國立臺灣大學電機資訊學院資訊工程學研究所

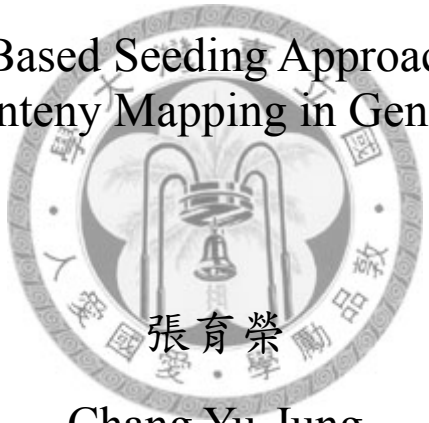
博士論文

Graduate Institute of Computer Science and Information Engineering
College of Electrical Engineering & Computer Science
National Taiwan University

doctoral dissertation

詞頻探索方法用於高效率之基因體同源與同線圖譜對映

Copy Number-Based Seeding Approaches to Efficient Orthology and Synteny Mapping in Genome Comparisons



張育榮

Chang Yu-Jung

指導教授：高成炎、何建明、黃明經 博士

Advisor: Kao Cheng-Yen, Ho Jan-Ming, Hwang Ming-Jing, Ph.D.

中華民國 97 年 7 月

July, 2008

誌謝

首先感謝 高成炎教授、何建明教授與黃明經教授對本篇論文的悉心指導，以及口試委員李德財教授，歐陽明教授，趙坤茂教授，呂學一教授，曾宇鳳教授，與施純傑教授所給予的建議與指正，使本論文得以順利完成。於論文研究期間，承蒙中研院計畫與國科會計畫的資助，特此誌謝。此外，要感謝台大與中研院的師長與研究伙伴們，彼此間的交流與砥礪，還有家人的支持與鼓勵。

最後，謹將本論文呈獻給辛苦培育我的母親陳月霞女士，以及所有關心、愛護我的師長、朋友與家人。

中文摘要

尋找與回溯不同生物基因體間在演化上之共同來源區段(稱之為演化同源與同線圖譜對映，synteny and orthology mapping)，是比較基因體學中基礎的工作。隨著定序技術的進展，愈來愈多的大型基因體序列已經定序完成或近乎完成。這一方面使得以全基因體比對進行演化同源與同線圖譜對映顯得日益重要，另一方面也帶來了新的研究挑戰。面對為數眾多、隨時間分歧演化且動輒數十億萬鹼基對的基因體序列比對，我們要如何建立具備高靈敏度、高特異度以及高效率的比對引擎與方法是其中核心的研究課題。

我們首先針對近距大型基因體間同源與同線圖譜對映，發展出 UniMarker 方法。以人與小鼠比對為例，此方法採用長度 16 且在這兩個基因體都只出現一次的短序列來建立出次數頻譜，以偵測尋找同源與同線的基因體區段。實驗結果顯示，人與小鼠（基因體長度均為約三十億萬鹼基對）的基因體同源與同線對映只需數小時於一台個人電腦即能完成，同時其產出之圖譜與小鼠基因體定序協會(MGSC)之圖譜有 99%的一致。

接著，針對非近距大型基因體間同源與同線圖譜對映，我們提出新型態的種子詞彙(seed)，稱為 maximal α-marker pairs(簡稱α-pairs)，α代表該種子詞彙在兩個欲比對序列上之總出現次數的上限，這種選取方式有別於常見以限制種子詞彙長度而不考慮詞頻的選取方式,例如:採用固定長度的 *k*-mer 與設定長度下限的 MEM 方法。奠基於增強式後綴陣列(enhanced suffix arrays)，我們提出了一個線性演算法來產生所有的α-pairs。根據人比對小鼠、雞與河豚的實驗結果，上述α-marker 方法較之限制長度的方法(*k*-mer, MEM)在連續性匹配(contiguous matching)的同源種子詞彙選取(orthology seeding)上，能同時達成明顯較佳的靈敏度與較佳的效率。此外，我們更延伸此詞頻探索方法到非連續性匹配(discontiguous matching)的同源種子詞彙選取。從 ROC 曲線上的比較結果顯示，非連續性的 wobble α-pairs 明顯優於其他未限制詞頻之非連續性種子詞彙(spaced *k*-mer seeds)。

關鍵詞：比較基因體學，演化同線對映，演化同源對映，序列比對，後綴陣列。

Abstract

Motivation: Orthology/synteny mapping—finding orthologous regions among genomes and organizing these evolutionary counterparts into a coherent global picture—is fundamental to studies of comparative genomics. With the increasing number of completely sequenced genomes and thus the increase in comparisons of massive nucleotide sequences, the need for orthology/synteny mapping methods of high sensitivity/specificity and high efficiency becomes even more compelling.

