPrimer. The ranking results revealed the rationality of our ranking method. Six primer pairs were not found by MRPrimer because the six sequences containing them are not present in the version of the RefSeq database (released on 07/11/2007) used for our experiments.

qPCR analysis of MRPrimer

To validate the quality of primer pairs designed by MRPrimer, we performed qPCR experiments using the mouse CCDS database rather than the RefSeq database (http://www.ncbi.nlm.nih.gov/CCDS/CcdsBrowse.cgi). It provides a gold standard for coding-region locations [50, 51]. In the CCDS datasets, there are currently a total of 29,064 human gene DNA sequences (the last update was 29/11/2013) and a total of 23,874 mouse gene DNA sequences (the last update was 07/04/2014). We primarily used mouse genes for our qPCR analysis.

We randomly selected 96 OR genes and 99 non-OR genes (including pheromone receptors, G proteins, ion channels, signaling molecules, etc.). Thus, we performed a total of 195 qPCR experiments. For each gene, we selected the top-1 primer pair for that gene, according to the ranking method of MRPrimer. We summarize the forward and backward primers designed and selected automatically by MRPrimer in Appendix Tables S1 and S2. We followed the MIQE guidelines [52] for the qPCR experiments.

Comparative analysis between MRPrimer and PrimerBank

To demonstrate the effectiveness and superiority of MRPrimer for qPCR, we performed both qPCR and sequencing analyses and compared the results with those obtained using PrimerBank. Because the primers of both MRPrimer and PrimerBank easily succeeded in amplifying normal target sequences, we compared their performance using “difficult” target sequences, i.e., OR genes that have many homologous regions and, therefore, often fail in qPCR experiments [45]. OR genes form the largest multigene family in mammals

[53]. These genes share many homologous regions; consequently, it is difficult to design valid primer pairs for them [45]. A number of studies reported the expression of ORs in olfactory as well as non-olfactory tissues [54, 55]. For such studies, qPCR using valid primer pairs is an effective and simple way to detect OR genes [54, 55].

To prepare the mouse OR genes, we searched for the NCBI Gene IDs of the mouse OR genes from CCDS database in PrimerBank and collected 860 validated primer pairs, each of which amplifies a single OR gene. We first checked their specificity using Primer-BLAST. Among the 860 primer pairs, 599 primer pairs were of high quality (i.e., high specificity). These 599 primer pairs were 100% matches to their intended expected target genes, and the possibility of matching an off-target gene was no more than 80%. Among the remaining 261 primer pairs, 96 were 100% matched with the expected target genes, and the possibility of matching an off-target gene was no more than 85%. These 695 (599 plus 96) primer pairs are considered highly specific for their target genes. Among the remaining primer pairs, 75 primer pairs were 95% matched, and 69 primer pairs were 90% matched to both target and off-target genes. These 144 (75 plus 69) primer pairs can amplify target genes along with off-target genes (i.e., wrong target or multi-target). We selected about 6% of the pairs corresponding to the 695 highly specific genes (i.e., 40 primer pairs) and about 24% of the pairs corresponding to the 144 less specific genes (i.e., 34 primer pairs). Next, we selected 74 primer pairs from the results of MRPrimer for the same 74 genes. The selection ratios differed (6% vs. 24%) because the 695 genes are relatively easy to amplify, whereas the 144 genes are relatively hard. We also note that the 74 OR genes used for this experiment are distinct from the 96 OR genes in the above experiment; furthermore, they represent harder target sequences. We summarize the forward and backward primers for the 74 genes of MRPrimer and PrimerBank used in our experiments (Appendix Table S3).

To identify amplified samples, we compared the sequences of qPCR amplicons with the expected gene sequences using NCBI BLASTn (http://blast.ncbi.nlm.nih.gov/Blast.cgi) and checked the percent identity between the two sequences. For BLAST analysis, we applied the following criteria, used in a previous study [45]. If more than 50% of the length of an expected PCR product sequence matches with only the expected target sequence, multiple genes, or another gene with 100% identity between the sequences, it is considered to be target-specific, multiple-target, or wrong target, respectively. Finally, if a qPCR product sequence does not match with at least 50% of the length of its expected target sequence, it is considered to be a sequencing failure.

3.3.2 qPCR analysis

For validation of MRPrimer, we performed qPCR using the top-1 primer pairs designed and selected automatically by MRPrimer, covering 195 genes randomly selected from among the mouse CCDS database (Appendix Table S1 and S2). The qPCR results reveal that all primer pairs designed by MRPrimer successfully amplified the corresponding target genes (Figures 3.12 and 3.13). Each of the qPCR melting curves clearly yielded a single peak, suggesting that each qPCR product is a single product without off-target gene amplification. We confirmed the qPCR products by sequencing analysis (data not shown), indicating that MRPrimer specifically amplified the corresponding target genes.

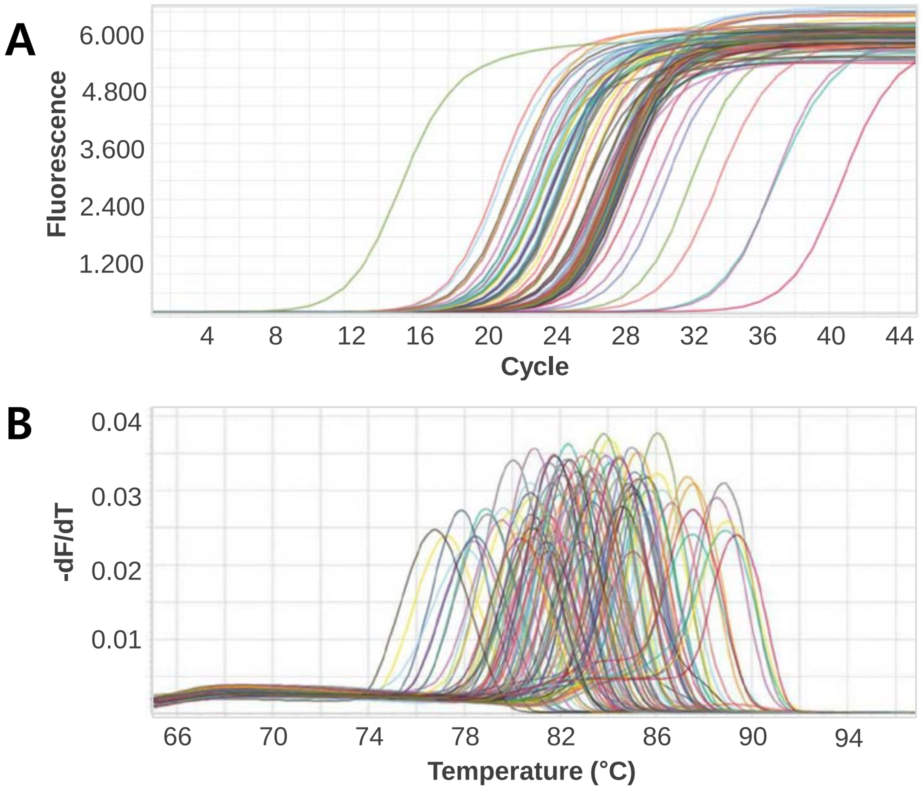


Figure 3.12. Verification of 99 primer pairs for non-OR genes using qPCR analysis.

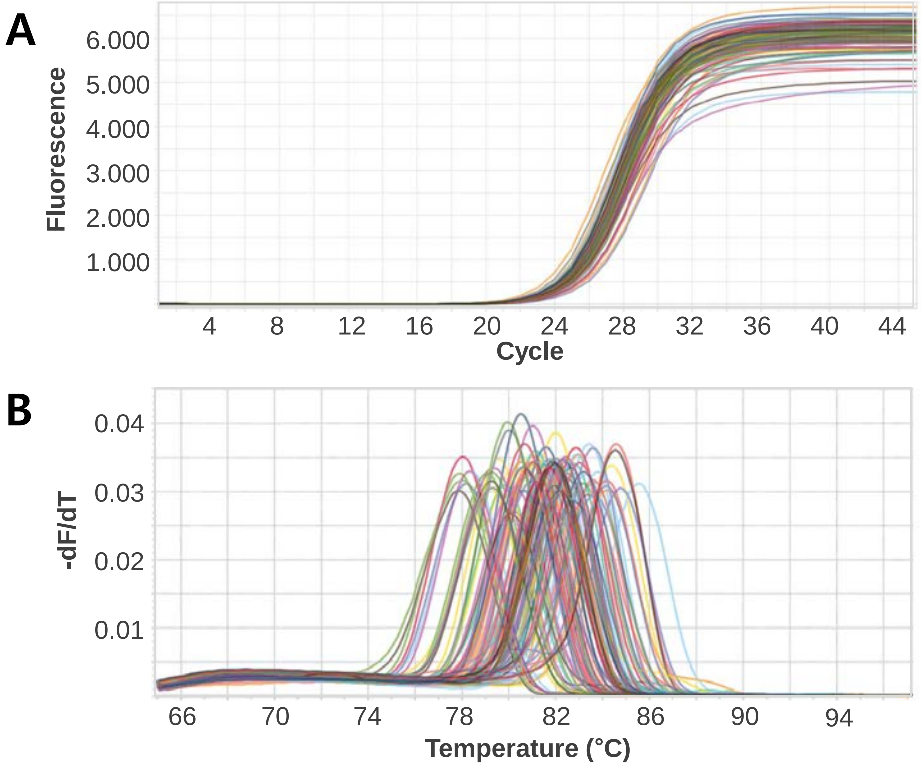


Figure 3.13. Verification of 96 primer pairs for OR genes using qPCR analysis.

3.3.3 Comparative analysis

For this comparative analysis, MRPrimer yielded similar results in qPCR analysis, and better results in sequencing analysis, relative to PrimerBank. Before starting the experiments, we analyzed primer sets (see Section 3.3.1). The selected 74 primer sets (Appendix Table S3) from MRPrimer and PrimerBank were used to perform qPCR. The result (Figure 3.14) shows that both MRPrimer and PrimerBank primers successfully amplified even the difficult target sequences like OR genes.

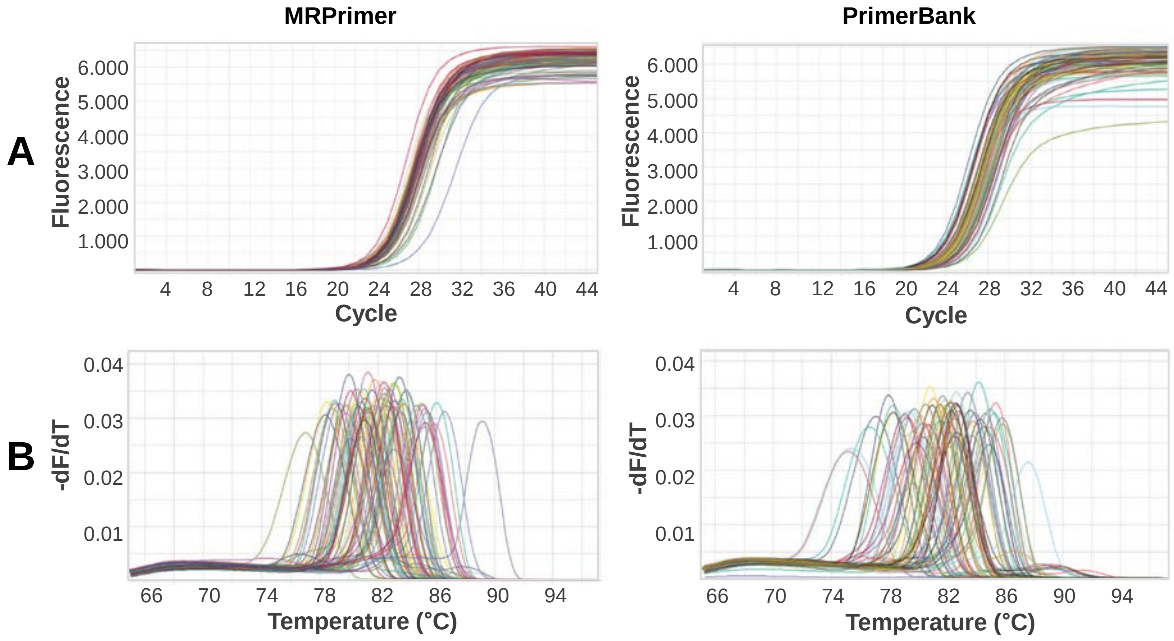


Figure 3.14. Comparative analysis between MRPrimer and PrimerBank using qPCR analysis.

However, the sequencing analysis yielded somewhat different results. We examined all PCR products by sequencing and compared the qPCR amplicon sequences to the expected gene sequences by NCBI BLASTn. Figure 3.15 shows the sequencing results. Among the MRPrimer 74 qPCR amplicons, 64 samples (86.48%) were target-specific, and these samples were 100% matched to the only expected target. Four samples (5.4%) were matched to both the expected target and an unexpected target at the same time (multi-target). Only one sample was matched to another gene (wrong target). The remaining five samples (6.75%) did not satisfy our criteria for sequencing analysis. On the other hand, among the 74 PrimerBank samples, 57 (77.02%) were target gene specific, 9 (12.16%) were matched to multiple genes (multi-target), 1 was matched to another gene (wrong target), and 7 (9.45%) did not satisfy our criteria for sequencing analysis. Based on these results, we confirmed that a single qPCR peak does not indicate the amplification of a specific single target. Because we intentionally selected difficult target sequences for this comparative analysis, the target-specific ratio of 86.48% does not indicate the overall effectiveness of MRPrimer. These findings suggest that primers designed by MRPrimer were more effective than PrimerBank primers.

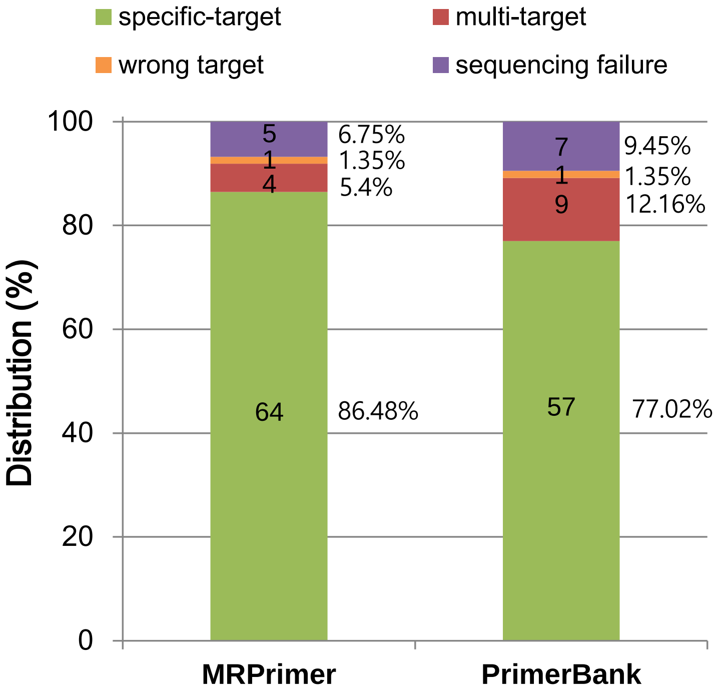


Figure 3.15. Comparative analysis between MRPrimer and PrimerBank using sequencing analysis.

3.3 Experiments for computational performance

3.4.1 Data and setup

To demonstrate the computational efficiency and scalability of MRPrimer, we measured the elapsed time required for design of complete sets of validated primer pairs for the human and mouse CCDS databases. We conducted most of the computational experiments on a MapReduce cluster of one master node and 40 slave nodes, in which each node consisted of two Intel Xeon 8-core 2.6 GHz CPUs with 64 GB memory and a 6 TB HDD. Those nodes are connected with each other via a 1 Gbps network. All computing nodes were running on CentOS Linux version 6.4 and Apache Hadoop version 1.2.1. For the Hadoop configuration parameters, we set the number of Map per node to 4, the number of Reduce per node to 4, the Java heap memory size for Map to 8 GB, and the Java heap memory size for Reduce to 16 GB.

3.4.2 Result of completeness and effective ranking system

In terms of the number of primer pairs designed, MRPrimer found a much larger number of feasible and valid primer pairs than PrimerBank under the same filtering constraints (Table 3.1). Table 3.2 shows the number of primer pairs designed by MRPrimer and the number of genes covered by those primer pairs, relative to the corresponding values for PrimerBank. In Table 3.2, we show that PrimerBank yields a coverage ratio of 94% for their RefSeq database, which is not available now. In terms of the version of the RefSeq database released on 07/11/2007, which is the available version most similar to that used for PrimerBank, the coverage ratio decreases to 78% for human genes and 69% for mouse genes. The size of the RefSeq database is increasing continuously, and the latest version (released on 03/02/15) contains a total of 99,722 sequences for human and 128,898 sequences for mouse. However, the number of primer pairs in PrimerBank is fixed, and has not increased since 2012. Because PrimerBank consists of primers collected manually, it is extremely hard to update it according to the release of a new version of the RefSeq database. By contrast, MRPrimer is not a static collection, but a program that can generate a collection immediately when given a new version of a database. For the same RefSeq database, the coverage ratios for MRPrimer (88% and 81%) are much higher than those for PrimerBank (78% and 69%). In addition, for up-to-date human and mouse CCDS databases, MRPrimer exhibits the highest coverage ratios ever: 95% for human and 96% for mouse. These impressive ratios are mainly due to the high quality of the CCDS database.

Table 3.2. The statistics of PrimerBank and the results of MRPrimer.

	PrimerBank[a]		**PrimerBank**[b]		**MRPrimer**[c]		**MRPrimer**[d]	
Data sets	Human N/A	Mouse N/A	Human 22,942	Mouse 27,305	Human 22,942	Mouse 27,305	Human 29,064	Mouse 23,889
# of primer pairs	129,692	118,886	129,692	118,886	63,419,755	86,867,667	63,632,594	84,226,391
# of genes covered	17,973	18,955	17,973	18,955	20,199	22,253	27,980	22,798
Coverage ratio	94%		78%	69%	88%	81%	95%	96%

N/A indicates data sets that are not available.

[a] Statistics are from PrimerBank [7].

[b] Statistics are the same as with [a] , but the data set is the RefSeq database (released on 07/11/07), the available data set most similar to the one used for [a].

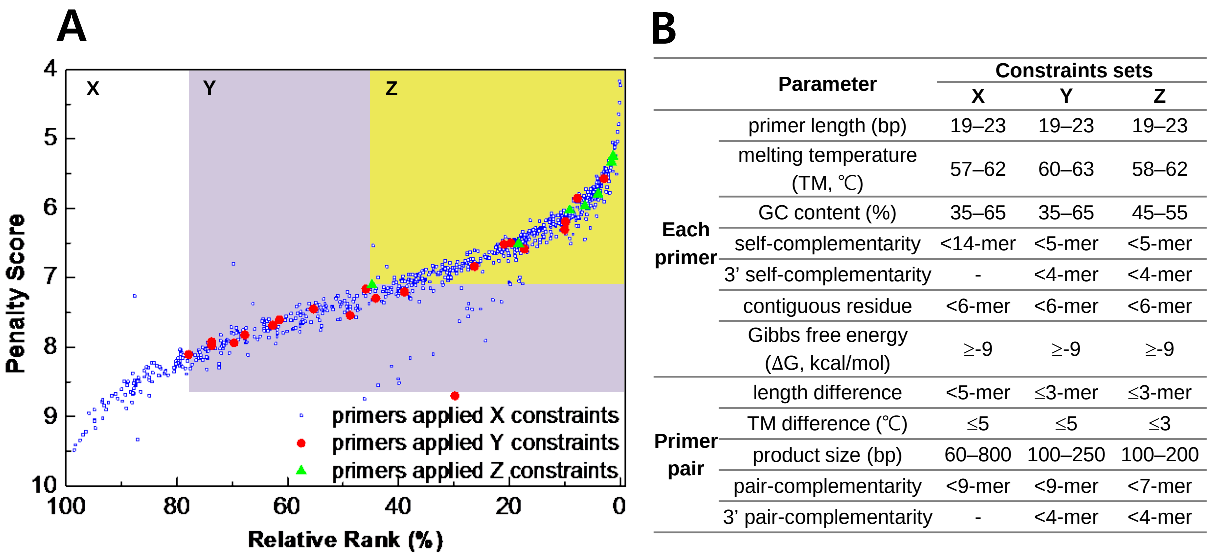
[c] The data set and filtering constraints (Table 3.1) are the same as in [b]. Statistics are from MRPrimer.

[d] The data set is the CCDS database, and the filtering constraints are the same as in [b]. Statistics are from MRPrimer.

MRPrimer yielded effective ranking results for a large number of the resultant primer pairs. We extracted a total of 772 common validated primer pairs that specifically cover mouse OR sequences and analyzed them using the ranking method of MRPrimer.

Figure 3.16A shows the relationship between ranks and penalty scores of those 772 primer pairs. MRPrimer calculates a penalty score for each primer pair and determines the ranking among primer pairs for a specific target sequence, as described in Step 7. Because there are different numbers of primer pairs for each target sequence, we normalized the ranks to between 0% and 100%, denoted as relative rank in the figure. Figure 3.16A shows the strong correlation between ranks and penalty scores. Primer pairs with small penalties have high rank (i.e., small %).

Figure 3.16B shows three sets of filtering constraints. X is a relatively relaxed constraint, Y is the set of constraints used in PrimerBank, and Z is a relatively strict constraint. According to X, Y, and Z, the 772 primer pairs can also be categorized into the corresponding three groups designed by using X, Y, and Z (denoted as blue, red, and green dots, respectively). Although the authors of PrimerBank claimed that they used the Y constraints to construct PrimerBank, we observed that the primer pairs in PrimerBank did not strictly follow the Y constraints, but instead followed the X constraints, which are looser. Groups X, Y, and Z contain 737, 28, and 7 primer pairs, respectively. Some primer pairs (blue dots) exist in the area of Y or Z because a primer pair that satisfies all constraints except one (or a few) could have a low penalty score and a high rank. Along with this, we suggest that primer pairs following strict constraints have high ranks and small penalty scores without loss of generality. Because a primer pair with a low penalty score has a high chance of success in amplifying a target sequence [21], and MRPrimer returns the resultant primer pairs ordered by rank, users simply need to select the top-1 primer pair, i.e., the probably best primer pair.



	Parameter	Constraints sets		
		X	Y	Z
Each primer	primer length (bp)	19–23	19–23	19–23
	melting temperature (TM, ℃)	57–62	60–63	58–62
	GC content (%)	35–65	35–65	45–55
	self-complementarity	<14-mer	<5-mer	<5-mer
	3' self-complementarity	-	<4-mer	<4-mer
	contiguous residue	<6-mer	<6-mer	<6-mer
	Gibbs free energy (ΔG, kcal/mol)	≥-9	≥-9	≥-9
Primer pair	length difference	<5-mer	≤3-mer	≤3-mer
	TM difference (℃)	≤5	≤5	≤3
	product size (bp)	60–800	100–250	100–200
	pair-complementarity	<9-mer	<9-mer	<7-mer
	3' pair-complementarity	-	<4-mer	<4-mer

Figure 3.16. The advantage of the ranking method of MRPrimer.

3.4.3 Results of the coverage and specificity

MRPrimer finds all possible primer pairs regardless of their coverages, that is not only all the primer pairs of coverage = 1, but also, all the primer pairs of coverage > 1. Due to the completeness and exactness of MRPrimer, it could find a primer pair having very high coverage. Figures 3.17A and 3.17B show the number of primer pairs and the number of unique genes covered by primer pairs, at each coverage for human and mouse CCDS data sets, respectively. MRPrimer could design a huge number of primer pairs of up to coverage = 25 for human CCDS data set and up to coverage = 20 for mouse CCDS data set. While considering the primer pairs of MRPrimer are not degenerate ones, those high coverages are quite impressive. For human CCDS data set, the number of primer pairs of coverage = 1 is 25,181,775 (67.6%), whereas that of primer pairs of coverage > 1 is 12,054,846 (32.4%). The number of primer pairs tends to decrease while the coverage increases.

However, in some cases, e.g. when the coverage increases from 24 to 25, the numbers of primer pairs also increase. This is because MRPrimer only designs a kind of essential primer pairs, i.e. does not produce a primer pair only covering a proper subset of the genes that are covered by another primer pair.

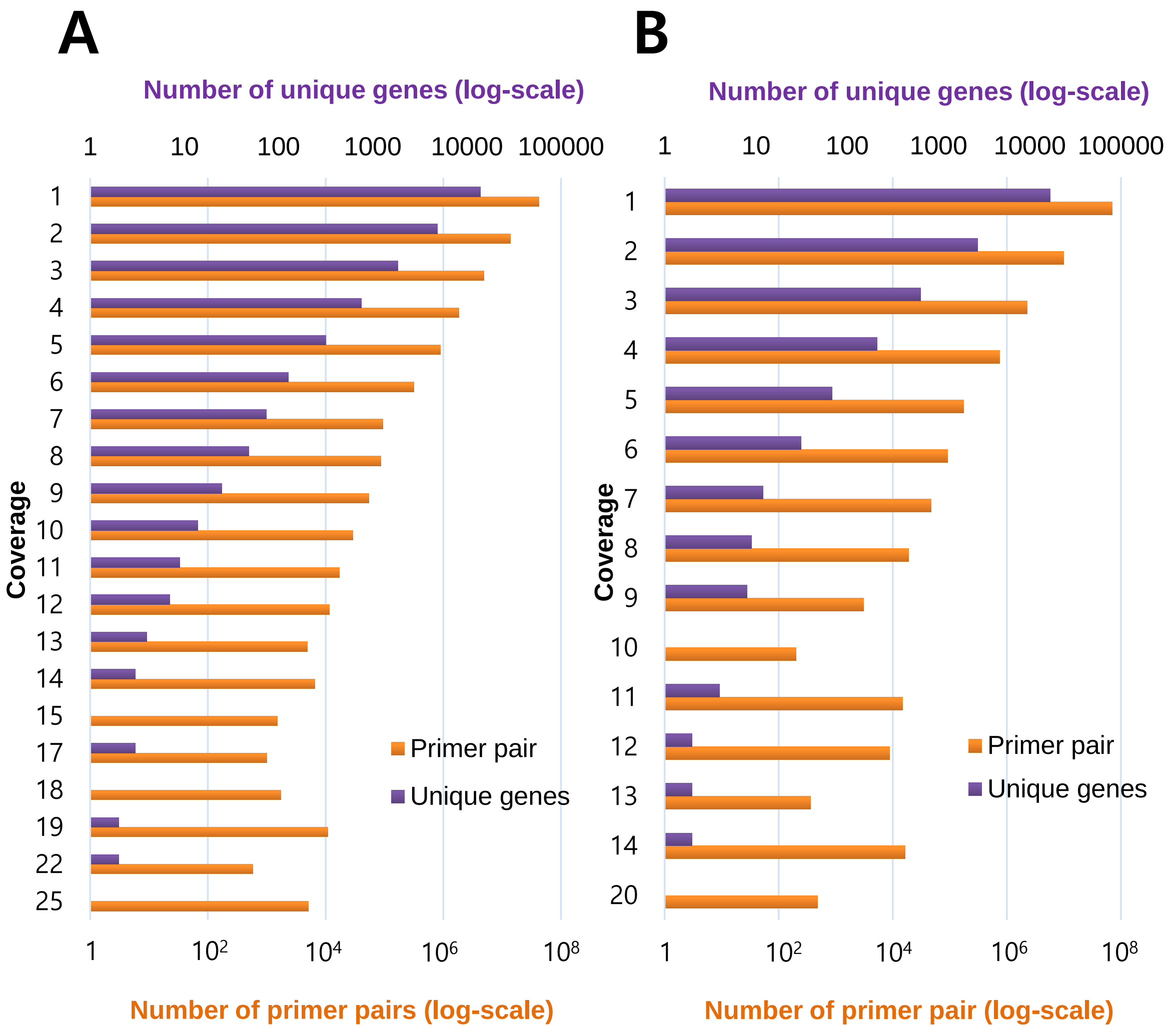


Figure 3.17. The distribution of the number of primer pairs (in orange) and the number of unique genes (in purple) according to coverage.

MRPrimer starts with all possible subsequences generated in Step 1 and eliminates the candidate primers violating filtering constraints gradually, as following the flow of single primer filtering (Step 2), 5’ cross-hybridization filtering (Step 3), general cross-hybridization filtering (Step 4-5), and pair primer filtering (Step 7). This series of filtering improves the specificity of the resulting primer pairs designed by MRPrimer as a result. Figures 3.18A and 3.18B show the number of primers (or primer pairs) passed in each major step, for human and mouse CCDS data sets, respectively. The general cross-hybridization filtering again is divided into two sub-steps of #mismatch=1(i.e. relatively low specificity) and #mismatch=2(i.e. relatively high specificity). Different from the existing methods like PrimerBLAST, MRPrimer allows users to increase or decrease the specificity of resulting primer pairs in homology tests, by adjusting #mismatch. In Figure 3.18A, the number of primers for human genes decreases gradually to 7,253,513, and self-join computation on them in Step 6 generates much more results, 37,236,621, which are primer pairs, not single primers. Likewise, in Figure 3.18B, the number of primers for mouse genes decreases gradually to 8,508,645, and self-join computation on them generates 48,532,297 primer pairs.

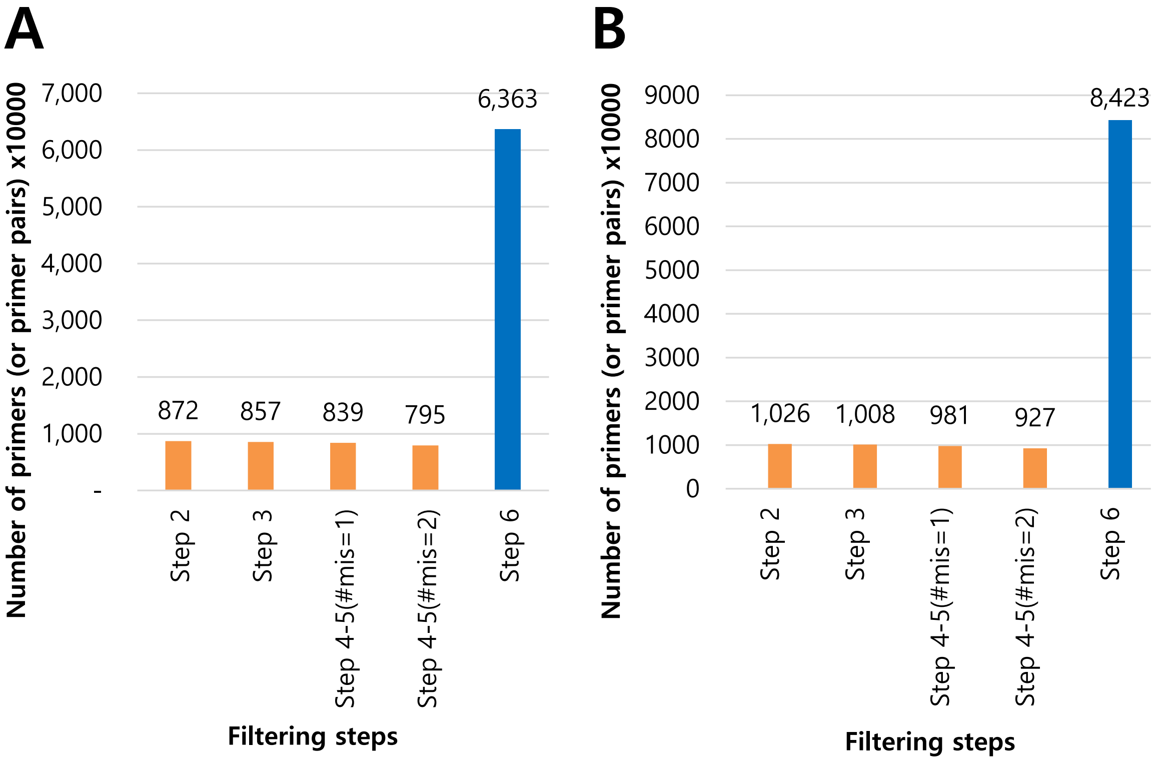


Figure 3.18. The number of primers (in orange) and primer pairs (in blue) passed in each major step of MRPrimer.

3.4.4 Results of the computational efficiency and scalability

MRPrimer showed a good performance in terms of computation time. Table 3.3 shows the elapsed times of MRPrimer at each step for human and mouse CCDS data sets. Even though MRPrimer designs all feasible and valid primer pairs, without omitting any one, it finishes within one or two hours. Once obtaining the results, users do not need to run it again and just need to pick the primer pairs, especially the top-1 primer pair, from the results that they want to use for experiments, unless the filtering constraints are changed.

Table 3.3. The elapsed times (sec.) of MRPrimer from Step 1 to Step 7 for human and mouse CCDS data sets (Step 4 and 5 are performed two times at #mismatch=1 and at #mismatch=2).

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Total
human	69	60	65	3,648	56	586	29	4,513
mouse	50	50	59	2,590	55	224	30	3,058

MRPrimer also showed fairly scalable performance in terms of database size (i.e. the number of DNA sequences). To show this feature authentically, we used a much larger DNA database, 105,180 DNA sequences of Homo sapiens from the Ensembl site (http://asia.ensembl.org/biomart/martview/). Figure 3.19 shows the elapsed times of MRPrimer while varying the number of sequences from 12,500 to 105,180 (i.e. an entire database). Even for 105,180 sequences, MRPrimer designs all feasible and valid primer pairs within a reasonable time of less than seven hours. Since MRPrimer is based on MapReduce, users can reduce the time easily just by adding more computers to the cluster.

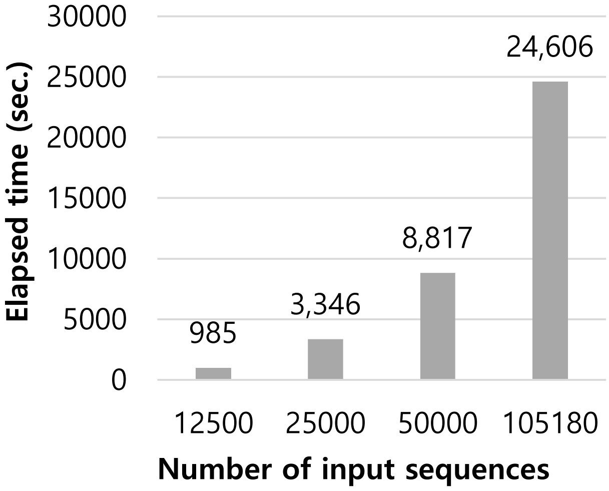


Figure 3.19. The elapsed times of MRPrimer as varying the database size.

Furthermore, MRPrimer was very efficient in terms of computational resource, i.e. the number of computers or the computing power of each computer. To show this feature, we perform the same experiments for human and mouse CCDS data set with a small-scale cluster of commodity PCs as well. The cluster consists of one master PC and ten slave PCs, where each PC consists of Intel i7-4770 4-core 3.4GHz CPU and 16GB memory, and 3TB HDD. Here, we use the same number of mappers and reducers, i.e. 4 and 4, respectively, but use smaller java heap memory sizes, 4GB for Map and 8GB for Reduce, due to the small memory capacity of PC. Table 3.4 shows the elapsed times of MRPrimer, which still finishes within a short time of two or three hours.

Table 3.4. The elapsed times (sec.) of MRPrimer from Step 1 to Step 7 for human and mouse CCDS data sets with a smaller cluster of less-powerful computers (Step 4 and 5 are performed two times at #mismatch=1 and at #mismatch=2).

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Total
human	303	106	165	7,583	56	557	69	8,839
mouse	259	91	110	3,880	50	272	70	4,732

IV. MRPRIMERW: Web-based primer design method

4.1 Overview

In this chapter, we describe the MRPrimerW, which performs complete homology testing, supports batch design of primers for multi-target qPCR experiments, supports design of TaqMan probes, and ranks the resulting primers to return the top-1 best primers to the user. To ensure high accuracy, we adopted the core algorithm of a previously described MapReduce-based method, MRPrimer, but completely redesigned it (i.e., seven MapReduce rounds into two parts: offline processing and online processing) to allow users to receive query results quickly in a web interface, without requiring a MapReduce cluster or a long computation.

Offline processing by MRPrimerW, which is independent of user queries, generates all validated candidate SYBR Green primers and TaqMan probes satisfying homology tests. Homology testing on an entire sequence database can be achieved by a large-scale self-join computation without specifying a target sequence. Because this stage of processing performs homology tests for every candidate primer and probe against the entire sequence database via a non-heuristic approach, the resultant primers and probes are all target gene–specific, and at the same time no valid (i.e., target gene specific) primers and probes are missed. Offline processing takes at least several hours on a cluster of computers (e.g., ten PCs). On the other hand, the online processing stage is responsive to user queries, i.e., a specified set of target genes. This stage quickly searches for the best primer pairs for the target genes and shows them to the user, and in particular returns the best pair among all valid primers that satisfy user-specified filtering constraints for the corresponding target gene. Along with SYBR Green primer pairs, online processing returns TaqMan probes for the target gene, if applicable. As with MRPrimer, the criteria used for ranking the primers in MRPrimerW are the same as those used in Primer3Plus [32].

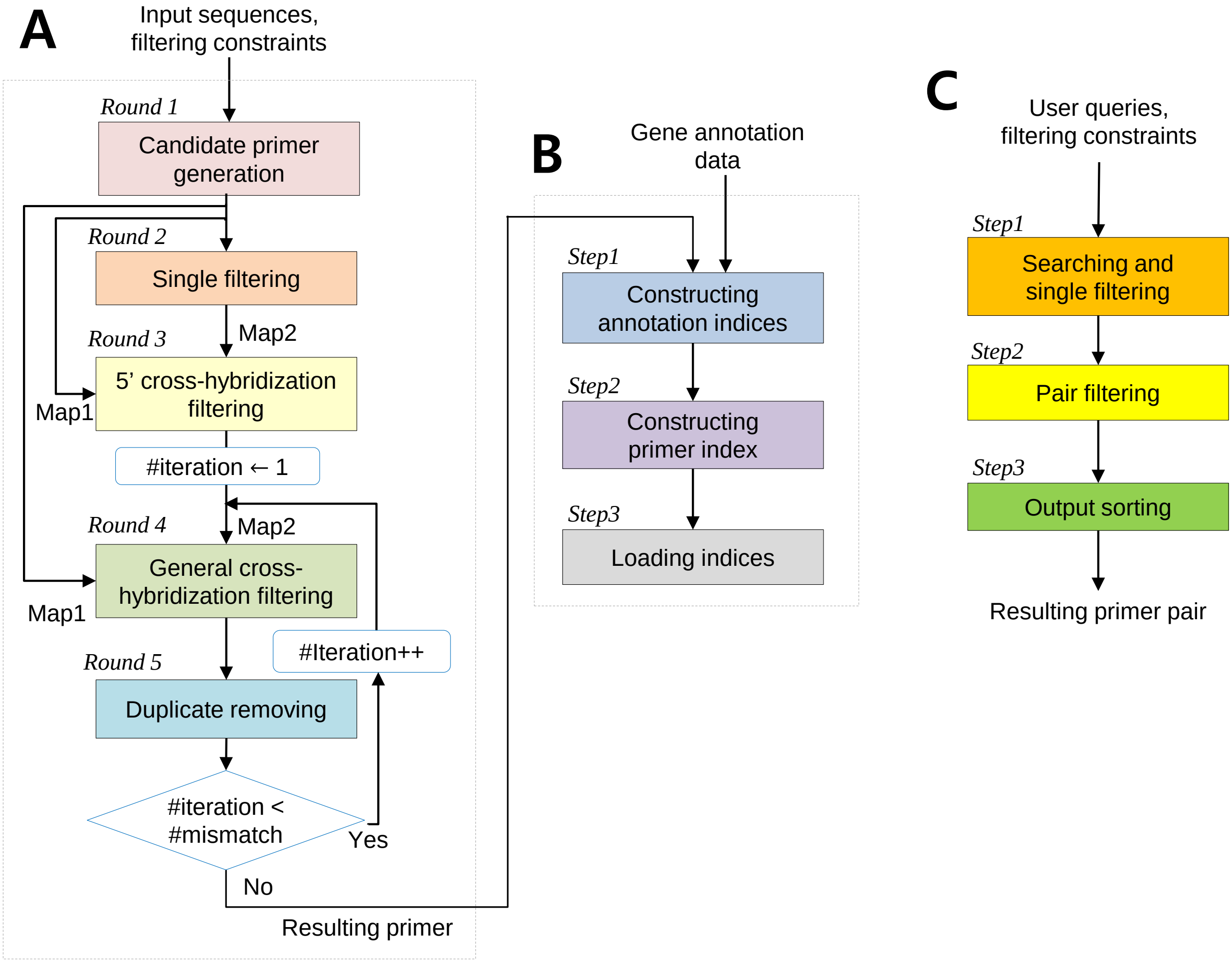


Figure 4.1. Overall flow of the MRPrimerW method.