Results: First we have developed the UniMarker (UM) method for synteny mapping of large genomes that are closely related, such as the human and mouse. In this method, the occurrence spectra of genome-wide unique 16mer sequences present in both the human and mouse genome are used to directly detected orthologous genomic segments. Being sequence alignment-free, the UM method is very fast and the high-quality human-mouse synteny maps based on DNA comparisons can be completed in a few hours on single desktop computer. Second, we propose a new type of DNA sequence seed for use in orthology mapping of not closely related genomes. We call our seeds α-pairs, where α is an integer equal to or greater than the number of times any qualifying seed can be found in the compared genomes. These copy number-based seeds are thus distinct from the well-known length-based seeds, such as the fixed-length k-mer seeds or the maximal exact match (MEM) seeds which have a length $\geq k$. We present a linear time algorithm to efficiently retrieve α-pairs in two given genomic sequences based on enhanced suffix arrays. A comparison of the results using α-pairs with those using length-based seeds for their ability to detect the orthologues annotated by Ensembl and COG for several vertebrate genomes/chromosomes and for prokaryote genomes of long evolutionary distances suggested that orthology seeding using copy number can achieve a higher sensitivity and better efficiency than orthology seeding using length. Moreover, we extend the α-pair method to generate discontiguous wobble seeds of maximal length with copy number constraints. The comparative results of ROC curves for human chr.15 vs. mouse chr.7, chicken chr.10, and pufferfish genome showed that the discontiguous wobble α-pairs achieved significantly better performances than spaced k-mer seeding methods tested.

Keywords: comparative genomics, synteny mapping, orthology mapping, sequence alignment, seeding, suffix array.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Orthology mapping is to find orthologous regions among genomes and synteny mapping is to organize these evolutionary counterparts into a coherent global picture. Similar to Rosetta stone, orthology/synteny maps intend to provide cross-references among different DNA languages of their species as a foundation for functional analogy and evolutionary studies. As the number of completely sequenced genomes continues to increase rapidly, orthology identification at the nucleotide level in both coding and noncoding regions of genomes is becoming an indispensable approach for studying genome evolution and for genome annotation (Deway and Pachter 2006). However, orthology identification and synteny mapping based on nucleotide comparisons have to face several challenging issues. 1) The nucleotide comparisons between genomes are computationally demanding, especially for large genomes such as the human (~3Gb) to mouse (~3Gb). 2) There are plenty noisy local similarities between nonorthologous locations, such as repeats and irrelevant ancestral duplications. 3) The evolution over time makes things complicated, such as sequence divergence, gene duplications and losses, duplications and deletions of genomic regions, genomic rearrangements and microrearrangements, and genome duplication (Jaillon *et al*. 2004). Thus, the need for developing orthology/synteny mapping methods of high sensitivity/specificity and high efficiency for large genomes of different evolutionary distances becomes even more compelling.

1.2 Dissertation organization

We introduced the necessary background in next chapter. In chapter 3, we described the UniMarker (UM) method for synteny mapping for closely related genomes. The UM method is very efficient by looking up only genome-wide unique seeds of fixed length and an alignment-free design for sequence comparison and the details of the method are given in section 3.2. The experiment results of the UM method are located in section 3.3, which showed that the whole synteny mapping process of giga-base genomes, such as human vs. mouse, can be completed in a few hours on single desktop computer. In chapter 4, we proposed the α-marker method based on enhanced suffix arrays for orthology seeding using maximal exact matches with copy number constraints. The definitions and algorithms of the α-marker method are stated in section 4.2. Comparisons of different contiguous seeding methods to detect orthologues are presented in section 4.3. In chapter 5, we extended the α-marker method to generate discontiguous wobble seeds with copy number constraints and described the method in section 5.2. Different contiguous and discontiguous seeding methods are compared using ROC curves and colinear identities per orthologue in section 5.3. Finally in chapter 6, we made the discussion and conclusions for this dissertation.

Chapter 2

Background

2.1 Homology and synteny

2.1.1 Homology

Homology is a very important term in biology and features are said to be homologous if they share a common evolutionary origin (Theiβen 2002). When homology is applied to genes or nucleotide sequences, homologues are genes (or nucleotide sequences) derived from a common ancestor gene (or nucleotide sequence). There are three disjoint subtypes of homology depending on what kind of evolutionary events it resulted from: orthology, paralogy and xenology, where orthology resulted from speciation events, paralogy resulted from duplication events, and xenology resulted from inter-species transfer of genomic materials (Fitch 2000). Further, since orthologous relationships are not limited to one-to-one (Theiβen 2002), we can divide orthologues into 1-to-1/mono-orthologues and co-orthologues according to if there are no duplication events after speciation events. More detailed definitions of homology are well described by Koonin (2005). In Figure 2.1, we provide examples to illustrate orthologues, co-orthologues, in-paralogues, and out-paralogues (Kooin 2005). Suppose that B and C are two genomes to be compared and genome A is the last common ancestor of B and C in the species tree shown in Figure 2.1. Let there be two genes g1, g2 in A, where g2 was duplicated from g1 before the speciation, there be two genes g1, g2 in B without duplications after the speciation, and there be three genes g1, g2, g3 in C, where g3 was duplicated from g2 after the speciation. Then, g1 of B and g1 of C form 1-to-1

orthologue, and g2 of B and g2, g3 of C form co-orthologues since the duplication from g2 to g3 was after the speciation. Within genome C, genes g2 and g3 form an in-paralogue and genes g1 and g2 form an out-paralogue because the former duplication happened after the speciation and the latter duplication happened before the speciation.

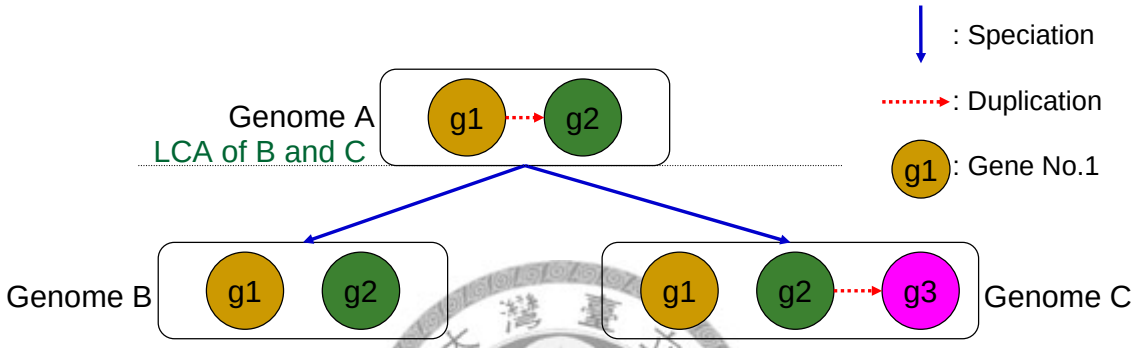


Fig. 2.1 Illustration of homology

2.1.2 Synteny

Synteny (literally "same thread") indicates the condition of two or more genes/regions being on the same chromosome within one species. When synteny is applied to inter-species comparisons, conserved synteny refers to two or more orthologous (including co-orthologous) regions that are syntenic in two or more species, without regard to their order on each chromosome (Ehrlich *et al.* 1997, Frazer *et al.* 2003). Operationally speaking, we define components related to conserved synteny in a bottom-up hierarchical way, including orthologous anchors, conserved segments, and syntenic blocks. Given two compared genomes, an orthologous anchor of them is a pair of gene/region from different genomes that are significantly similar and believed to be

orthologous (including co-orthologous). A conserved segment contains two or more orthologous anchors that are syntenic (i.e., on the same chromosomes) and contiguous (i.e., no interrupt by other anchors) on the both compared genomes and are arranged collinearly with preserving order and orientation. A syntenic block consists of two or more conserved segments that are syntenic and contiguous on the both compared genomes regardless their orientation. Hence, synteny mapping is to locate and group regions that are orthologous/co-orthologous among genomes by order and/or orientation.