4.2 Offline processing part

Offline processing by MRPrimerW takes as input a DNA sequence database and several filtering constraints, and yields as output all possible primers that satisfy both homology testing and given filtering constraints (Figure 4.1A). As an input source DNA sequence database for MRPrimerW, we used the consensus coding sequence (CCDS) database for human and mouse genes (https://www.ncbi.nlm.nih.gov/CCDS/) (Table 4.1). We selected these templates because the gene annotations have been defined by extensive manual curation and are represented consistently in the NCBI, Ensembl, and UCSC Genome Browsers [56-58]. The most up-to-date CCDS datasets contain 31,394 human gene sequences (Release 18, the last update was May 12, 2015) and 24,833 mouse gene sequences (Release 19, the last update was July 30, 2015). About 1% of human and mouse genes in CCDS do not have any target gene–specific primers; as a result, the offline processing stage produced valid primers for 31,376 human genes (99%) and 24,797 mouse genes (99%), comprising 165,923,450 and 176,039,685 distinct primers, respectively.

Table 4.1. Statistics of MRPrimerW primers.

	Human	**Mouse**	**Both species**
Total number of genes	31,394	24,833	56,227
Number of covered genes (%)	31,376 (99%)	24,797 (99%)	56,173 (99%)
Number of valid primers	165,923,450	176,039,685	341,963,135

For filtering constraints, MRPrimerW considers eight parameters for each primer and five parameters for each pair, as in MRPrimer (Table 4.2). Most of these constraints are checked during online processing. However, four parameters (primer length, melting temperature, GC content, and contiguous residue) are checked during offline processing, because primers with values out of the appropriate range (e.g., primer length 10 bp) are non-functional in general; consequently, they do not need to be considered during online processing. Table 4.2 shows the list of filtering constraints used in offline and online processing. The parameter ranges in the 'Online' column indicate the default settings, which can be adjusted in each web search.

Table 4.2. List of filtering constraints used in offline and online processing of MRPrimerW.

	Parameter	**Value range**	
		Offline	**Online (default)****
each primer	primer length	17–27 bp	19–23 bp
	melting temperature (TM)*	56–64℃	58–62℃
	GC content	30–70%	40–60%
	self-complementarity	-	<5-mer
	3' self-complementarity	-	<4-mer
	Contiguous residue	<5-mer	<6-mer
	Gibbs free energy (ΔG)	-	≥-9 *kcal/mol*
	Hairpin		<3-mer
Primer pair	length difference	-	≤5-mer
	TM difference	-	≤3℃
	product size	-	100–250 bp
	pair-complementarity	-	<5-mer
	3' pair-complementarity	-	<4-mer

- indicates not applicable.

* To calculate the melting temperature, we adopted the nearest-neighbor thermodynamic model [40].

** The value ranges in this column indicate the default setting, which can be freely adjusted by users.

Offline processing consists of five MapReduce rounds (Figure 4.1A). The first and second rounds generate all possible subsequences satisfying the four filtering constraints described in the 'Offline' column of Table 4.2 for forward and reverse primers.

The next three rounds perform homology tests on the resultant candidate primers against the entire CCDS database. It extracts all possible subsequences from the database as candidate primers and compares all possible pairs among them for homology tests. This requires large-scale computation on a tremendous number of pairs. The 5’ cross-hybridization filtering round (Round 3) eliminates candidate primers that are the same as any subsequence of an off-target sequence at the 3’ end and has only a few mismatches (up to four mismatches) at the 5’ end, and thus might cross-hybridize with an off-target sequence due to their high complementarity, especially at the 3’ end. The general cross-hybridization filtering round (Round 4) eliminates candidate primers that are similar to any subsequence of an off-target sequence (up to two mismatches anywhere). The duplicate removal round (Round 5) eliminates false-positive primers that still violate the general cross-hybridization filtering constraint. Rounds 4 and 5 are iterated until the checking of the general cross-hybridization filtering constraint is completed. The details of offline processing algorithm flow are shown in Figure 4.3. The large-scale computation of each round of homology testing relies on distributed data processing in MapReduce.

Figure 4.2 explains how MRPrimerW eliminates primers that are homologous to off-targets. In the figure below, we assume there are four 20-mer candidate primers of (a)-(d) that occur in different sequences from each other. In terms of the candidate primer (a), other candidate primers (b) and (c) are homologous with (a), and so, they are all filtered out in homology tests. The candidate primer (d) is not homologous with (a), and so, passes the homology tests. Two kinds of homology tests are used: 5’ cross-hybridization and general cross-hybridization. The former test considers up to four mismatches (in nucleotide) at the 5’ end as being homologous, where the 5’ end is the starting point of primer annealing. In general, a primer can successfully amplify a target gene although its 5’ end is not exactly matched to target gene, if its 3’ end is exactly matched to the target. The candidate primer

(b) belongs to this case, and the red colored nucleotides in the primer indicate the mismatches. Here, the candidate primer (b) might amplify not only Sequence j, but also Sequence i, and thus, should be filtered out. The latter test considers up to two mismatches anywhere as being homologous. The candidate primer (c) belongs to this case. However, candidate primer (d) does not belong to either of these cases. The offline processing part of MRPrimerW performs this kind of homology tests over all possible pairs between all possible subsequences, which are extracted in a sliding window manner from the database.

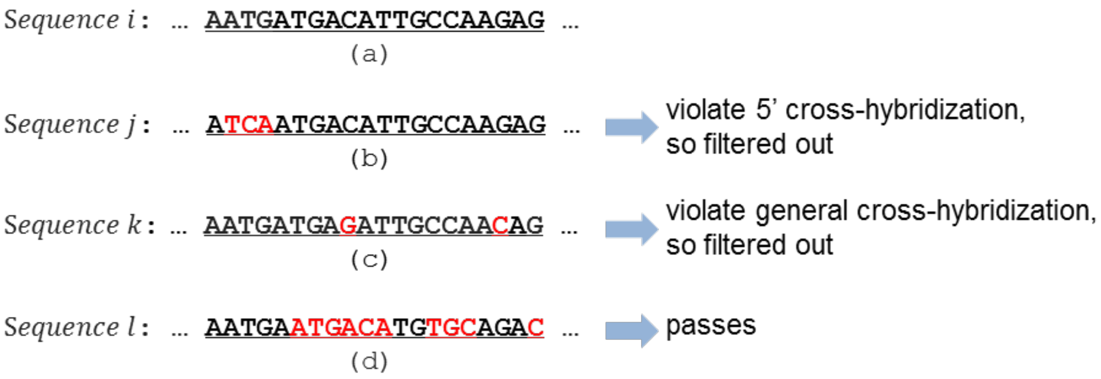

Sequence i : ... AATGATGACATTGCCAAGAG ...
(a)
Sequence j : ... ATCAATGACATTGCCAAGAG ...
(b)
violate 5' cross-hybridization, so filtered out
Sequence k : ... AATGATGAGATTGCCAACAG ...
(c)
violate general cross-hybridization, so filtered out
Sequence l : ... AATGAATGACATGTGCAGAC ...
(d)
passes

Figure 4.2. An example of how MRPrimerW eliminates primers that are homologous to off-targets.

For TaqMap probes, we performed the same offline processing algorithm flow with a different set of filtering constraints (Table 4.3). In detail, we have extracted all possible candidate TaqMan probes satisfying both the TaqMan probe constraints (Table 4.3) and homology tests from the database. Then, we have loaded the candidate probes into a TaqMan probe index in the main memory. If a user selects the TaqMan probe design option in the query web page, MRPrimerW returns a TaqMan probe for each target gene.

Table 4.3. Summary of the filtering constraints used for TaqMan probe design.

Parameter	Value range
primer length	18–30 bp
melting temperature (TM)	68–70℃
GC content	30–80%
self-complementarity	<5-mer
3' self-complementarity	<4-mer
Contiguous residue	<6-mer
Gibbs free energy (ΔG)	≥-9 *kcal/mol*
Hairpin	<3-mer

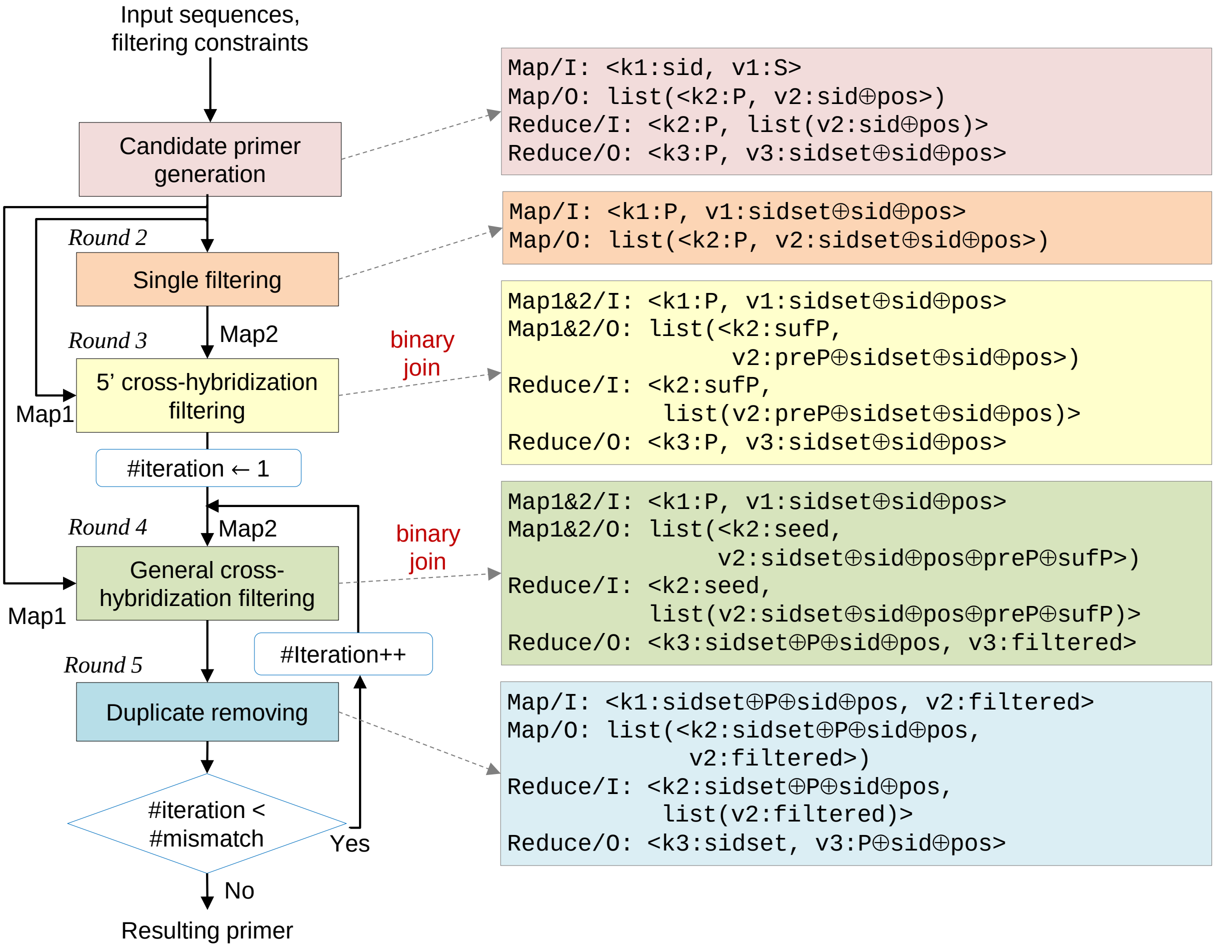

Input sequences,
filtering constraints
Candidate primer generation
Round 2
Single filtering
Round 3
Map2
binary join
5' cross-hybridization filtering
Map1
#iteration ← 1
Round 4
Map2
binary join
General cross-hybridization filtering
Map1
#Iteration++
Round 5
Duplicate removing
#iteration < #mismatch
Yes
No
Resulting primer
Map/I: <k1:sid, v1:S>
Map/O: list(<k2:P, v2:sid⊕pos>)
Reduce/I: <k2:P, list(v2:sid⊕pos)>
Reduce/O: <k3:P, v3:sidset⊕sid⊕pos>
Map/I: <k1:P, v1:sidset⊕sid⊕pos>
Map/O: list(<k2:P, v2:sidset⊕sid⊕pos>)
Map1&2/I: <k1:P, v1:sidset⊕sid⊕pos>
Map1&2/O: list(<k2:sufP, v2:preP⊕sidset⊕sid⊕pos>)
Reduce/I: <k2:sufP, list(v2:preP⊕sidset⊕sid⊕pos)>
Reduce/O: <k3:P, v3:sidset⊕sid⊕pos>
Map1&2/I: <k1:P, v1:sidset⊕sid⊕pos>
Map1&2/O: list(<k2:seed, v2:sidset⊕sid⊕pos⊕preP⊕sufP>)
Reduce/I: <k2:seed, list(v2:sidset⊕sid⊕pos⊕preP⊕sufP)>
Reduce/O: <k3:sidset⊕P⊕sid⊕pos, v3:filtered>
Map/I: <k1:sidset⊕P⊕sid⊕pos, v2:filtered>
Map/O: list(<k2:sidset⊕P⊕sid⊕pos, v2:filtered>)
Reduce/I: <k2:sidset⊕P⊕sid⊕pos, list(v2:filtered)>
Reduce/O: <k3:sidset, v3:P⊕sid⊕pos>

Figure 4.3. Overall flow of the five-found MRPrimerW offline processing method.

4.3 Index building part

After offline processing, we create a set of indices based on the results, which are then loaded into the main memory of the web server for online processing (Figure 4.1B). The detailed structures of the indices are illustrated in Figure 4.4. Nine indices are built: seven gene annotation indices (A-C), one primer index (D), one probe index, and one cached primer pair index (E). All indices follow the structure of a key–value database, in which each row is a pair of key and value. Annotation data were downloaded from GenBank ftp (ftp://ftp.ncbi.nlm.nih.gov/genomes/).

Online processing supports six kinds of queries (i.e., ‘Search by’ options) including NCBI gene symbol, NCBI CCDS ID, NCBI gene ID, GenBank accession number, GenBank alias, and keyword. Accordingly, six partial annotation indices are used for the various query types (Figure 4.4A and 4.4B). The six annotation indices include (4.4A) GenBank Accession number and NCBI CCDS ID as hash structure indices with unique identified annotation, and (4.4B) NCBI gene symbol, NCBI gene ID, GenBank aliases, and keyword (gene description) as list structure indices having duplicated annotation. The key, portions of the indices formatted as “Species:searchtype:query,” are used for matching with user queries. For instance, if a user sets the query type to ‘NCBI Gene Symbol’ and inputs “Adcy6 Anxa2 Cacna1c” in the text field of the website, those three symbols are used to match with key portions of the corresponding index. The value portions of the indices are single sequence IDs (sids) or lists of sids in which the key occurs in the full annotation index and the primer index.

The full annotation index, which simply combines all six kinds of annotation information, is used to generate the resulting web page (Figure 4.4C). The primer index contains primer sequences and positions (Figure 4.4D). The hash index contains the primer sequence and position in the sequence of the sid. The key portion, formatted as Species:sid+len(*), where len is primer length and * tag refers to the array of reverse primers, is a pair of sid and primer length, and the value portion, an array of primer data with p (primer sequence), sid, and pos (position) concatenating + tag, is a pair consisting of the primer sequence and the <sid, pos> where the primer sequence occurs. For example, when a user inputs gene symbol ‘Olfr156,’ MRPrimerW first accesses the partial annotation index for NCBI Gene Symbol and finds a sid corresponding to ‘Olfr156’. Then, using the sid as the key, MRPrimerW retrieves a set of candidate primers, especially their sequences and positions, from the primer index, which are subjected to online processing. The structure of the probe index is the same as (4.4D), but the content is a little different from that of the primer index. The key is formatted as Species:taqman:sid, and the value is the array of probe data of probe+sid+pos.

The cached top-1 primer pairs hash index contains pre-computed top-1 primer pairs for each target (4.4E). The key is formatted as Species:top:sid and the value is the array of primer data formatted f.p (forward primer), f.pos (forward position of f.p), r.p (reverse primer), and r.pos (reverse position of r.p) in the sequence sid.

In addition, for the case of the set of user queries that may be amplified by relaxing filtering constraints, MRPrimerW returns a list of the filtering constraints and how the constraints should be adjusted. We build constraints metadata index which contains minimum and maximum values for each single filtering constraints for all target genes. The key portion is formatted as Species:meta:sid and the value is the array of constraints metadata concatenating + tag. Before retrieving all candidate primers containing the user query from indices, constraints metadata are retrieved and examined whether the values of user given single filtering constraints are between the minimum and maximum values of metadata. If not, MRPrimerW returns the single filtering constraints and suggestion values. For pair filtering constraints, the metadata can be varied depending on single filtering constraints. The pair filtering constraints metadata is calculated while performing pair filtering step without loading index. If the pair filtering step fails to design primer pairs, MRPrimerW outputs the pair filtering constraints and how the constraints should be adjusted.

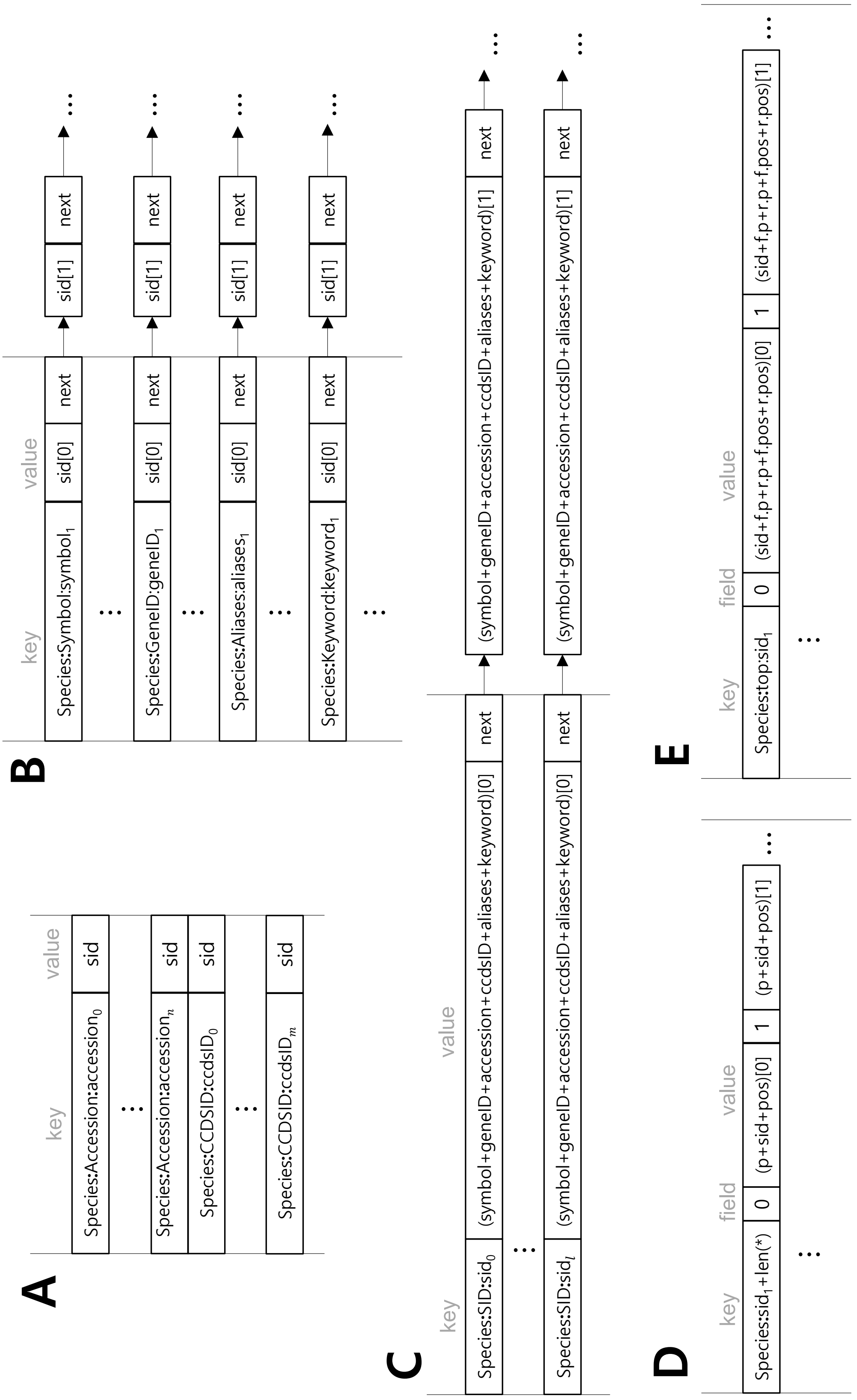

A
key
value
Species:Accession:accession$_0$
sid
…
Species:Accession:accession$_n$
sid
Species:CCDSID:ccdsID$_0$
sid
…
Species:CCDSID:ccdsID$_m$
sid
B
key
value
Species:Symbol:symbol$_1$
sid[0]
next
sid[1]
next
…
Species:GeneID:geneID$_1$
sid[0]
next
sid[1]
next
…
Species:Aliases:aliases$_1$
sid[0]
next
sid[1]
next
…
Species:Keyword:keyword$_1$
sid[0]
next
sid[1]
next
…
C
key
value
Species:SID:sid$_0$
(symbol+geneID+accession+ccdsID+aliases+keyword)[0]
next
(symbol+geneID+accession+ccdsID+aliases+keyword)[1]
next
…
Species:SID:sid$_l$
(symbol+geneID+accession+ccdsID+aliases+keyword)[0]
next
(symbol+geneID+accession+ccdsID+aliases+keyword)[1]
next
D
key
field
value
Species:sid$_1$+len(*)
0
(p+sid+pos)[0]
1
(p+sid+pos)[1]
…
E
key
field
value
Species:top:sid$_1$
0
(sid+f.p+r.p+f.pos+r.pos)[0]
1
(sid+f.p+r.p+f.pos+r.pos)[1]
…

Figure 4.4. Structures of indices used in MRPrimerW web server.