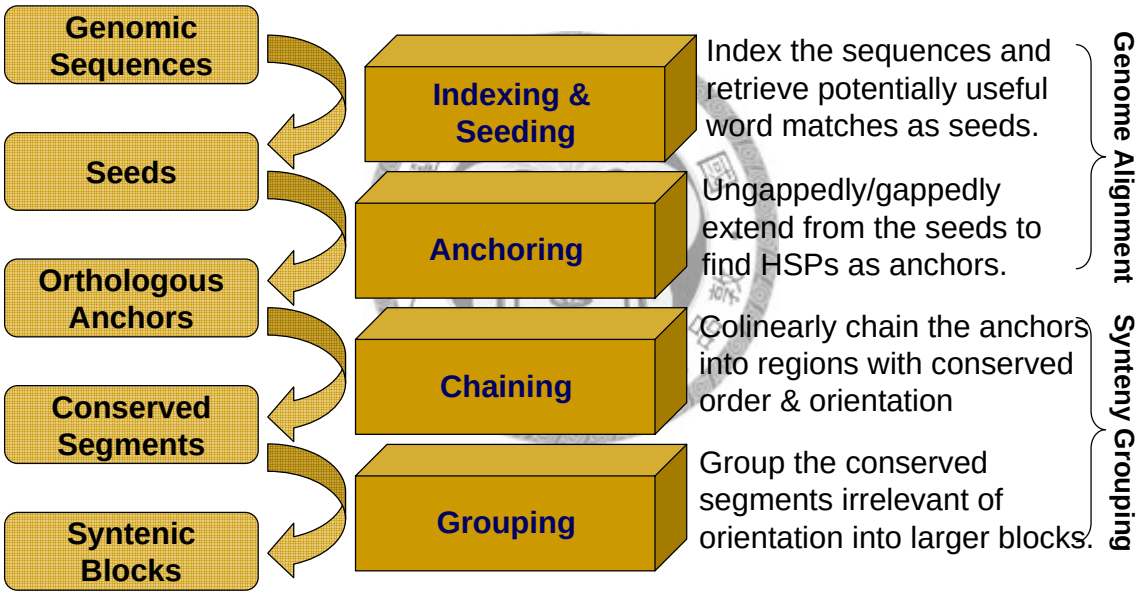


Fig. 2.2 Common stages of large-scale genome comparison and synteny mapping

In Figure 2.2, we introduce the common stages of large-scale genome comparison and synteny mapping. The major four stages are 1) indexing & seeding, 2) anchoring, 3) chaining, and 4) grouping. First, we can index the input genomic sequences and retrieve potentially useful word matches of the input sequences as seeds. Then, we can extend those seeds ungappedly and/or gappedly to obtain longer high-scoring segment pairs

(HSP, Altschul *et al*. 1997) as orthologous anchors. Third, we can colinearly chain those anchors into conserved segments with preserving anchor order and orientation. Finally, we can group those conserved segments into larger syntenic blocks, regardless the orientations of the conserved segments.

2.2 Index-based sequence comparison

The index-based alignment method has revolutionized sequence comparison and has led to numerous tools for different purposes (Batzoglou 2005). Index-based alignment methods first build indices for one or all of the compared sequences and then retrieve seeds—often word matches or transformed pieces of sequence matches—from the indices to obtain the alignments for inferring homology. Since seeding is necessarily the first step of all index-based genome alignment methods (Ureta-Vidal *et al*. 2003), the strategy employed for selecting seeds and their retrieval is fundamental to the performance of genome alignment methods (Brown *et al*. 2004). In Figure 2.3, we presented taxonomy of index-based sequence comparisons by seed design.