4.4 Online processing part

Online processing consists of three steps that check the filtering constraints provided by the users and rank the primers to return the top-1 best primers (Figure 4.1C). The first step takes the user query and uses the indices to retrieve all candidate primers containing the query. While extracting the candidates, MRPrimerW applies eight filtering constraints for each primer, described in Table 4.2. Here, the constraints for length, melting temperature, GC content, and contiguous residue must be within the range pre-defined in offline processing. Figure 4.5 illustrates the flowchart of the searching and single filtering step.

The second step applies five filtering constraints for primer pairs, described in Table 4.2. For this purpose, MRPrimerW performs a self-join computation on each group of candidate primers from the same target sequence, i.e., it joins forward primers and backward primers into the same group. This pair-filtering step may take a long time if the length of the input query (i.e., the number of gene symbols) is long or the number of candidate primers retrieved is very large. Figure 4.6 shows the flowchart of the pair filtering step.

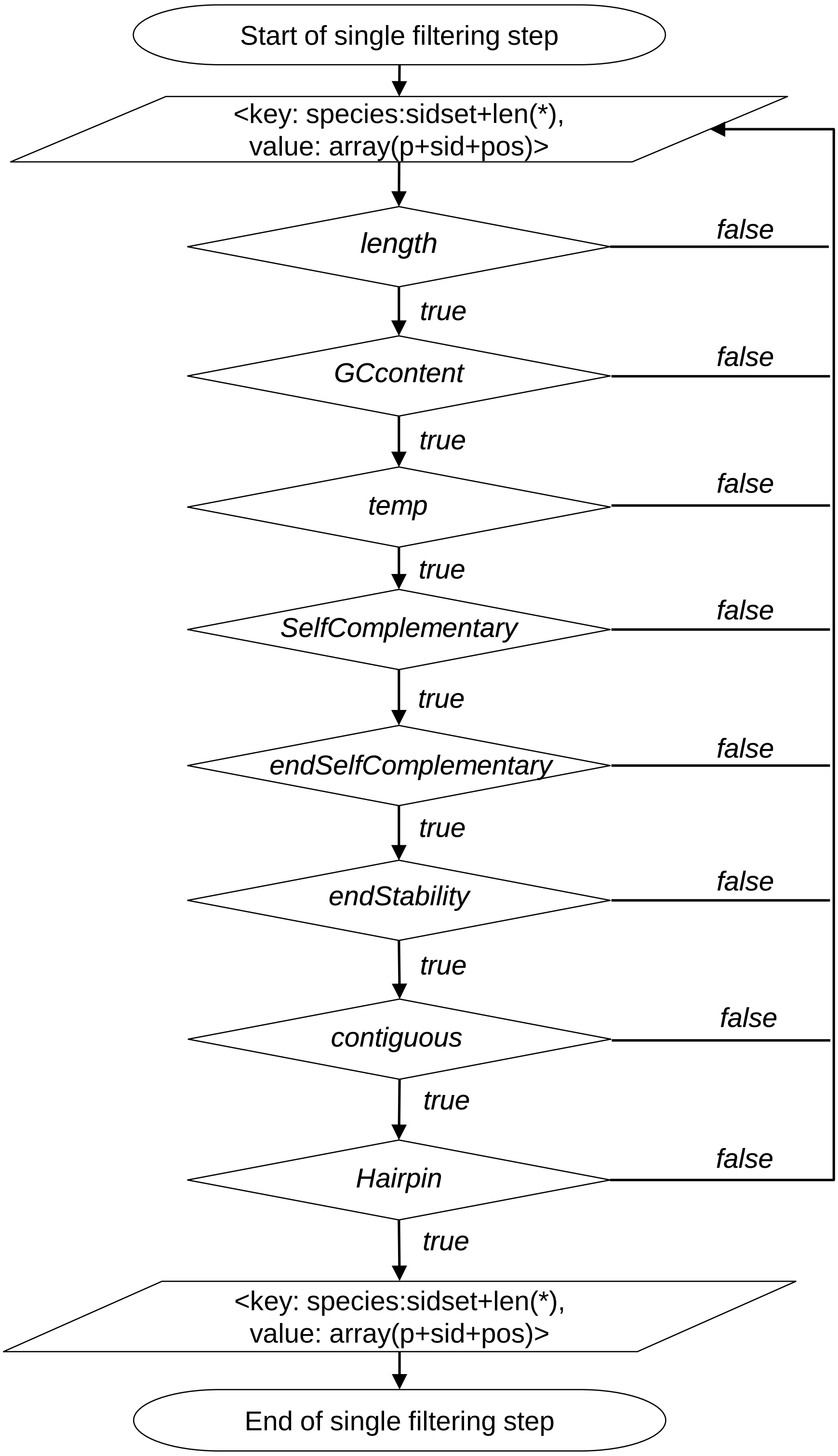

Start of single filtering step
<key: species:sidset+len(*), value: array(p+sid+pos)>
length
GCcontent
temp
SelfComplementary
endSelfComplementary
endStability
contiguous
Hairpin
true
false
<key: species:sidset+len(*), value: array(p+sid+pos)>
End of single filtering step

Figure 4.5. Flowchart of searching and single filtering step.

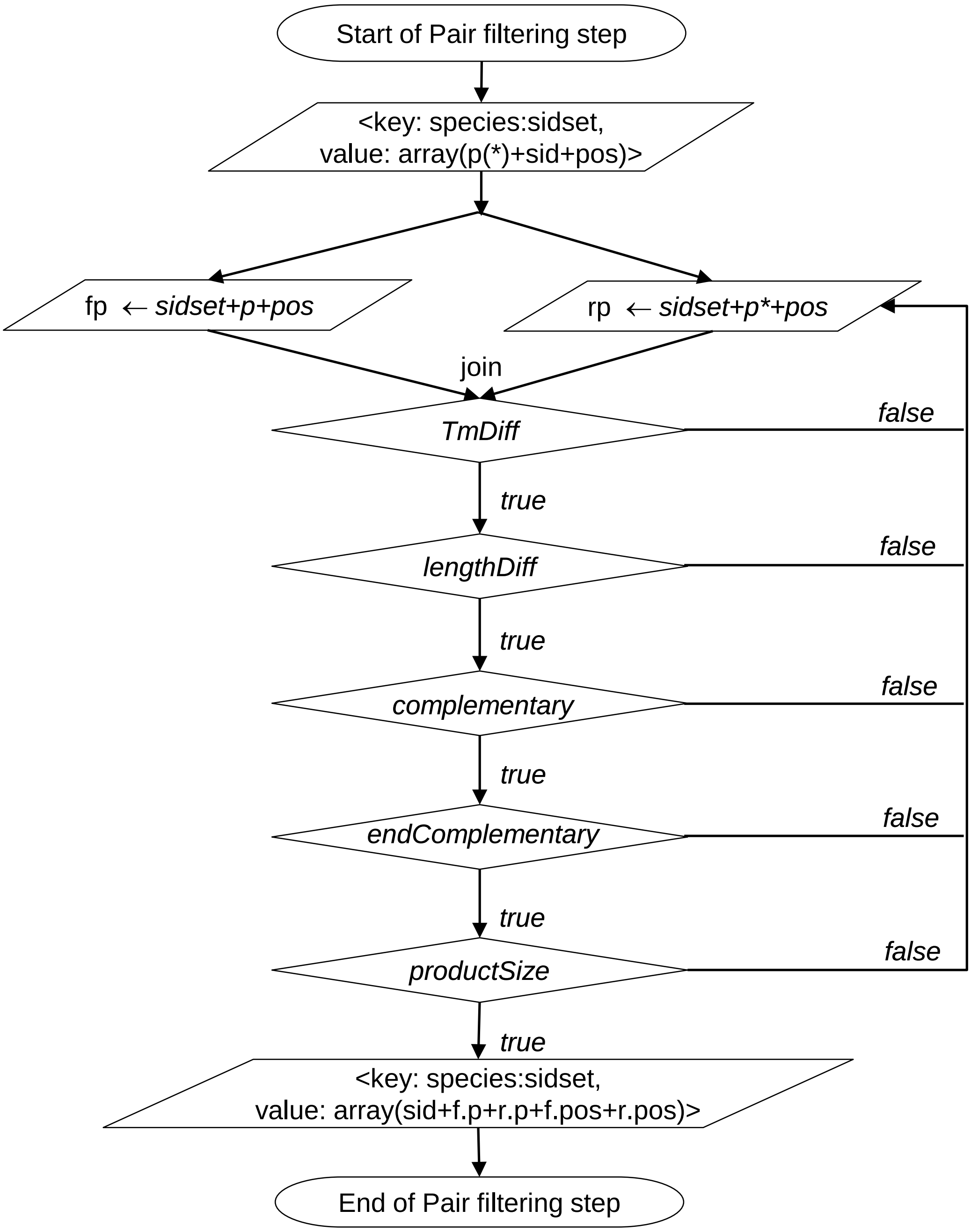


Figure 4.6. Flowchart of pair filtering step.

The final step calculates the penalty scores of the primers obtained in the previous step and sorts the primers according to their scores. Then, it returns the top-1 best primer, i.e., the primer with the lowest penalty score, for each target sequence. The penalty score is calculated according to the method used in Primer3Plus [21]. Figure 4.7 shows the flowchart of output sorting step. If the user inputs 12 target genes, MRPrimerW shows the 12 top hits, which can be used for qPCR experiment in most cases. However, if some of the target genes have no top-1 best primers, the user can relax the filtering constraints (i.e., using Advanced Settings) and click the search button to design a set of top primers that satisfy the same stringent filtering constraints and are target gene–specific. If a user selects the TaqMan probe design option, MRPrimerW returns a TaqMan probe for each target gene, where the probe is located between forward and reverse primers. Since in many cases users do not change default settings on filtering constraints, the response time can be improved by using the cached top-1 primer pairs index, which stores the top-1 primer pairs under the default setting for sequences of the database in key-value format in the main memory of the web server (Figure 4.4E).

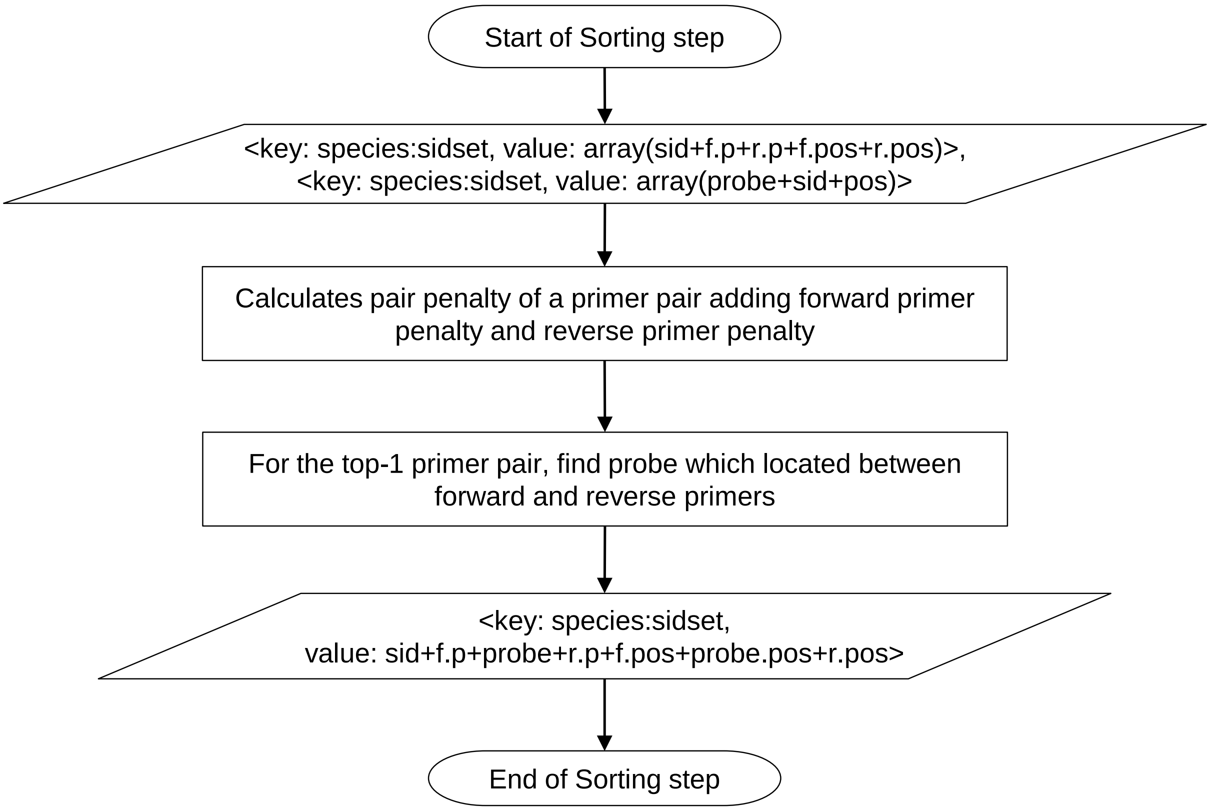

Start of Sorting step
<key: species:sidset, value: array(sid+f.p+r.p+f.pos+r.pos)>,
<key: species:sidset, value: array(probe+sid+pos)>
Calculates pair penalty of a primer pair adding forward primer penalty and reverse primer penalty
For the top-1 primer pair, find probe which located between forward and reverse primers
<key: species:sidset,
value: sid+f.p+probe+r.p+f.pos+probe.pos+r.pos>
End of Sorting step

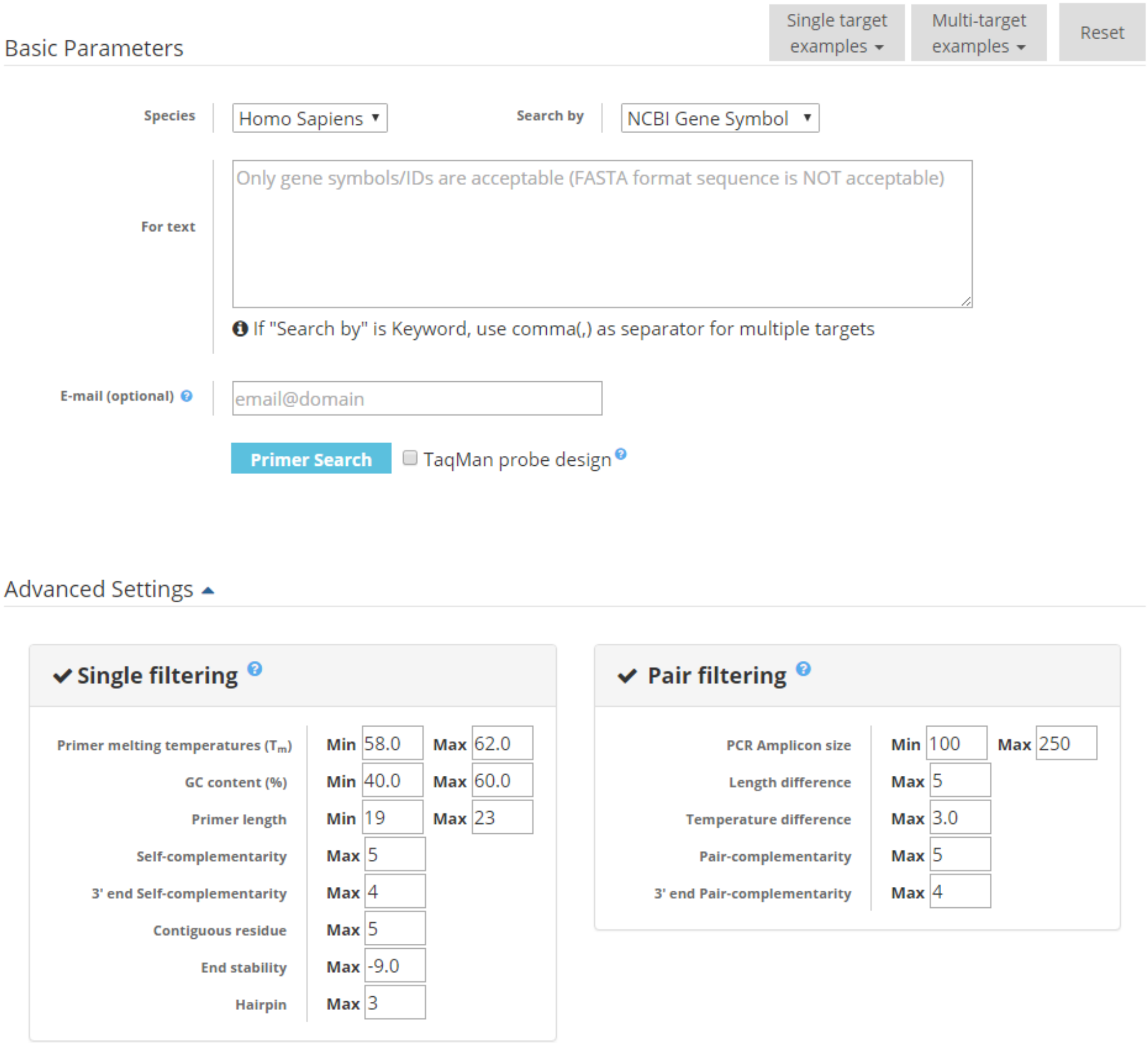
Figure 4.7. Flowchart of output sorting step.

4.5 Web interface

The MRPrimerW web server is implemented using Redis (http://redis.io/), an in-memory key–value store, for data management. Redis supports various kinds of data structures for various types of values, including string, hash, list, set, and sorted set. Among these, MRPrimerW uses hash and set for annotation and primer indices (Figure 4.4). In detail, for the server side, we adopted phpredis (https://github.com/phpredis/phpredis) for communication between Redis and PHP, and AJAX (asynchronous JavaScript and XML) for client–server communication. For the client side, MRPrimerW generates web pages using HTML with CSS and bootstrap (http://getbootstrap.com/) for styling interactive user-interface components. For dynamic HTML behavior, we used JavaScript and jQuery.