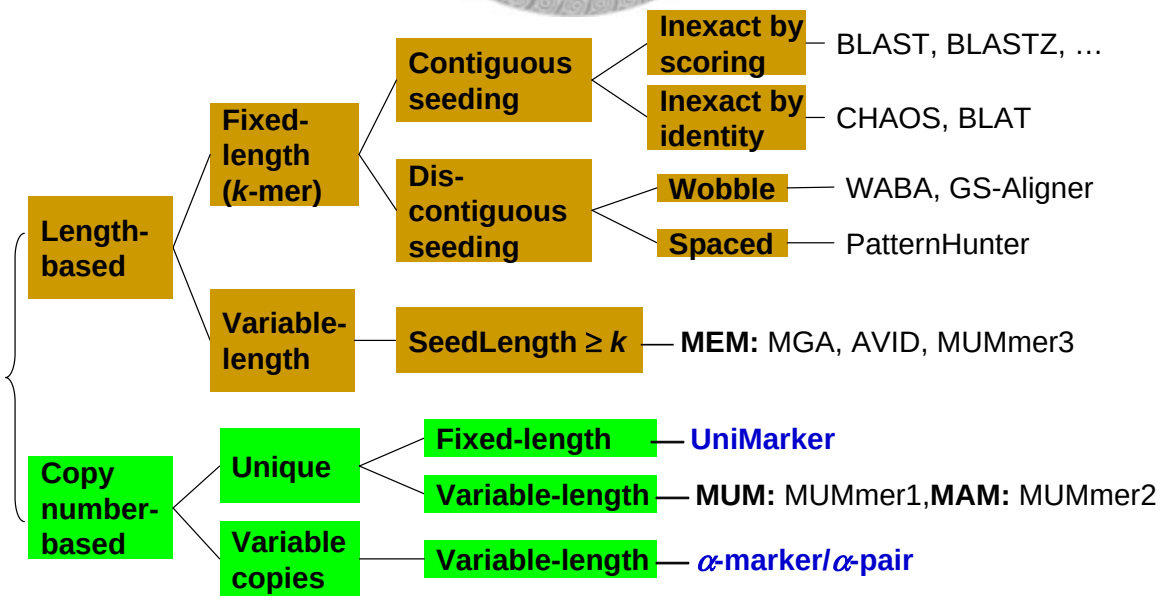


Fig. 2.3 Index-based sequence comparison: taxonomy by seed design

Most seeding strategies developed thus far are length-based, i.e., seeds are selected via fixed-length or variable-length constraints. The *k*-mer (aka *k*-tuple) strategy using exact matches of sequence words of a fixed-length as seeds is perhaps the most popular, which is adopted in general-purpose sequence comparison methods such as FASTA (Lipman and Pearson 1985) and BLAST (Altschul *et al*. 1990), and also in various genome comparison programs, such as WABA (Kent and Zahler 2000), BLAT (Kent 2002), PatternHunter (Ma *et al*. 2002), CHAOS (Brudno *et al*. 2002), BLASTZ (Schwartz *et al*. 2003), and GS-Aligner (Shih and Li 2003). Then we categorized BLASTZ, CHAOS, and BLAT into two sub-branches of fixed-length contiguous seeding: providing inexact matching by scoring or identity as shown in Figure 2.3. In addition, one notable advance of *k*-mer approach is discontiguous seeding, such as WABA (Kent and Zahler 2000) and PatternHunter (Ma *et al*. 2002), which will be detailed in chapter 5.

Another length-based seeding strategy employed in genome comparison programs, such as MGA (Höhl *et al*. 2002), AVID (Bray *et al*. 2003), and MUMmer3 (Kurtz *et al*. 2004), uses maximal exact matches (MEMs) (Höhl *et al*. 2002), aka maximal pairs (Gusfield 1997), which include all exact matches of maximal lengths greater than or equal to *k*. By excluding numerous redundant matches, which are particularly abundant in short-length words, MEM methods can acquire a better efficiency of seeding than *k*-mer methods for large-scale sequence comparison (Chain *et al*. 2003).

To gain more on seeding efficiency, Delcher *et al*. (1999) consider a subset of MEMs, using only the maximal unique matches (MUMs) to align two genomes, where a MUM is a shared substring occurring exactly once in each of the two compared genomes and it cannot be extended without introducing mismatches (i.e., maximal

length). In addition, our work in chapter 3 demonstrated that use of fixed-length seeds constrained by the one-to-one mapping (called UniMarkers, which, for their fixed length, are a subset of MUMs) is sufficient to construct a high-quality human-mouse synteny map with very high efficiency (Liao *et al*. 2004). Furthermore in chapter 4, in purpose to detect orthologous as well as co-orthologous regions for not closely related species, we designed a new seeding method, called α-marker/α-pair method, by relaxing the constraint of genome-wide uniqueness in MUM and UniMarker to allow variable copies in an upper bound way. The above mentioned methods, as shown in the bottom of Figure 2.3, make the branch of copy number-based seeding more solid and useful.