MRPrimerW supports most major web browsers including Microsoft Internet Explorer, Google Chrome, Apple Safari, and Opera.

Figure 4.8 illustrates an example query of MRPrimerW. MRPrimerW allows the user to choose species (human or mouse) and query type (NCBI gene symbol, NCBI CCDS ID, NCBI gene ID, GenBank accession number, GenBank aliases, or keyword), and then enter the input query. The user can select the TaqMan probe design option to design TaqMan probes as well as SYBR Green primers. MRPrimerW also provides a feature that sends the query result to a user via email. If a user enters his/her email address in the query web page, the web server sends an email containing a link to the result page to the user after query processing is completed. The users do not need to wait to get a query result and the result page accessible via the link in the email is available for two weeks (i.e., 14 days). In Advanced Settings, the user can adjust single- and pair-filtering constraints. MRPrimerW provides six example queries for single target genes and another six example queries for multiple target genes, in particular, 24 genes related to signaling molecules.



Basic Parameters

Species: Homo Sapiens ▾

Search by: NCBI Gene Symbol ▾

For text: Only gene symbols/IDs are acceptable (FASTA format sequence is NOT acceptable)

If "Search by" is Keyword, use comma(,) as separator for multiple targets

E-mail (optional): email@domain

Advanced Settings ▴

✓ Single filtering

Primer melting temperatures (T_m)	Min	58.0	Max	62.0
GC content (%)	Min	40.0	Max	60.0
Primer length	Min	19	Max	23
Self-complementarity	Max	5		
3' end Self-complementarity	Max	4		
Contiguous residue	Max	5		
End stability	Max	-9.0		
Hairpin	Max	3		

✓ Pair filtering

PCR Amplicon size	Min	100	Max	250
Length difference	Max	5		
Temperature difference	Max	3.0		
Pair-complementarity	Max	5		
3' end Pair-complementarity	Max	4		

Figure 4.8. Input interface of MRPrimerW.

Figure 4.9 illustrates the results of the example query for nine target genes (SAMD11, TNF, IL10, TP53, A1CF, UBE2J2, HES4, THDP1, KFK2), where the species is human and the search type is NCBI Gene Symbol. Among nine target genes, we assume that three target genes will have primer pairs that amplify each of them solely (case 1: IL10, SAMD11, TNF), two target genes will have only less target-specific primer pairs that may amplify multiple targets (case 2: TP53, A1CF), two target genes will have target-specific primer pairs, but the given filtering constraints are too strict to return them (case 3: UBE2J2, HES4), and two target genes will have typos in their symbols (case 4: THDP1, KFK2).

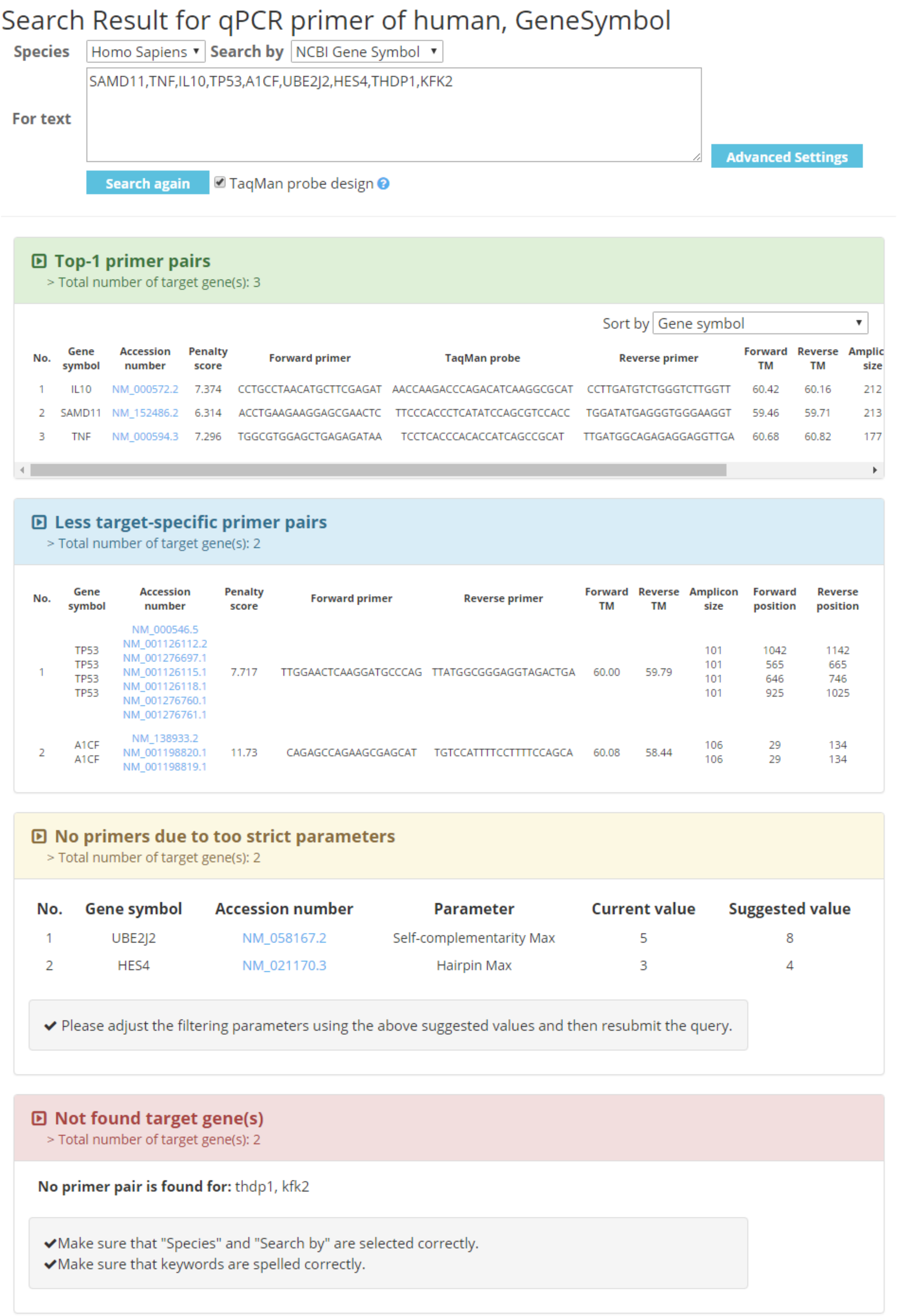
Then, resultant web page shows four tables, each of which contains the primers for each of the above cases. In detail, the first table (for case 1) shows three top-1 primer pairs satisfying the same stringent and uniform constraints. The table shows forward and backward primer sequences, TaqMan probe sequences, gene symbol, GenBank accession number (with a link to detailed gene information from GenBank and primer information), penalty score, melting temperatures (TM), amplicon size, and primer positions.

The second table (for case 2) shows a set of less target-specific primers that may amplify multiple targets for two target genes. In other words, there is no target-specific primer for the target gene. The indices of MRPrimerW contain both the primers amplifying a single target and the primers that may amplify multiple targets (i.e., less specific). Highly homologue genes (e.g. TP53, A1CF) do not have any primer pair that amplifies the corresponding gene solely. The approach that does not show the results may be too strict, especially compared with the existing tools. For such target genes, the existing tools may return some results because they cannot perform homology tests on the entire set of genes. We think that even less target-specific primers, which may amplify multiple targets, could have practical use in qPCR experiments. Thus, users can use such less target-specific primers if and when necessary. The format of the second table is the same as that of the first table.

The third table (for case 3) shows the set of genes that have the top-1 primer pair that amplifies the target gene solely but may be amplified by relaxing filtering constraints and how the constraints should be adjusted for each gene. MRPrimerW returns a list of filtering constraints that require adjustment. How the constraints should be adjusted for each of the target genes belonging to this case is shown in the third table in the output web page.

The fourth table (for case 4) shows the set of genes given by the user that may be wrong or have typos so that cannot be identified in the annotation indices of MRPrimerW. With this information, users can modify their queries or input parameters to obtain primers for query genes having no results.

In addition, the headline of each table shows the number of query genes belonging to the table. For example, among 100 query genes, if 80 genes have suitable target-specific primer pairs, 10 genes have less target-specific primer pairs that may amplify other genes, 7 have no results because the parameters are too strict, and 3 have no results because of typos. The headlines of four result tables show 80, 10, 7, and 3, respectively, and so, users can easily figure out the problematic query genes.



Search Result for qPCR primer of human, GeneSymbol

Species: Homo Sapiens ▾ Search by: NCBI Gene Symbol ▾

For text: SAMD11,TNF,IL10,TP53,A1CF,UBE2J2,HES4,THDP1,KFK2

Top-1 primer pairs

> Total number of target gene(s): 3

Sort by: Gene symbol ▾

No.	Gene symbol	Accession number	Penalty score	Forward primer	TaqMan probe	Reverse primer	Forward TM	Reverse TM	Amplic size
1	IL10	NM_000572.2	7.374	CCTGCCTAACATGCTTCGAGAT	AACCAAGACCCAGACATCAAGGCGCAT	CCTTGATGTCTGGGTCTTGGTT	60.42	60.16	212
2	SAMD11	NM_152486.2	6.314	ACCTGAAGAAGGAGCGAACTC	TTCCCACCCTCATATCCAGCGTCCACC	TGGATATGAGGGTGGGAAGGT	59.46	59.71	213
3	TNF	NM_000594.3	7.296	TGGCGTGGAGCTGAGAGATAA	TCCTCACCCACACCATCAGCCGCAT	TTGATGGCAGAGAGGAGGTTGA	60.68	60.82	177

Less target-specific primer pairs

> Total number of target gene(s): 2

No.	Gene symbol	Accession number	Penalty score	Forward primer	Reverse primer	Forward TM	Reverse TM	Amplicon size	Forward position	Reverse position
1	TP53 TP53 TP53 TP53	NM_000546.5 NM_001126112.2 NM_001276697.1 NM_001126115.1 NM_001126118.1 NM_001276760.1 NM_001276761.1	7.717	TTGGAACTCAAGGATGCCCAG	TTATGGCGGGAGGTAGACTGA	60.00	59.79	101 101 101 101	1042 565 646 925	1142 665 746 1025
2	A1CF A1CF	NM_138933.2 NM_001198820.1 NM_001198819.1	11.73	CAGAGCCAGAAGCGAGCAT	TGTCCATTTTCCTTTTCCAGCA	60.08	58.44	106 106	29 29	134 134

No primers due to too strict parameters

> Total number of target gene(s): 2

No.	Gene symbol	Accession number	Parameter	Current value	Suggested value
1	UBE2J2	NM_058167.2	Self-complementarity Max	5	8
2	HES4	NM_021170.3	Hairpin Max	3	4

✓ Please adjust the filtering parameters using the above suggested values and then resubmit the query.

Not found target gene(s)

> Total number of target gene(s): 2

No primer pair is found for: thdp1, kfk2

✓Make sure that "Species" and "Search by" are selected correctly.
✓Make sure that keywords are spelled correctly.

Figure 4.9. Output interface of MRPrimerW.

V. CONCLUSIONS

In this dissertation, we proposed MRPrimer and MRPrimerW that could overcome the drawbacks of existing design methods, while also integrating several desirable features required by researchers in this field into a single method. Design of high-quality primers for multiple target sequences is essential for qPCR experiments, but is a challenging due to the need to consider both homology tests on off-target sequences and the same stringent filtering constraints on the primers.

In Chapter 3, we propose MRPrimer that can design all possible feasible and valid primer pairs for an entire DNA database. The seven steps of MRPrimer are following. Step 1 receives a given DNA sequence database to extract partial sequences for candidate primers having all possible lengths between the minimum length and the maximum length Step 2 excluds the primers which do not satisfy input single filtering conditions when the candidate primers extracted in Step 1 are subjected to the single filtering conditions. Step 3 performs pair-joining Map1, which includes all the possible partial sequences obtained in Step 1, and Map2, which includes candidate primer sets satisfying the single filtering conditions obtained in Step 2, and removing the primers for Map2 when the primers for Map1 and Map2 have the same sequences other than the 5' termini thereof. Step 4 performs pair-joining Map1, which includes all the possible partial sequences obtained in Step 1, and Map2, which includes candidate primer sets satisfying the single filtering conditions and 5’ cross-hybridization filtering conditions obtained in Step 3, and removing the primers for Map2 when the primers for Map1 have the same sequences as the primers for Map2 except the sequences having a given mismatch number (#mismatch). Step 5 removs false-positive primers which still remain from the results of Step 4 and do not satisfy general cross-hybridization filtering conditions. Step 6 divides the primers remaining from the results of Step 5 into forward primer sets and reverse primer sets and excluding the primers which do not satisfy the filtering conditions for self-join calculation when the divided forward and reverse primer sets are subjected to the filtering conditions. Finally, Step 7 calculates penalty scores for the primer pairs passing Step 6, and sorting the primer pairs in the same sidset groups according to the calculated penalty scores (Step 7).

Our biological and computational validation results in Section 3.3 and 3.4 indicate that the resultant primers are very useful and effective for qPCR and sequencing analyses. We can summarize its major advantages in terms of practical usage as follows.

First, MRPrimer performs both single/pair primer filtering and homology tests, in a combined and integrated manner. Furthermore, it automatically sorts the resulting primer pairs for each target sequence, based on penalty scores. Thus, users do not need to be concerned about mistakes when validating a candidate primer. Because it produces a complete set of primer pairs, users can repeatedly reuse the results, unless filtering constraints need to be changed.

Second, MRPrimer designs all feasible primer pairs strictly, following the same filtering constraints. For example, it can design a large number of primers that follow a very strict constraint on product size (e.g., between 100 and 150 bp) for a given set of tens of thousands of sequences all at once. This powerful feature would be especially useful for qPCR experiments.

Third, MRPrimer is computationally efficient and scalable, and able to design entire primer pairs for a whole DNA database within a few hours using only a small-scale cluster of PCs. Even for a database of 105,180 DNA sequences, it could design all primer pairs within 7 hours. This feature is very useful, especially for sequence databases that are updated frequently, like the RefSeq database.

In Chapter 4, we proposed MRPrimerW web server, a straightforward but powerful tool for designing high-quality primer pairs that can be used simultaneously to detect multiple target genes in qPCR experiments. MRPrimerW overcomes the major drawbacks of existing web servers for primer design by enabling users to freely adjust filtering constraints, performing complete homology tests, supporting batch designing for qPCR, supporting TaqMan probe design, and supporting ranking of primers.

These powerful features were achieved by performing large-scale computation for homology testing on all possible candidate primers in an exact manner, and then materializing the resultant valid primers in eight kinds of indices in the main memory of the web server. Based on these indices, the web server quickly performs online processing in three steps and returns a complete set of the top primer pairs corresponding to the user’s query.

The current version of MRPrimerW is built based on the CCDS database, a collection of coding sequences from human and mouse. We think it is important to include other popular model organisms such as rat, monkey, Arabidopsis, yeast, nematodes, and bacteria. Thus, we have a plan to do this in future work.

In conclusion, we believe that we have developed an advanced technology, a straightforward but powerful method for designing high-quality primer pairs and for increasing the efficiency and specificity of experiments involving PCR. We also believe that MRPrimer, MRPrimerW or a variation of MRPrimer could be very useful for other application areas such as DNA construction and genetic engineering. If there is a specific fragment to be amplified from a given DNA template, there are a variety of putative primers that could accomplish this. An MRPrimer-style method of screening and ranking in parallel could be very effective at designing ideal primer pairs for that purpose. For example, it could be effectively used to alleviate the problem of lack of novel primer pairs for detecting unauthorized genetically modified organisms (GMOs) in the collection of GMO detection methods, called GMO Detection method Database (GMDD) [59, 60].

REFERNECES

[1] T. M. Rose, E. R. Schultz, J. G. Henikoff, S. Pietrokovski, C. M. McCallum, and S. Henikoff, "Consensus-degenerate hybrid oligonucleotide primers for amplification of distantly related sequences," *Nucleic acids research,* vol. 26, pp. 1628-1635, 1998.

[2] E. Gorrón, F. Rodríguez, D. Bernal, L. M. Rodriguez-Rojas, A. Bernal, S. Restrepo*, et al.*, "A new method for designing degenerate primers and its use in the identification of sequences in Brachiaria showing similarity to apomixis-associated genes," *Bioinformatics,* vol. 26, pp. 2053-2054, 2010.

[3] B. Dwivedi, R. Schmieder, D. B. Goldsmith, R. A. Edwards, and M. Breitbart, "PhiSiGns: an online tool to identify signature genes in phages and design PCR primers for examining phage diversity," *BMC bioinformatics,* vol. 13, p. 37, 2012.

[4] L.-Y. Chuang, Y.-H. Cheng, and C.-H. Yang, "Specific primer design for the polymerase chain reaction," *Biotechnology letters,* vol. 35, pp. 1541-1549, 2013.

[5] J. Huang, I. Khan, R. Liu, Y. Yang, and N. Zhu, "Single primer-mediated circular PCR for hairpin DNA cloning and plasmid editing," *Anal Biochem,* p. aheadofprint, Jan 11 2016.

[6] E. Meglécz, C. Costedoat, V. Dubut, A. Gilles, T. Malausa, N. Pech, *et al.*, "QDD: a user-friendly program to select microsatellite markers and design primers from large sequencing projects," *Bioinformatics,* vol. 26, pp. 403-404, 2010.

[7] X. Wang, A. Spandidos, H. Wang, and B. Seed, "PrimerBank: a PCR primer database for quantitative gene expression analysis, 2012 update," *Nucleic acids research,* vol. 40, pp. D1144-D1149, 2012.

[8] A. Spandidos, X. Wang, H. Wang, and B. Seed, "PrimerBank: a resource of human and mouse PCR primer pairs for gene expression detection and quantification," *Nucleic acids research,* vol. 38, pp. D792-D799, 2010.

[9] J.-S. Wu, C. Lee, C.-C. Wu, and Y.-L. Shiue, "Primer design using genetic algorithm," *Bioinformatics,* vol. 20, pp. 1710-1717, 2004.

[10] H. Cao and J. M. Shockey, "Comparison of TaqMan and SYBR Green qPCR methods for quantitative gene expression in tung tree tissues," *J Agric Food Chem,* vol. 60, pp. 12296-303, Dec 19 2012.

[11] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *J Comput Biol,* vol. 1, pp. 337-48, Winter 1994.

[12] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res,* vol. 22,

pp. 4673-80, Nov 11 1994.

[13] S. N. Gardner, A. L. Hiddessen, P. L. Williams, C. Hara, M. C. Wagner, and B. W. Colston, Jr., "Multiplex primer prediction software for divergent targets," *Nucleic Acids Res,* vol. 37, pp. 6291-304, Oct 2009.

[14] C. Linhart and R. Shamir, "The degenerate primer design problem: theory and applications," *Journal of Computational Biology,* vol. 12, pp. 431-456, 2005.

[15] J. A. Iserte, B. I. Stephan, S. E. Goni, C. S. Borio, P. D. Ghiringhelli, and M. E. Lozano, "Family-specific degenerate primer design: a tool to design consensus degenerated oligonucleotides," *Biotechnol Res Int,* vol. 2013, p. 383646, 2013.

[16] G. Baker, J. Smith, and D. A. Cowan, "Review and re-analysis of domain-specific 16S primers," *Journal of Microbiological Methods,* vol. 55, pp. 541-555, 2003.

[17] H. Mori, F. Maruyama, H. Kato, A. Toyoda, A. Dozono, Y. Ohtsubo*, et al.*, "Design and Experimental Application of a Novel Non-Degenerate Universal Primer Set that Amplifies Prokaryotic 16S rRNA Genes with a Low Possibility to Amplify Eukaryotic rRNA Genes," *DNA Research,* p. dst052, 2013.

[18] P. S. Kumar, M. R. Brooker, S. E. Dowd, and T. Camerlengo, "Target region selection is a critical determinant of community fingerprints generated by 16S pyrosequencing," *PloS one,* vol. 6, p. e20956, 2011.

[19] L. Giacomucci, K. Purdy, E. Zanardini, A. Polo, and F. Cappitelli, "A new non-

degenerate primer pair for the specific detection of the nitrite reductase gene nrfA in the genus desulfovibrio," *Journal of molecular microbiology and biotechnology,* vol. 22, pp. 345-351, 2012.

[20] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, pp. 107-113, 2008.

[21] A. Untergasser, I. Cutcutache, T. Koressaar, J. Ye, B. C. Faircloth, M. Remm, *et al.*, "Primer3--new capabilities and interfaces," *Nucleic Acids Res,* vol. 40, p. e115, Aug 2012.

[22] J. Ye, G. Coulouris, I. Zaretskaya, I. Cutcutache, S. Rozen, and T. L. Madden, "Primer-BLAST: a tool to design target-specific primers for polymerase chain reaction," *BMC Bioinformatics,* vol. 13, p. 134, 2012.

[23] S. Rozen and H. Skaletsky, "Primer3 on the WWW for general users and for biologist programmers," in *Bioinformatics methods and protocols,* ed: Springer, 1999, pp. 365-386.

[24] T. M. Rose, J. G. Henikoff, and S. Henikoff, "CODEHOP (COnsensus-DEgenerate hybrid oligonucleotide primer) PCR primer design," *Nucleic Acids Research,* vol. 31, pp. 3763-3766, 2003.

[25] R. Boyce, P. Chilana, and T. M. Rose, "iCODEHOP: a new interactive program for designing COnsensus-DEgenerate Hybrid Oligonucleotide Primers from multiply

aligned protein sequences," *Nucleic acids research,* vol. 37, pp. W222-W228, 2009.

[26] R. Giegerich, F. Meyer, and C. Schleiermacher, "GeneFisher--software support for the detection of postulated genes," *Proc Int Conf Intell Syst Mol Biol,* vol. 4, pp. 68-77, 1996.

[27] D. Hagemeier, "GeneFisher2-an AJAX based implementation of GeneFisher," *Bachelor's thesis University Bielefeld, Faculty of Technology,* 2006.

[28] X. Wei, D. N. Kuhn, and G. Narasimhan, "Degenerate primer design via clustering," in *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, 2003, pp. 75-83.

[29] S. N. Jarman, "Amplicon: software for designing PCR primers on aligned DNA sequences," *Bioinformatics,* vol. 20, pp. 1644-1645, 2004.

[30] O. J. Jabado, G. Palacios, V. Kapoor, J. Hui, N. Renwick, J. Zhai*, et al.*, "Greene SCPrimer: a rapid comprehensive tool for designing degenerate primers from multiple sequence alignments," *Nucleic acids research,* vol. 34, pp. 6605-6611, 2006.

[31] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li*, et al.*, "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega," *Molecular systems biology,* vol. 7, 2011.

[32] A. Untergasser, H. Nijveen, X. Rao, T. Bisseling, R. Geurts, and J. A. Leunissen, "Primer3Plus, an enhanced web interface to Primer3," *Nucleic Acids Res,* vol. 35, pp.

W71-4, Jul 2007.

[33] F. M. You, N. Huo, Y. Q. Gu, M. C. Luo, Y. Ma, D. Hane, *et al.*, "BatchPrimer3: a high throughput web application for PCR and sequencing primer design," *BMC Bioinformatics,* vol. 9, p. 253, 2008.

[34] J. Fredslund and M. Lange, "Primique: automatic design of specific PCR primers for each sequence in a family," *BMC Bioinformatics,* vol. 8, p. 369, 2007.

[35] S. Arvidsson, M. Kwasniewski, D. M. Riano-Pachon, and B. Mueller-Roeber, "QuantPrime--a flexible tool for reliable high-throughput primer design for quantitative PCR," *BMC Bioinformatics,* vol. 9, p. 465, 2008.

[36] S. Lefever, J. Vandesompele, F. Speleman, and F. Pattyn, "RTPrimerDB: the portal for real-time PCR primers and probes," *Nucleic Acids Res,* vol. 37, pp. D942-5, Jan 2009.

[37] F. Pattyn, F. Speleman, A. De Paepe, and J. Vandesompele, "RTPrimerDB: the real-time PCR primer and probe database," *Nucleic Acids Res,* vol. 31, pp. 122-3, Jan 1 2003.

[38] F. Pattyn, P. Robbrecht, A. De Paepe, F. Speleman, and J. Vandesompele, "RTPrimerDB: the real-time PCR primer and probe database, major update 2006," *Nucleic Acids Res,* vol. 34, pp. D684-8, Jan 1 2006.

[39] W. Cui, D. D. Taub, and K. Gardner, "qPrimerDepot: a primer database for quantitative real time PCR," *Nucleic Acids Res,* vol. 35, pp. D805-9, Jan 2007.

[40] J. SantaLucia, Jr. and D. Hicks, "The thermodynamics of DNA structural motifs," *Annu*

Rev Biophys Biomol Struct, vol. 33, pp. 415-40, 2004.

[41] J. SantaLucia, "A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics," *Proceedings of the National Academy of Sciences,* vol. 95, pp. 1460-1465, 1998.

[42] R. A. Baeza-Yates and C. H. Perleberg, "Fast and practical approximate string matching," *Information Processing Letters,* vol. 59, pp. 21-27, 1996.

[43] M. Kim, K. Whang, and J. Lee, "n-Gram/2L-approximation: a two-level n-gram inverted index structure for approximate string matching," *Computer Systems Science and Engineering,* vol. 22, p. 365, 2007.

[44] M.-S. Kim, K.-Y. Whang, J.-G. Lee, and M.-J. Lee, "n-Gram/2L: A space and time efficient two-level n-gram inverted index structure," in *Proceedings of the 31st international conference on Very large data bases,* 2005, pp. 325-336.

[45] A. Spandidos, X. Wang, H. Wang, S. Dragnev, T. Thurber, and B. Seed, "A comprehensive collection of experimentally validated primers for Polymerase Chain Reaction quantitation of murine transcript abundance," *BMC genomics,* vol. 9, p. 633, 2008.

[46] X. Wang and B. Seed, "A PCR primer bank for quantitative gene expression analysis," *Nucleic acids research,* vol. 31, pp. e154-e154, 2003.

[47] K. Pruitt, T. Tatusova, and J. Ostell, "The reference sequence (RefSeq) project," *The*

NCBI Handbook [Internet]. National Library of Medicine (US), National Center for Biotechnology Information, Bethesda, MD, 2002.

[48] T. Tatusova, S. Ciufo, B. Fedorov, K. O'Neill, and I. Tolstoy, "RefSeq microbial genomes database: new representation and annotation strategy," *Nucleic Acids Res,* vol. 42, pp. D553-9, 2014.

[49] K. D. Pruitt, G. R. Brown, S. M. Hiatt, F. Thibaud-Nissen, A. Astashyn, O. Ermolaeva, *et al.*, "RefSeq: an update on mammalian reference sequences," *Nucleic Acids Res,* vol. 42, pp. D756-63, 2014.

[50] K. D. Pruitt, J. Harrow, R. A. Harte, C. Wallin, M. Diekhans, D. R. Maglott, *et al.*, "The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes," *Genome research,* vol. 19, pp. 1316-1323, 2009.

[51] J. H. Fong, T. D. Murphy, and K. D. Pruitt, "Comparison of RefSeq protein-coding regions in human and vertebrate genomes," *BMC genomics,* vol. 14, p. 654, 2013.

[52] S. A. Bustin, V. Benes, J. A. Garson, J. Hellemans, J. Huggett, M. Kubista, *et al.*, "The MIQE guidelines: minimum information for publication of quantitative real-time PCR experiments," *Clinical chemistry,* vol. 55, pp. 611-622, 2009.

[53] Y. Niimura, "Olfactory receptor multigene family in vertebrates: from the viewpoint of evolutionary genomics," *Current genomics,* vol. 13, p. 103, 2012.

[54] N. Kang, H. Kim, Y. Jae, N. Lee, C. R. Ku, F. Margolis, *et al.*, "Olfactory marker protein expression is an indicator of olfactory receptor-associated events in non-olfactory tissues," *PLoS One,* vol. 10, p. e0116097, 2015.

[55] N. N. Kang, "Olfactory receptors in non-chemosensory tissues," *Biochemistry and Molecular Biology Reports,* vol. 45, pp. 612-622, 2012.

[56] K. D. Pruitt, J. Harrow, R. A. Harte, C. Wallin, M. Diekhans, D. R. Maglott, *et al.*, "The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes," *Genome Res,* vol. 19, pp. 1316-23, Jul 2009.

[57] C. M. Farrell, N. A. O'Leary, R. A. Harte, J. E. Loveland, L. G. Wilming, C. Wallin, *et al.*, "Current status and new features of the Consensus Coding Sequence database," *Nucleic Acids Res,* vol. 42, pp. D865-72, Jan 2014.

[58] R. A. Harte, C. M. Farrell, J. E. Loveland, M. M. Suner, L. Wilming, B. Aken, *et al.*, "Tracking and coordinating an international curation effort for the CCDS Project," *Database (Oxford),* vol. 2012, p. bas008, 2012.

[59] W. Dong, L. Yang, K. Shen, B. Kim, G. A. Kleter, H. J. Marvin, *et al.*, "GMDD: a database of GMO detection methods," *BMC bioinformatics,* vol. 9, p. 260, 2008.

[60] N. Marmiroli, E. Maestri, M. Gullì, A. Malcevschi, C. Peano, R. Bordoni, *et al.*, "Methods for detection of GMOs in food and feed," *Analytical and Bioanalytical*

Chemistry, vol. 392, pp. 369-384, 2008.

요 약 문

맵리듀스 기반의 특이성 조건을 만족하는 유효한 모든 프라이머들을 빠르게 디자인하는 방법

프라이머 디자인은 중합효소 연쇄반응법 (PCR)에 있어서 가장 기본이 되는 기술이며 일반적으로 사용되는 기술이다. 많은 방법들이 프라이머 디자인을 위해 제안되었지만, 이들은 유효한 프라이머를 디자인 하기 위해 BLAST와 같은 추가적인 툴을 사용하여 비 표적 서열에 대한 상동성 테스트를 포함하여 많은 노력과 주의를 필요로 한다. 또한, 이 방법들은 같은 엄격한 제약 조건을 만족하는 다수의 표적 서열에 대한 프라이머를 필요로 하는 정량적 중합효소 연쇄반응법 (qPCR)에 적합하지 않다. 이에 본 학위논문에서는 기존 방법들의 단점을 극복한 완전히 새로운 방법을 제안한다.

본 학위논문의 첫 번째 파트에서는 기존 연구들의 문제점들을 모두 해결하기 위해 맵리듀스 기반의 완전한 프라이머 디자인 방법인 MRPrimer를 제안한다. MRPrimer는 주어진 서열 데이터베이스에서 사용자에 의해 주어진 여러 제약조건들을 만족하면서 동시에 상동성 테스트를 통과한 모든 가능한 프라이머들을 찾는 것이다. MRPrimer는 범용 컴퓨터들의 클러스터와 그 클러스터 상에서 작동하는 7단계로 구성된 맵리듀스 알고리즘으로 구성되어 프라이머 디자인 시 특이성 검증을 동시에 수행하는 것은 기존의 방법들이 제공하지 못했던 특징이며, 상기 특이성 검증 조건은 다시 5 ‘ cross-hybridization filtering 조건과 general cross-hybridization filtering 조건으로 구성된다. 또한, 주어진 DNA 서열 데이터베이스 상에 존재하는 모든 적합한 프라이머

쌍들을 빠짐없이 구하며, coverage가 1인 primer들 뿐만 아니라 coverage가 1보다 큰 프라이머들도 모두 구하는 것을 특징으로 한다. 마지막으로, 사용자가 결과 프라이머들 중 생물학적 실험의 성공률이 높은 프라이머들을 쉽게 선택할 수 있도록 랭킹 기능을 지원하는 특징을 가진다. 343개의 프라이머 쌍에 대해 정량적 중합효소 연쇄반응법 분석과 시퀀싱 및 비교 분석을 통해 MRPrimer로 디자인한 프라이머들은 매우 안정적이고 효과적인 것으로 나타났다. 또한, MRPrimer는 효율적이고 확장성이 높아 RefSeq 데이터베이스와 같이 자주 업데이트되는 데이터베이스에 대해 유효한 프라이머를 디자인 하는데 매우 유용하다.

프라이머 디자인에 대한 기존의 웹 사이트들은 상동성 테스트를 위한 BLAST와 같은 추가의 툴 사용, 프라이머 랭킹 미지원, TaqMan probe 미지원, 그리고 다수의 표적에 대해 동시에 디자인 하지 못하는 등 여러 단점을 가지고 있다. 또한, 대규모 계산에 대한 오버헤드 때문에 몇 웹 사이트들은 휴리스틱한 방법을 사용하거나, 제한된 범위 내에서 상동성 테스트를 수행한다. MRPrime는 고품질의 프라이머를 디자인 할 수 있지만, 컴퓨터 클러스터에서 작동되고 제약조건을 조정할 때마다 수 시간의 런타임이 요구되기 때문에 일상적인 사용이 불편하다.

본 학위논문의 두 번째 파트에서는 구글 검색 시스템과 같이 맵리듀스의 클러스터나 긴 계산을 요구하지 않고 웹 인터페이스에서 사용자가 최고의 프라이머를 디자인 할 수 있도록 MRPrimer의 온라인 버전인 MRPrimerW를 제안한다. MRPrimerW는 완전한 상동성 테스트를 지원하고, 정량적 중합효소 연쇄반응법 실험을 위해 다수의 표적에 대해 프라이머 디자인을 제공하며 TaqMan probe 디자인을 지원하고, 결과 프라이머의 순위를 계산하여 가장 순위가 높은 top-1의 프라이머를 제공한다. 높은 정확성을 보장하기 위해 MRPrimer의 핵심 알고리즘을 적용하면서,

사용자가 웹 인터페이스를 통해 빠르게 질의에 대해 결과를 얻을 수 있다. MRPrimerW는 프라이머 디자인 서비스를 제공하며 사람과 쥐 전체 유전자의 99%를 커버하는 341,963,135개의 유효한 프라이머 세트를 갖고있다.

요약하여, 본 학위논문에서는 기존 방법의 단점을 극복한 프라이머 디자인을 위한 새로운 방법들을 제안하였다. 대규모 DNA 데이터베이스의 경우, 동시에 다수의 제약조건을 고려하고 특이성 검증을 통해 가능한 모든 유효한 프라이머 쌍들을 디자인 할 수 있는 MRPrimer를 제안하였다. 또한, 웹 인터페이스에서 주어진 사용자의 질의에 대해 완전한 상동성 테스트를 지원하고, 정량적 중합효소 연쇄반응법 실험을 위해 다수의 표적에 대해 프라이머 디자인을 제공하며 TaqMan probe 디자인을 지원하고 프라이머 랭킹을 지원하는 MRPrimerW를 제안하였다. 제안된 방법들은 중합효소 연쇄반응법을 포함하는 모든 실험에서 그 효율성과 특이성을 높이는데 유용하게 활용 될 수 있는 방법들이라 사료된다.

핵심어: 맵리듀스, 프라이머 디자인, 중합효소 연쇄반응법, 상동성 테스트

Appendix

Table S1. Primers for non-OR genes used in the biological experiments for MRPrimer.

Gene	Sequence		Size (bp)	Gene	Sequence		Size (bp)
Vmn1r26	F	TTTATTCCTCCGGTCTGTGCCA	100	Gcg	F	CACCAGCGACTACAGCAAATAC	169
	R	TAGTGGACCTGTATGGTGGAGAT			R	CTGGCCCTCCAAGTAAGAACT	
Vmn1r54	F	ACTACATCGTGCTCTCTGGCA	180	Rtp1	F	TACCCTCTTTCCCCACGTTCT	180
	R	GAGGGAAAAGCTGGTGATATGG			R	ACACATTGTGCTTGAGGTTGGG	
Vmn1r65	F	GAGGACAACAGAAGAAGTGGCT	142	Rtp4	F	TTCCTCCCCATCAAAGAGCTG	170
	R	TTGGATGATCTGAATGGGCCTC			R	GGGCAAATGCAGCAATAGACA	
Vmn1r66	F	ACATCCACAGCTCTCAGGTTTC	164	Rtp2	F	CGAGCAGTGTTACGATGAGGAT	174
	R	GACCATCACCAGACACCAACT			R	TTCTTGGAGGCATCGGTATAGG	
Vmn1r70	F	AGAGTTTGCAGGGGATTTTCC	142	Gnas	F	TACGATCAGGACGACTACGAGAC	175
	R	AGCACAGGACCAGAGAAGAAC			R	GAGTGAGTGACTGGTTGAAGGT	
Vmn1r71	F	CCAACACATCCGTAGCACTCA	109	Gnaq	F	GGTTGATGTGGAGAAGGTGTCT	183
	R	GGTGAGAGAACAAAAGGCCAGA			R	TGTGTAGGCAGATAGGAAGGGT	
Vmn1r72	F	TAACTCCAAAGGGCTGATGCT	178	Gnai2	F	CTTATGACTTGGTGCTGGCTGA	183
	R	TGAAAACCATGAGCAGTAGGC			R	ACTCAGGGAAACAGATGGTCAG	
Vmn1r73	F	ACTAAGAGTATCAGGTCCCAGGT	101	Gnao1	F	TCACCCTTGACCATCTGCTTTC	107
	R	ACAATGCAGCTCCCACATTTC			R	TTGTTGGGTGAGCGGTTTTTG	
Vmn1r78	F	TCTACTCTGCTTCTCTGATGGCT	160	Gna11	F	TCCGCACAATCATCACCTACC	136
	R	GATAGTTTTGGTGGCTTGGTCC			R	CTCTGTGGCCCATCAAACTCA	
Vmn1r80	F	GTTACGGCCTACTCCAAATACC	139	Gna12	F	CTAGAAAGGCCACCAAGGGAAT	152
	R	GAAAGCAGGGTAGAAACAGGTT			R	CGAGGACACCATGAACAGGATA	
Vmn1r84	F	GGTCTGTGTTTGAGCATCATGG	178	Gnai3	F	GCAGATGATGCCCGACAGTTA	136
	R	TGGAGTAGGAGAGGACAAAGGT			R	ATTCCCTGGACCTGCTAAAGC	
Vmn1r87	F	CTCATCAGAAGAAGCCCGTAGA	175	Gnat2	F	TGGACGTCATCAGGAAGTTGT	186
	R	GAAAGGCCCCTAGTAACACTGT			R	CGATGATGCCTGTTGTCTTGAC	
Vmn1r89	F	CTTCTCCTCACTCACGATCTCT	110	Gnaz	F	CAGAGAGCAAGGGTGAGATTACA	139
	R	CTACTGTGGAGATGCTGGGAT			R	AGGTCGTTCAGGTAGTAGGCT	
Vmn1r179	F	ACCAATCGACACTACAGAGGC	156	Gnal	F	TGGGACGATGAAGGAGTGAAG	124
	R	ACTCCAATGCCTCACAAATGC			R	GGTCTGTGGGTGTGTAGTCAA	
Vmn1r195	F	GGCATTGCAGGCTGTAAAACT	181	Gnai1	F	CGGAAGAGGAGTGTAAGCAGTA	152
	R	TACAAAGGAGGAGGAGAGAGGG			R	CCCAGCAAGCACGAAAAGTT	
Vmn1r211	F	GCTGTAAAGTTGCTGTCTACCTG	165	Cacna1c	F	GCAGCGTAAGGATGAGTGAAGA	170
	R	GGCAGAGGAGTGAGGAAAGAAT			R	TAGAGAGGCAGAGCGAAGGAA	