Chapter 3

The UniMarker method for synteny mapping

3.1 Introduction

With the number of completely sequenced genomes increasing rapidly, comparative genomics is becoming an indispensable approach for genome annotation and for studying genome evolution. Essential to this approach is whole-genome alignment, which is computationally demanding, particularly for large genomes, such as those of mammals. Thus, despite recent advances, scores, or even hundreds, of computing processors are still required to compare the human and mouse genomes in a time period of hours or days (Waterston *et al.*, 2002; Schwartz *et al.*, 2003), a practical time scale for doing competitive research in such a rapidly evolving field as genomics. Moreover, there appears to be considerable discrepancy in the various human-mouse synteny maps created independently by several research groups (Waterston *et al*., 2002; Gregory *et al*., 2002; Clamp *et al*., 2002), even though they may use similar alignment algorithms and strategies (Ureta-Vidal *et al*., 2003).

As many more large genomes will be sequenced in the next few years (Ureta-Vidal *et al*., 2003), there is a pressing need to develop a whole-genome alignment tool that can render the task feasible and practical using minimal computing facilities, such as a single desktop computer. To achieve this goal, methods that deviate significantly from existing approaches using sequence alignment, such as BLAST (Altschul *et al.*, 1990) or BLAST-derived algorithms (Schwartz *et al.*, 2003; Zhang *et al.*, 2000; Kent 2002;

Ma *et al.*, 2002), merit exploration.

Various articles have demonstrated that the use of a hash table (Schuler 1997; Ning *et al.*, 2001) or suffix-tree (Delcher *et al.*, 2002; Bray *et al.*, 2003) can significantly speed up the computation time required for sequence mapping. Our previous work (Chen *et al.*, 2002) showed that, by matching unique 15-mer words (those that appear exactly once in the genome and are therefore called UniMarkers or UMs), it is possible to dispense with the usual requirement for sequence alignment and to genomically position the entire database of human single nucleotide polymorphism (SNP) sequences in just a few days of computing time on a single desktop computer. In the present study, we introduced a new concept of using UMs to detect sequence orthologues without doing sequence alignment and extended the UM method for whole-genome synteny mapping.

To align two very long DNA sequences, such as those of metazoan genomes, the most common approach starts by finding the so-called high scoring pairs (HSPs) of sequence fragments that are derived from words matched by consecutive (Altschul *et al.*, 1990; Zhang *et al.*, 2000) or spaced (Schwartz *et al.*, 2003; Ma *et al.*, 2002) matching models. These HSPs, in which a word or segment in one sequence may have multiple matches in the other sequence, then serve as seeds, which are subsequently filtered and combined to identify a set of longer segments that are thought to be orthologous

between the two sequences. In the final step, these segments, often called anchors or landmarks, are extended or processed to yield an alignment or mapping of the two sequences (Ureta-Vidal *et al.*, 2003). Our UM method differs from these approaches by avoiding the time-consuming step of finding and processing the HSP seeds; instead, orthologues anchoring segments are detected directly from a genome-wide occurrence spectrum of UMs common to the two genomes compared. Consequently, and as detailed below, the UM method is very fast and can map the entire human genome against the entire mouse genome, and vice versa, in just one day on a single Pentium IV personal computer. This is a considerable time saving, since the time required is about one-tenth or one-hundredth that using, for example, the approach of MGSC (Mouse Genome Sequencing Consortium) (Waterston *et al.*, 2002). To evaluate the quality of the resulting UM human-mouse map, it was compared with the MGSC map and with that produced by the Ensembl team (Clamp *et al.*, 2003; Hubbard *et al.*, 2002). The UM map was shown to be in excellent agreement with the MGSC map, missing only a few small MGSC segments, while having several small unique segments of its own. The agreement with the Ensembl map was also very good, though not as good as that with the MGSC map. Sequence alignment using BLASTZ (Schwartz *et al.*, 2003) on segments that were map-unique or disagreed between maps indicated that the UM method, despite being sequence alignment-free, achieved high specificity and sensitivity