Vmn1r212	F	GGAGTTCTGACTGGATTTTGGC	135	Cacna1i	F	AAACGTGCTCCTGCTCTGTTT	159
	R	AACCTTCTTGGTGGGATCTGA			R	TCGTCCTCTTCTGGTTGGTAGT	
Vmn1r228	F	GGGACATGGCAGTAGGAATAGT	170	Cacnb1	F	AGATGACCGACAACAGGAACC	100
	R	GGAGAGAAGGATCAAGGCGTTG			R	CAGCCCTCCAGCTCATTCTTAT	
Vmn1r230	F	GCCAGGAATTTGGGAACAGGAA	124	Kcnk16	F	GTCATTCTCATCTTCCCACCCA	117
	R	CTGTGGGCTTTCGTTTGTGTT			R	AACAACATAGTCCCCGAAGCC	
Vmn1r231	F	TCGCATGAATCAAGAGCCACT	127	Kcnh2	F	ACCTGCTTACTGCCCTCTACT	133
	R	TAATCATCCACCAGCCAGCAC			R	GACTTTCCAGGACGGGCATAT	
Vmn1r232	F	GGAACATGGCAATAGGAGTAGGA	183	Kcnc3	F	CAGAAGACAAGAGCCCAATCAC	128
	R	GGGGGAAACCTTTGGAGATAATG			R	GCGGGACTTCTCGTAACCTTT	
Vmn1r233	F	CTGGTCTCTGGCAAATGTAGCT	139	Kcnh6	F	TACAGCAAATGCCCCCAAGTC	105
	R	GGCTAGAGGCTTTGGGGAAAT			R	GTCTGTTCATCTGGGCTTGGA	
Vmn1r234	F	GTGCATCAGCTCTTCCCTATACT	199	Kcnj11	F	TCGTGTCCAAGAAAGGCAACT	110
	R	GCACACAGCACCAGGGATAAT			R	AGTGTGTGGCCATTTGAGGTC	
Vmn1r236	F	CTGTACTATGTAAAGGAGTGCCC	139	Kcnk12	F	CTACTTCTGCTTCGTCACCTTCA	159
	R	CTAAGAAATGAGGTGCTGCCA			R	ATGGAGATGACGTTGAAGAGCG	
Vmn1r237	F	TCCTGGGCAACTCCTTCTTAGT	160	Kcnn1	F	TTAACCGCGTCACCTTCAACA	157
	R	TTCAACCCAAAGGCAGACACA			R	CTGGTCACTTCCTGCTTATCGT	
Vmn1r194	F	CTGTTCGTGATCTCGTCTTCCA	138	Kcnc4	F	GGAGGTAGAAACAGAGCCCATT	185
	R	CTCTGATCTCTGGGCTGAAAGT			R	ACAAACCACTCAATCCCACCTC	
Vmn1r235	F	GGCTTCTGCTCTGTTTGTCTTG	192	Kcnb1	F	GGAGAAAAATGGAGAGGGCGT	180
	R	GGACTCCGTGGATGATTGTGA			R	TTCAAGTGCTGCGGACTAGAC	
Vmn1r67	F	GGTGTGTACCTTCCTGGCATT	174	Kcnj9	F	TCGTCTCACCTCTCGTCATCA	178
	R	CTGAGTCTGGGCACAAAAGTAC			R	CACAACACTTCATCCACCAGGTA	
Vmn1r1	F	CTGCTCTCTCTGGGTTGTTAGT	170	Kcnd1	F	ACTGCAGCCCTGGTTTTCTAC	128
	R	GGGAAATGCTGGTGTTGTGAA			R	TCACCACACGACTGCTCTTTG	
Vmn1r196	F	CACAGTGGTCCAAGCAGTTATC	156	Kcnk1	F	TTGTCACCGTTTCCTGCTTCT	157
	R	GCTGTGTCTCTGATGGAAAGGA			R	AACTTCTGGTTGTAGCCTTCCC	
Vmn1r224	F	ACATTGGCTCCAGAACATCTCC	141	Scn1b	F	ACGTCTACCGTCTCCTCTTCTT	171
	R	CCAGCCACCAACCAGGATTATA			R	CCATCTCTGCCACAAGCCATAT	
Vmn1r77	F	ATTGGCCCCTTCTGCTTAGTCT	160	Clcn4-2	F	TGGAGTCTTTGGGGGTTTATGG	114
	R	AGAGTACAGCTCGCACATGATC			R	ACCGCAATAACCTCCAACACT	
Vmn1r88	F	CCTACGTTTGTCTCCTGGCTTA	145	Clcn3	F	GAGCATCTCGAGCAACTAAAGC	162
	R	CACGGCCAACGAGAGTCATAT			R	TTCTGTCTCCTCTCTGTCCTCA	
Vmn1r82	F	AACTCTGGCCAACTCCTTGTC	181	Clcn6	F	ATCCTTGGGGAGACACAGGAA	152
	R	GATTATGGCCGCTTGGAAACA			R	CACTTCACCGCCTCGTATCTT	
Taar1	F	GCGGCTGTTCTCCCTTCTTTA	188	Clcn7	F	CGACACAGCGTCTAATCACAAC	136
	R	GCTTTGTGGTGCTTGGCTTTT			R	GGACCTCTCCACAAACACCTT	

Taar2	F	GAGGCTTACGCTGATGGAATTG	147	Plcg1	F	CGACAGCACCAAGCAAAAGAC	116
	R	GCCGTAAATCCCCACCATCAT			R	CAAAGCGCAGAAAGGCAAACT	
Taar3	F	GCGAACACAAAAGGAGCAGTAG	191	Plcb4	F	AGTGAAGGCAAGGAAGGACAAG	154
	R	TACCCGAGCCATACCAGAAGAT			R	CGCTGCAGACACACAATATCC	
Taar4	F	GGCCCTCTCAGAAAGCAAAATG	104	Plcb3	F	TTAATCGGCGGCACATCACT	133
	R	AGGGTAGCCAACACAACACAA			R	AGCTTGGGTTCCTCTTCCTCTA	
Taar5	F	TGTCAAGCGGGAAAGAAAAGC	126	Pld3	F	TCCTTCTACTGGACCCTCACAA	124
	R	AGGGGTGGGGTGATGAAGTTA			R	CAGCGATGCGAACCTTTACAC	
Taar6	F	CAGAGTGGCGAGAAGAGAAAGAA	137	Sstr3	F	GCGAACAGCCTTCATCATCTAC	100
	R	AAATGTAGGCAGGGGTGATGAAG			R	CGACCGCACCTTTACCACAAT	
Taar9	F	CCTCCTTCTGTTTTGCGTCTCT	119	Insig1	F	GCTGTATTGCCGTGTTCGTT	111
	R	GCACAGTCCAGAAACCGATACA			R	TCCACCACAAACCCAAAGAGA	
Fpr1	F	GGTTCATCATTGGGTTCAGCAC	124	Anxa2	F	GAGACGGTGATTTTGGGCCTAT	127
	R	ACAAAGGAGAGAACCCGCAAA			R	GGTTGGTTCGTGAGCAGATGAT	
Fpr2	F	TCCCTGCCTTATAGTCTTGAGAG	108	Anxa4	F	AGAGAAGAGATGGGGGACAGA	168
	R	TGGGGCCTTTAACTCAATGTCT			R	GCCAACAGGGCATCTTCAAAG	
Fpr3	F	CATTCTCACTTTGCCCCTTTTCC	100	Pde6d	F	GTGGTTCTTCGAGTTTGGCTT	119
	R	AACAGAGTTGCCCCAGGATAC			R	TGATGACATTGCCCGTTAGGA	
Ifnar2	F	ACTGGCCCCTATGAGAGAAGAA	191	Pde3b	F	CCGTCGTTGCCTTGTATTTCC	164
	R	TCGTCTAGGAGGATGGTGTCTT			R	CTTGGGTCAATCAGCAGGTCT	
Reep2	F	ACCCTGTACCCAGCCTATTCTT	157	Pde10a	F	GGACAGAGACAAGCGAGATGAA	160
	R	GCTCAAAGTAGAAGGGGAACCA			R	GCGAATTACCTTCTCCCACTGA	
Reep3	F	GTGGTGCTGGTGTTTGGAATG	121	Adcy6	F	GACCAACTGCGTAAGGACCAT	178
	R	CAGTGTAGAGGGCAAAGACGAT			R	TCAGGGTGGAGTATGGGAACA	
Reep5	F	GTGGCTTTGTATCTGGTGTTCG	102	Adcy9	F	AATGAAACAAGGGGACGAGGAG	119
	R	GGGACTCTCGATGGCTTTCAT			R	TAAAGGGGCGGAATGCTATCG	
Reep1	F	TACAAGGCTGTGAAGTCCAAGG	157	Syt13	F	AGAACCTCCACTCCAACCAATC	110
	R	GCCAGGCTACAAACGCTATTT			R	GGCCCGTTTTGTCTGTTTCTT	
Syt7	F	GAAAGCCATCAACGACCTAGAC	120				
	R	TAAGGGGCGTAGGGTGAAATG					

Table S2. Primers for OR genes used in the biological experiments for MRPrimer.

Gene	Sequence		Size (bp)	Gene	Sequence		Size (bp)
Olfr631	F	GTGGCCATTTCAGGCAATTGT	190	Olfr1208	F	GGGTGATGTCCATTCTGACCTT	197
	R	GGATTTGGCAGGCTCCAAAAT			R	CTTGTCGTCCCCAACAGAATCA	
Olfr1133	F	CTTGTGGCTGTTGCCTATGCA	197	Olfr1230	F	TGTTCCCGTTGTTGCAACTAG	155
	R	CACCCTACCACACAGCCAATTA			R	TCCTTCAGAGCTGGAAGACTTT	
Olfr560	F	GTCATGGAATCCTCAGTGCTGTT	158	Olfr1234	F	GCAAGGCCTACATCCACATTTC	165
	R	GCGACCATCGGTGTCAACATTA			R	TAGCCCACTTACGATGGAGCTA	
Olfr855	F	GTCTCATCTTCAGCCTCTTCCT	164	Olfr1239	F	GTTGTGGTCAGCCCAAGTTTAG	169
	R	GATGACTGTGGCTGTGCTTAAA			R	GTCCTATGAAAAGCTGGCTCATG	
Olfr1010	F	GCTACCAGGCTGTGCTCTATTT	107	Olfr1240	F	AGTCATCGTTGTTTGCTACCTC	101
	R	GCGGATCTTCAGAATGGCTACA			R	AAGTGTCAGTACATGCTAAGCCC	
Olfr1055	F	CCAGCTGACCTTCTTCAGCATAT	148	Olfr48	F	TGTGGGCTTGACTCAGAACAT	112
	R	AGGTATTGCCACCAGAACATGAC			R	GCTGCTGATGGTGACCATGATA	
Olfr16	F	TTAGGCACTGGCCTGGTTATG	196	Olfr1258	F	GTTGGTCTCCTATGTGGTCATTC	156
	R	ATATCAGCCCTATGGGCACAAG			R	GACAAAGTGGCTGAAGGTCTCA	
Olfr1406	F	GCCATATTCAGGCTGCCATTTT	141	Olfr1260	F	CTGTGACCTTTTCCCGCTCTTA	173
	R	GGCTGTGATGACCATAAGGCTA			R	CCTTCAGCTGTTCTTCAGTGCTA	
Olfr218	F	CCTAACGCTGTACCTTCTGACT	160	Olfr1262	F	GAGCACCATATCCTTCAATGGC	185
	R	GCCAGCTAGCATTCGTGGAATA			R	CGATTCCAGCTGCTACCACTAA	
Olfr1404	F	GGCGTGGTCTTCATCTCCTAT	170	Olfr140	F	TAGTCCACGCAATGTCACAGA	148
	R	AGCAAGCTCTGGGACTTAGGT			R	GGGGCTGAAGGTAATCGTGATA	
Olfr432	F	CTGGCATGCTCGGCTTTATA	100	Olfr1278	F	CCAGGGTTGTGTCTTCCAGATAT	167
	R	CAGCACAGCTCTCACAATACCT			R	CCCAAGCACCAGATAGAAGCATA	
Olfr429	F	CTGGGCCAGTGGTGGAAATTT	199	Olfr1279	F	GTGCACTGATGGAGACAAGTT	152
	R	TGCAAGTAGGAGCTGAGGATCA			R	TGACAAGCGCCTTTGACAAG	
Olfr417	F	ATAGACAAGGACAGCCGCATTT	172	Olfr1284	F	AAGCATCATCGTGGGAAACCT	125
	R	GTCCACATGCAACATTGGTCAT			R	CTGTGGTAGAGGAAAGTCCAACA	
Olfr248	F	CCATCTGCAATGCCCTCAAATA	187	Olfr1289	F	TGGCTACTGCATGGGCAATT	170
	R	AGCCTCAGAACTGCCCTCATA			R	GGCAATGACCATGATGTCTAACC	
Olfr345	F	CACCCTCTGCACTATTCACAAA	174	Olfr1301	F	TGACCATTGTGTTGGTGCAGTA	191
	R	CAACCAGTCAGAGAGGTCACA			R	GCAGTTGCACAACTGCCAAAA	
Olfr50	F	GTTCTGCTAGTGATGGTGTCCT	200	Olfr1305	F	CCACGGATGTGCCTGCTAATAT	158
	R	GTGATGACCACCACTGCTAAAG			R	GTGCATGCAAGCTTGACAAGT	
Olfr350	F	CTCTTATGGTCACATTGTGGCC	179	Olfr1311	F	ACTTGGCCCTTCCCTTCATCA	138
	R	CAGAAGCAACTATGCCCTTGTC			R	CTGGGAGCAGAGTTTCCTCATT	
Olfr354	F	TGACCAACTGTCCTGCCCTTAT	174	Olfr71	F	AGCGGCTACTCTTTCCTCTGT	177

	R	CAGCTAGAAACACCAAGCCCAT			R	CCAGCATCAGGGGTACAAAAGT	
Olfr356	F	CTCAGCAGTCTGCCTGTTCTA	180	Olfr275	F	TGGAAACTGGACCTGTGATTCA	165
	R	AGGAACCTTCTCAAGGCACTT			R	GCAACATTGGCAATGGGAGAA	
Olfr362	F	GAGATGCTGGGGGCAATGTTAA	199	Olfr273	F	AGGTCTTAGCTGTCCTCAAGCT	161
	R	GCAGCTGAAGGCAATCGTAAGA			R	CCCTGTGGCTGAGTTCATTCT	
Olfr368	F	CTTCACCCGGTCCAGAGTTAT	140	Olfr1340	F	GACCTGGATTTTTGCAGCTATGG	101
	R	GTAGCACCATGCCCACATTTC			R	CCAGGGCTATGCATAGGGTTT	
Olfr988	F	CATGGGCTTCCTAAATGCTTCTG	185	Olfr38	F	CACAAGGCTGGTCATCACATC	127
	R	GTACAGTGCTCACTAGGTTGAAC			R	AGCAAGGGTTTCACAGGCTAT	
Olfr992	F	AGCCACTTTAGTGGGCAACATT	181	Olfr452	F	CACAAGGCTGGTCATCACATC	140
	R	TCCTGTGAAGGAGATGGATGGT			R	CCAATCGGACCACAGCTAGAAT	
Olfr1009	F	ACCATGACGGGAAACTTAGGT	120	Olfr450	F	CTTAGCCTTGGGTGGTTCTGA	192
	R	CACGACTGACGAGAAGCAAAT			R	AGAGGAAGACGGAAGGTGATTAC	
Olfr1014	F	CTGGATCTTGGATTGTCCACAGT	128	Olfr307	F	ACCCTTGACCTACAGCTCCAA	147
	R	TCAGTATACCCAAGTCCAGCAG			R	GGCACAAGCCAGAACGGAAATA	
Olfr17	F	CAGCACTGCCATCCTCACATA	176	Olfr305	F	TTCATGGCTCTCCTTGGATCAG	153
	R	GGCCCAGAGTTCTGTGAATGA			R	TAATCCAGTGCCCCAGGAAAC	
Olfr1022	F	CAACACTGCAGTGATGGATTTC	117	Olfr552	F	GTGGGGACACACGCTTCAATA	114
	R	CACAGGTTTCCTGTCAGTGTT			R	GGAGAACAGCACGAAGGATGAA	
Olfr1030	F	CATATGCTGCGTGTTTAGTCCAG	193	Olfr553	F	CACCTCTGATGCCAGGTTTAAA	131
	R	AGGCCACTAAGGAAGCCATAGA			R	GGATATGCGAAGGCACATTGT	
Olfr1043	F	GCCAAATGCGCTGGTGAATTT	180	Olfr556	F	CCGTCATGTGATGCTGGGAATTA	166
	R	CGCGGCATGAGAATGACATAGA			R	GGGGTCAGCACATGCTAACTT	
Olfr1052	F	GCTCAGCTTCTGGACGACAAAA	113	Olfr575	F	ATCCCATAAAGAGCGCCTCAAG	104
	R	ATACTGGGAGCTTGGCTGAATG			R	AGCGGTGCATGGATGCTAAA	
Olfr1079	F	CCTTGGCTCCTATCTGCTCATCT	118	Olfr577	F	CCCATTGCCTTTCATGCTCAAA	160
	R	CCCATAAAAGACAGAAACCACGG			R	GTCCACTCCCACTGTAGAAACA	
Olfr1089	F	TAGGCATCACTAATCGGCCTG	122	Olfr592	F	CAACGCTGTGTATGGCCTTTT	134
	R	CAGTCGAGGGTCCACTATTGT			R	TGCTGAGGGCTTTTAACCGT	
Olfr1090	F	ACAACCTCAGTCCAGTCATTCTG	138	Olfr599	F	GTCGTTCTGGTGGTTGCAATT	180
	R	CTCCAAGTCCTATGTAGGGCAAA			R	AGCCACAAAGAGCCCATATGAT	
Olfr1093	F	TGGAGTTGTGCATGGTGCTAT	167	Olfr606	F	CAGCCTTACTTCGGAGTGCTAT	191
	R	CAATCAAGCCCACCAAGTACAA			R	ATCCATTCCCCACAGCATAAGA	
Olfr1106	F	GTGTTGGGATCGTGCTTAGGA	171	Olfr622	F	AGGTCTAGTTGGCCTGATGAGA	179
	R	CACCAAAAGCAGCAGTTCATTC			R	GGGCCACTGAAATTCCATATGC	
Olfr1111	F	TGCAGATGCTGAGTGCCTTAT	146	Olfr65	F	TGTCATGGGCATTGCTTCTACAG	148
	R	AGGTCACGCTTCCGCTAAAA			R	GTGGACGACATGAGGAACATTCT	
Olfr1112	F	CCTAGCTTGTGGGGACACTTTTA	142	Olfr648	F	CATAGCTGTGGTGGGAAACTGTA	186

	R	CCCAGTTGCAGATGGAAGCTT			R	AGGCAGCCTGGAAATGTGATT	
Olfr1124	F	GTGCTGCATTCTTATCCTGGGA	153	Olfr661	F	AAATTCCACTCATCCCGGTTCTG	126
	R	CCACTTATCCAGGAGCCAATCA			R	CCAGCCTGAGCACCTTGTATAA	
Olfr1128	F	GGGGTCTTTGGAGATACAGAATG	138	Olfr6	F	CGAGTTGCCTTCTGTGGCAATA	151
	R	TGCATAGGCAACAGCCACAAG			R	CAGTGGCTGAGAGTGGGAATAT	
Olfr1136	F	GGCTGTTTTCTCCAACTCCTGA	133	Olfr885	F	GTGCCTTGTCCCACATAGTTT	167
	R	CTCTACTAGACATGTCCACAGCA			R	GGGCACTGTTATATTGACACCA	
Olfr1152	F	GACCATCAGCTTCACATCCCAA	113	Olfr933	F	GACTTTGCTCACTGCTAGTGTTG	167
	R	CTCTTGGCAAGCAGGTCCAATA			R	CAGCTTTATGTCAGAGCAGGAG	
Olfr1157	F	CCCTGGCATGTATGGTCCAATT	168	Olfr768	F	AATGGCGTCTGGGGACAATAC	181
	R	GGTAGCATCCAGACACCAGTAT			R	GAACTGGATGCAGACCCTGTTG	
Olfr74	F	TGCTAGCCCTTTCTTCCTCTGA	147	Olfr788	F	AACTGTGCTGCCCAACTCTTT	141
	R	CACTGGCTGAACGCATCTTAAG			R	GGTGCAGAGTTTCCTGTTCATG	
Olfr1161	F	CTCAAGATGCAATCATCCAGGGG	144	Olfr812	F	ACCAGTCAGGAGAGTTAGAGTTC	120
	R	CACTTTGAATGTGAGCTGGGAG			R	TCGTTAAGTTCCCCATCATGC	
Olfr1162	F	AGTCGTTAAAGTGGCCTCTGT	131	Olfr827	F	AAGGAGTGTCTGTGTCCAGTTG	113
	R	ACTGAAGAGGGATCTTTGTGCTC			R	CAATGACCCTAGAGGCACAAAA	
Olfr1164	F	TGAGGCTTGTTGCCTCTTGAT	146	Olfr1389	F	GGTTTCTGGCCTTGTGAACTC	148
	R	CGATCCCATGACAGACACCAATA			R	CATCTTGACCTCCGTTCCATTAG	
Olfr1176	F	CCCCACTAAAGGAGCACTACAA	109	Olfr51	F	ACCTGCATGCTCTCTTGCATA	160
	R	TGACTTGAGCACGCAGTATAGA			R	ACCAATCCACCAATGGTGAAG	
Olfr1179	F	CTTGCTGTGCTGGTGCTTTTA	185	Olfr11	F	CGCTGTGGCCACTATGTCATA	199
	R	CAGTCCCATCATGCCTGAATTG			R	GACCTTCGGCAGATTGGATTTTC	
Olfr1180	F	CTTCATTTCCTTGGTGCCATTG	166	Olfr745	F	CTTCTGGTGCTGGTCGAACTA	131
	R	GACTGGCTGAGTGCACAAATC			R	CTTCTGCATTCCTGCAGGATTC	
Olfr1184	F	GCCATGCACTTCTTTGGAATGA	129	Olfr283	F	TCGGAAACTTCCTCCTGATACT	150
	R	GATCAGGATATGGCACCTGCTT			R	GGAGTAACCCTGAGCCGTAAAA	

Table S3. Primers used in the comparative biological analysis.

		MRPrimer			PrimerBank		
No.	Gene	Sequence		Size (bp)	Sequence		Size (bp)
1	Olfr613	F	TGGTTAGAGCGGAGCAGAATC	247	F	CCTTCTGGTTAGAGCGGAGC	110
		R	AGTGGAGCACAGATAGCAACC		R	CCAAGACACTAGGCATTGTTGAC	
2	Olfr911-ps1	F	TGGGGCTGGAAAATGGTTCTT	124	F	TTGTGCCCAGTGTTATCATCTTT	120
		R	TTCCCCACTGCTGTTGTTGT		R	CAGCAAGTATATGGGAGCTACAG	
3	Olfr130	F	GTCGCACATGCTGGTAGTAGT	223	F	TGACACTGGTAGGCAACACAG	176
		R	AAGAATTTTCTTCCCAGTGCTGT		R	TGTGGCAGTAATTGTCTTGGC	
4	Olfr235	F	GGCTGTAACCTGGAACTTTTCC	127	F	GTGTTTTGGAATAGCAAATGCCT	180
		R	GGGGGCTGTGGAAGTGATATAG		R	GAGCTGGATCGCAAATAGACAA	
5	Olfr1053	F	GTGAATGTGCTACCCAGTTGTC	172	F	CATCACTGCCTGGGTTCATCT	108
		R	CTGTAGAGGTACGGGATGCC		R	GGTTGCCCATGACTGTGACT	
6	Olfr611	F	GAGGAGGCTCTACTTTTGTCGT	134	F	TTCCCCACACTGTTGAGAATCT	141
		R	CCAAGGTAGAGAGCACCACAA		R	CACATACCAATCGAAGGCCAT	
7	Olfr1303	F	CTGCTCTGTTACTTCCCCCAA	120	F	GTGTCTGCGTTTGTGTTTCTG	106
		R	ACCATCTCCACTCCACCAACT		R	GGATGTTTCCAGCCATGCTTAAT	
8	Olfr401	F	CCCACAGGCTTACAGTTCCAT	187	F	TATTCCCTCTATGTTGGGTCGG	150
		R	CAGGATGAGGCCACCAAAATTC		R	CAGATGGCTAGGAAGCGGT	
9	Olfr118	F	TGCTCCCCACTCCATTACTCA	233	F	CACTTGCTTGTGGCGATACAT	153
		R	ACTACGGCCACAAAGATTGCA		R	CTTTATGGCGACCCTCAGGTG	
10	Olfr340	F	CATCAGTCGCATCTCCAAAAATG	250	F	TTGGGACTCCCCATTCGAG	195
		R	AGGGAGTTGGCAGTAGATAAAGT		R	TGGAGATGCGACTGATGAGAA	
11	Olfr1295	F	CTCCAGGGACTTTCCCACTCA	198	F	GGGGTTGTGGTTGTAACTTGC	104
		R	AGGAGTGATGTTTGAGGAAAGAC		R	GACAAAGCCTTAGATGCTCCAG	
12	Olfr453	F	TCTGGGCTTAGGAGGGATTGA	145	F	CCTGGTGGATGTGTCTTATGC	135
		R	ACCACCGACCCAAGAAACAAT		R	AACTCAATCCCTCCTAAGCCC	
13	Olfr510	F	TGGCTTCCATTGACATAGCCA	100	F	ACCACACTGTAGTCACAGAGT	123
		R	ATGCCACACCCAATGTAGGAT		R	TGGTGCTTAGATTCCCAGACA	
14	Olfr539	F	TCTGTACCTCTTCTGTGATTCCT	245	F	GTGGTCCCAAGGTTATCACCC	156
		R	AAGACCGATGAACCATACAACCA		R	TGCAGCCATAAGACAACAAGG	
15	Olfr1388	F	GGCCTTCTTTTTGGTGGGATTC	127	F	TCTCTCTCGACTGGACCTTCG	153
		R	GTCCAGTCGAGAGAGAGCAAT		R	ACACACCTTTCATAGCTGATGG	
16	Olfr1443	F	GACTGGAAACTTGGGGATGCT	121	F	ATGGAGAACAGGACAGAGGTG	143
		R	TGGGGTAACAGCAGAGGAGTA		R	AGAATCAGCACAAGCATCCCC	
17	Olfr1494	F	AGTCCCATTCCTGCTGATTTGT	167	F	CCCTCTACACTACAGCCTCAT	147
		R	AGCGGGGCCTCAGATATACA		R	TGGTTGATTTCCTGGTCATGTC	

18	Olfr1204	F	CCGCAGTTGCAGAAAATCTTG	201	F	TCACACAGAATCCGCAGTTG	113
		R	ATGGAAAGTATCAGCAAGCAGTT		R	CAGTTGGCTGTTTGTAATGGTG	
19	Olfr635	F	AAACACCACCATCCTAACCGTTA	188	F	CCATCCTAACCGTTATCCGCA	117
		R	GGAAGAAGAACTGGGCAAAACA		R	GCATGACTGTAGGGAGTGTGG	
20	Olfr694	F	CAGCCCTGTACTTTTTAGCCATA	122	F	GAGCTGCTCTGTGCCACTATC	173
		R	AAGCAAGTCCATGAGAGAGAGC		R	ATGGCCTTTGGAGTGATGACT	
21	Olfr1511	F	CGCCAGCAAGGTTATCGCATT	165	F	TTCTGGGTGTGCTCTCCTTC	101
		R	GTGCACAACTTCCCATTCATGA		R	ACCAAATGCGATAACCTTGCTG	
22	Olfr960	F	AGCAACCTCTCTATCTCTGACAT	148	F	AGCCTGTGAAGATTCCTCTCT	230
		R	AGCACAAAACGCATCCAATACA		R	GGTTGGGCTGTAGGTAGATGAC	
23	Olfr1411	F	CTCTCATATGATGGCTGTGTCCT	102	F	CTGCCATTGCCCCTAATGC	210
		R	ACAGATATCACCTTGCCCTGT		R	CCAGACGGGTACAGGTTGTAG	
24	Olfr619	F	GCTCGCATACTTTGTGCTGTG	215	F	GGCTCACCTATTGTGGGAAGA	74
		R	GGATTGAGGGAGGGTGGTAGA		R	ACAGGCCAACCTGGCAATG	
25	Olfr122	F	GCTGGTAATGCCCTCACGT	109	F	AGGAGAACAGCTTGTCTGTCA	60
		R	CCTCAAATAGGTGGCAGACGT		R	CTCCAGGGACCTCAGAGAACT	
26	Olfr1033	F	ACCTTACAGCAGTTGGCATATTT	195	F	TCCACCCCTCATCAAGATGG	258
		R	ACTTCTTGGCAATCACTTTGTCC		R	GACTCCTCAGTGGGTCGTCT	
27	Olfr1331	F	CCTTTGTTACCACCACCATGC	114	F	TTTGGTCGTTACTCCAATCTCCC	240
		R	TTATACCCAGGCCACCGAACAT		R	AGCAATGGTGTAAAAGCACTGT	
28	Olfr740	F	TCCTGTTCGTTCCTTTCCTCTTC	178	F	CCCGACCTCTGAGCATGAAG	83
		R	TCAGAGGTCGGGCTTAGATACA		R	TCACAGGATTAACGAGTGGAGT	
29	Olfr190	F	CTGGCTTTTGTGGATGCTTCC	162	F	CTGTGACATCGTTCCATTGCT	263
		R	TGCCATAGCTCCCAACAAGAA		R	CTTGGGACACGGGGAAAATATAC	
30	Olfr1465	F	CATGAGTGGGCTCCTAAAAGGA	189	F	ACATTGGGGACACCTTCAATC	254
		R	AGACACACACACACACCTGAA		R	TGCGGCACAAGTGGATACAG	
31	Olfr1255	F	CTCTTGGTCTCTTTGTTGCTGC	229	F	AAAAGGAACGTGACTGAGTTCAT	244
		R	GCTTTATCAATTGGCAGAGTGGT		R	GAAGGGAGTCCGCAATCAGC	
32	Olfr1392	F	CTAACTCTCTTTGGGAACACTGC	203	F	CTGGCCTGCACTAGAACTCAT	78
		R	GTTATGAAGAGCTGAGACACACA		R	ATGGCAGTGTTCCCAAAGAGA	
33	Olfr799	F	TCCATTGCCTATGCTGCTTGTA	214	F	ACAGATGACATTAGGCTGCAAA	227
		R	GGCTAATTGGTGGGAGAACGA		R	GCAATGGACTTATCCCCAGATG	
34	Olfr569	F	GCCTTGTTGGCTATCACTGAC	109	F	GGAATCCCAGGGTTGGAGAAT	87
		R	GGATGAGGCAGGCGTTGTATT		R	GGTGATATTTCCAGTCAGTGCC	
35	Olfr1297	F	CATTGCTTCAGGAGAGGTGGTAT	178	F	GGTATTGTTGGCTTTAATGGCCT	372
		R	CTGCAAAAGCACTACCACGTG		R	GCCTTGGAAGCTCCAGTTTTC	
36	Olfr967	F	GATGTCCTATCAAGTCTGCACC	175	F	GCTTTCTTGCTCCCCTACTTT	348
		R	AAAAGTAGGGGAGCAAGAAAGC		R	GTTAGTGCGACCTTGACATCC	

37	Olfr355	F	TCCTTTCCCCACTTCCATTCAC	198	F	GATGTGCCCTACTGGTGACC	336
		R	AGCAATGCATAGGAAGGGAGTC		R	AAAAGAGGGTTACCACAGTGAAG	
38	Olfr1225	F	CCACAGCTCTGAAGGGAAATTT	171	F	TGTGGCAATCTTGTGATGGTG	265
		R	GAATTGAGCAATGGGGTCAACAC		R	TGTGGGCTTTCAGAGAGTACA	
39	Olfr1079	F	TCCTGATCATCCTTGGCTCCTAT	128	F	ATGTCACAAACTGTATGTTGGGT	253
		R	CCCATAAAAGACAGAAACCACGG		R	GCAGATAGGAGCCAAGGATGA	
40	Olfr918	F	CCACAGGTCTGCTCAATGCTA	154	F	TGCTACTCTTCTGTATCCAGTCC	251
		R	AAGAAAGTTGGAGGAGGGGCA		R	CCAGCAAATGCCATCCCATAAG	
41	Olfr1336	F	CCTCTTCTTAACTTGTCCTGCAC	142	F	GAGTTGGGCAATGTGACCAGA	249
		R	ATGGCATGCCTAGAACTGTCCT		R	CCCCATGAGTAGTGTGGGC	
42	Olfr1408	F	ACAGTAAGGACCAAGACCAGCT	125	F	AAGATTGCCTCATCTGATGGC	243
		R	CCTACACAACACTTTCCGCAGA		R	CCTACACAACACTTTCCGCAG	
43	Olfr830	F	TGTTGGCCATCAAGTGTGACTT	182	F	CCTTATGAACCCCAGTTTCTGTG	75
		R	AACTACAACAAAGCAGGCCTGG		R	TGCAGCAGACCATTTACAATACT	
44	Olfr1356	F	CAGTCCTTCTTCTTTGGGTTGC	145	F	AACCTGTCCATAGCTGACATCG	76
		R	CGATGTCAGCTATGGACAGGT		R	TGCTTTGTGTGCGGATATTCT	
45	Olfr1353	F	TCTGGTTCTGGTGTCTTGGATTG	101	F	AACTTCTTGCTCACTATCATGGC	584
		R	AGTGTGGGATTTCTGGCTGTG		R	ACGAAGAGTCTTTTTAGGGCAC	
46	Olfr828	F	GCTCTTCATTCACTACCTCAGTC	144	F	GGACTGATGGTGTTGCGGTT	444
		R	TTGCCCTATTGATGTGCTTCC		R	CCTATTGATGTGCTTCCTCAAGG	
47	Olfr1145	F	TGACTCAGACTCCGACTAAATCC	120	F	CTGTCTGCACTGTTCTCCATC	391
		R	GGAGATTGGGAAGAGCAGAGAA		R	TCTGAATTGGAATGCCACTTAGC	
48	Olfr995	F	GTTGTCCCAATCATCAGCCTTTC	100	F	TATGCCACATTTGCGACCAGT	326
		R	TGACCAACACAGTGAACGTCA		R	AACACAGTGAACGTCAGGTTAAA	
49	Olfr978	F	TGGCCCTGGTCTCTTCATCTA	174	F	GGCCCTGGTCTCTTCATCTAC	63
		R	TGAGGAAGCACAGACCCATAT		R	GGCCACAATTCCATCTACAGC	
50	Olfr693	F	CGCCTCAGTTATCAGTCCCAAA	107	F	CCATGTACCTCTTGCTTGAGC	67
		R	TACCCAGTGCCAGTTCCAAGAA		R	TTGGGACTGATAACTGAGGCG	
51	Olfr967	F	GATGTCCTATCAAGTCTGCACC	175	F	GCTGGATTAACAAGCACACCA	72
		R	AAAAGTAGGGGAGCAAGAAAGC		R	CGTTACTGCATAGATTCCGAGG	
52	Olfr653	F	CTTCCACCCTCCCACATTTGT	112	F	CCACCCTCCCACATTTGTTTT	78
		R	ACCATTTCCAACCAGAGCAAGG		R	GAGCAGAAGGGAATAGCAATCC	
53	Olfr1284	F	GTAAGCATCATCGTGGGAAACC	133	F	GGACTCTCCAGTTCTTGGAAAAA	78
		R	TGGGCACTGTGGTAGAGGAAAG		R	TCCCACGATGATGCTTACATAGA	
54	Olfr348	F	GTAACACTGTCCACCACTTCTTC	161	F	ACCCCCTCCACTATACAAGAATC	89
		R	AGTGGCTCCAATGCGTCCATA		R	AAGGGCACCAGCAAATGATAA	
55	Olfr222	F	GGTCATCCTGACAGTGCAATT	114	F	CACCCCAAAGGACTTTATCCTC	103
		R	GCATCATTCGCCGTAGTATCAG		R	CCCCAGCATAGCCAGAATGT	

56	Olfr267	F	CTGACTGCTCTGCTGGAAACTA	168	F	ACCCGAGGTTAGAGATTGTTCT	111
		R	TGGAATGGGCAGAAGGAGAATAC		R	TTTGAAGGCGTGAATCCAGGA	
57	Olfr433	F	GGGCACTGACAAACTTATTGCCT	119	F	CATGTCTCACTGCCAGTTTACC	105
		R	TCTTTTCACTGACCCTCTTCACG		R	CATGCCACCTTGTGCTCAC	
58	Olfr461	F	CCTTTTTGCGACCTTCCTTCT	181	F	GGCACCAAAGAACTACACCAC	109
		R	AAGAAGCCCCTGGAGCATATA		R	GACGTTTGTCAGCATGGACTA	
59	Olfr1344	F	GCTTTGCACTCTCTGTACCCA	200	F	GAACCCTCAATGATTCAGGAACC	129
		R	AGAGGCTAGGATGAGGACCAAA		R	GGGCACCATTACCCAGCAC	
60	Olfr1358	F	GCCTTGATTGAGACCTGCATGA	144	F	CCATCCCTTGCGCTACTCTG	112
		R	GCTCAGGAAGAAGATGCCAAGT		R	GAAGATCATGCAGGTCTCAATCA	
61	Olfr1420	F	ATTAACCACTCACTCCACACCC	185	F	GGGGTGCAGATGGTGATTTTT	177
		R	TCAGCTCCACCCAAGAAAACA		R	CAGTGGTGCAATGGATGATGTAT	
62	Olfr1509	F	CAACTGGGTGTTGGAGATTCTG	109	F	CAACTGGGTGTTGGAGATTCTG	108
		R	GAGACGTGGCGTGAAGACTAT		R	AGACGTGGCGTGAAGACTATG	
63	Olfr5	F	GCTAGGTGGGCTATTGGTTTCT	186	F	TTGTGGGCACAGAGTGCATT	133
		R	AAGCAACCCCAAAGGATGACA		R	CCTAGCCATGAAATCATAGCCAA	
64	Olfr39	F	CATGCAGTGCCTCACTCAAGT	166	F	CAGATGATCCTAAATTGCAGCCT	113
		R	CCAGCACATTAGCACAAGGAAG		R	GAGATGGGAATCAGAACTGACAG	
65	Olfr49	F	ATTCCTGGGCTTTCTCCTGAC	185	F	CTTCCCCAAGATGCTAACCAAC	172
		R	GTCCTGTGATGATGTTGGTTAGC		R	GGTGGCATAACGCAAAGGG	
66	Olfr632	F	CTTCTGTGCTGGGGGTGTTAT	128	F	ATGAAGGTGTCTATTCCACCACG	100
		R	GTCCAGAGCCATAGCAAATAGC		R	GCAGTGAAATCCAATGATGAGCC	
67	Olfr1444	F	TTTCTTCTTCGTGGGGTTTGC	154	F	GGTTGACAGATGACCCCAATC	117
		R	GCCACAGGTGTAAGAGCCAAT		R	GGTGGGAATCCGAGAAGATGA	
68	Olfr424	F	GGTTCCTGCATCTTTGGCTTTC	124	F	GGATGGATACCCGTCTTCACA	127
		R	CCAAACGCAACACAGGTTCAA		R	ATGGTCCTTTGCTTACTGATGAG	
69	Olfr460	F	GTGGGAAACACGGTCATCATTG	127	F	TCCTGGCTCTGTAAACCTACG	188
		R	GCATCACGGGCACAATAACAG		R	CGGGCACAATAACAGTTGTAACC	
70	Olfr677	F	AGCTTGTGCACCCATCAAGAT	156	F	CCTTGTGGGAAACATCACCAT	175
		R	GTGTTTAGCGCCTTCAATCGG		R	GCCCCCGAAACTGATCTCC	
71	Olfr218	F	GAGCATGGCTATTGTCCAGGTTA	168	F	CTGCAATCCTCTAAGGTATTCGG	102
		R	AACACAAGGACACACACGCT		R	ACCTGGACAATAGCCATGCTC	
72	Olfr362	F	TGCCCCACACTTTCCTCTTTT	136	F	TGTGCCCCGAATGCTTCAC	122
		R	GAGAGGCCACGCAAGAGATAAT		R	TTCCCCAATATGGTGGTCAGATA	
73	Olfr558	F	TCCCTTTGTGTTCCCTCTACCT	153	F	TCAATAGCAATGAATCCAGTGCC	114
		R	TTGGCATGGATGAGGTGGAAA		R	GCACAGCAATAAGGTAGAGGGAA	
74	Olfr78	F	CATGCCACCTTCCTGCTTATTG	138	F	ATGAGTTCCTGCAACTTCACC	111
		R	GCTCCGCTCTGTTCTCACTAT		R	TGCTACAGCATACATGGAAAGC	