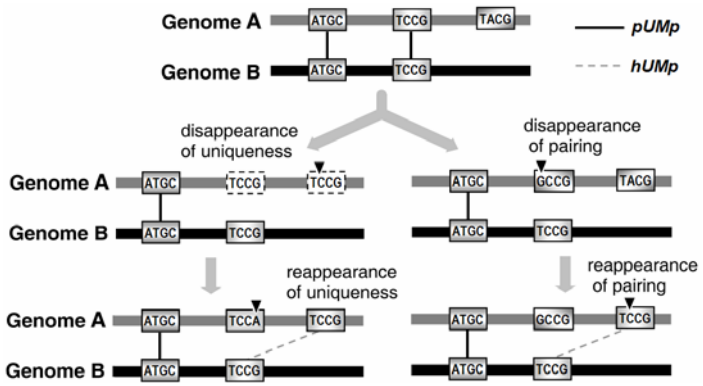
in mapping the two mammalian genomes.

3.2 Methods

3.2.1 pUMp vs. hUMp

Orthologous regions, by definition, are homologous regions shared by two genomes from a speciation event. The basic idea of our approach is that, between two genomes, orthologous regions should share more UniMarker pairs (UMps; an UMp connects identical UMs in both genomes) than non-orthologous regions. However, there are two kinds of UMp, those inherited from a common ancestor, hereafter referred to as primitive UMps (pUMps), and those that have arisen by random mutation, referred to as homoplastic UMps (hUMps) (Figure 3.1). Although it is not possible to tell whether a given UMp is a pUMp or a hUMp, it can be distinguished as a collective group, as illustrated in Figure 3.1. This is because, by definition, pUMps can exist only between orthologous regions, whereas hUMps can exist between any two regions, be they orthologous or not. Consequently, pUMps can provide a signal for pairs of orthologous regions against a background noise of hUMps, and, as long as the signal/noise ratio is sufficiently high, i.e., the evolutionary distance between the two genomes is not too great, orthologous pairs should be detectable by analyzing the UMp distribution in the two genomes.

(A)



(B)

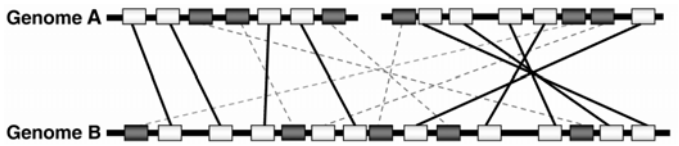


Fig. 3.1 The two types of UMp. All UMps shared by segments from two different genomes can be classified into two types, those that have descended from a common ancestor, called primitive UMps (pUMps; black solid lines), and those that have arisen by random mutation, called homoplastic UMps (hUMps; gray dashed lines). (A) Following evolutionary changes, a certain pUMp could change its pairing randomly, resulting in a pUMp evolving into a hUMp. UMs (illustrated by four-letter words) found in both genomes are represented by shaded boxes. The site of mutation causing a change in UM pairings is marked by a black triangle. (B) The distribution of pUMps and hUMps. When two genomes are compared, orthologous genomic segments will share both pUMps (shown as white boxes) and hUMps (shown by black boxes), but any two evolutionarily unrelated regions (e.g., the first segment of genome A and the second half of the genome B segment) can only share hUMps.

3.2.2 Occurrence spectra of UMps and anchoring islands

A simple, but efficient, method to identify *k*-mer UMs in the human genome has been described (Chen *et al.,* 2002). This method was used in the present study to identify 16-mer UMs for each of several assemblies of the human genome and for the draft mouse genome sequence. Those UMs common to a particular assembly of the human genome and the mouse genome were extracted; each of these constitutes an UMp, as defined above.

The UM method for mapping two genomes, A and B, involves the following. Each chromosome of genome B is divided into a set of minimally overlapped fragments, each containing an equal number of UMps, which, in this work, was set at 300,000, i.e., a number slightly greater than that (~290,000) on the human Y chromosome (consequently, the entire human Y chromosome was a fragment). We then scan genome A using a sliding window of 50 kb and a moving step of 10 kb to compute M_{ij}, the ratio of the number of UMps common to both the *i*th window of genome A and the *j*th chromosomal fragment of genome B (N_{ij}) to the total number of UMps found in the *i*th window of genome A (N_i) (i.e. $M_{ij} = N_{ij}/N_i$). The values of these parameters, and of those described below, were empirically determined in trial runs to minimize the computational cost while maintaining good resolution in the resulting human and mouse synteny map.

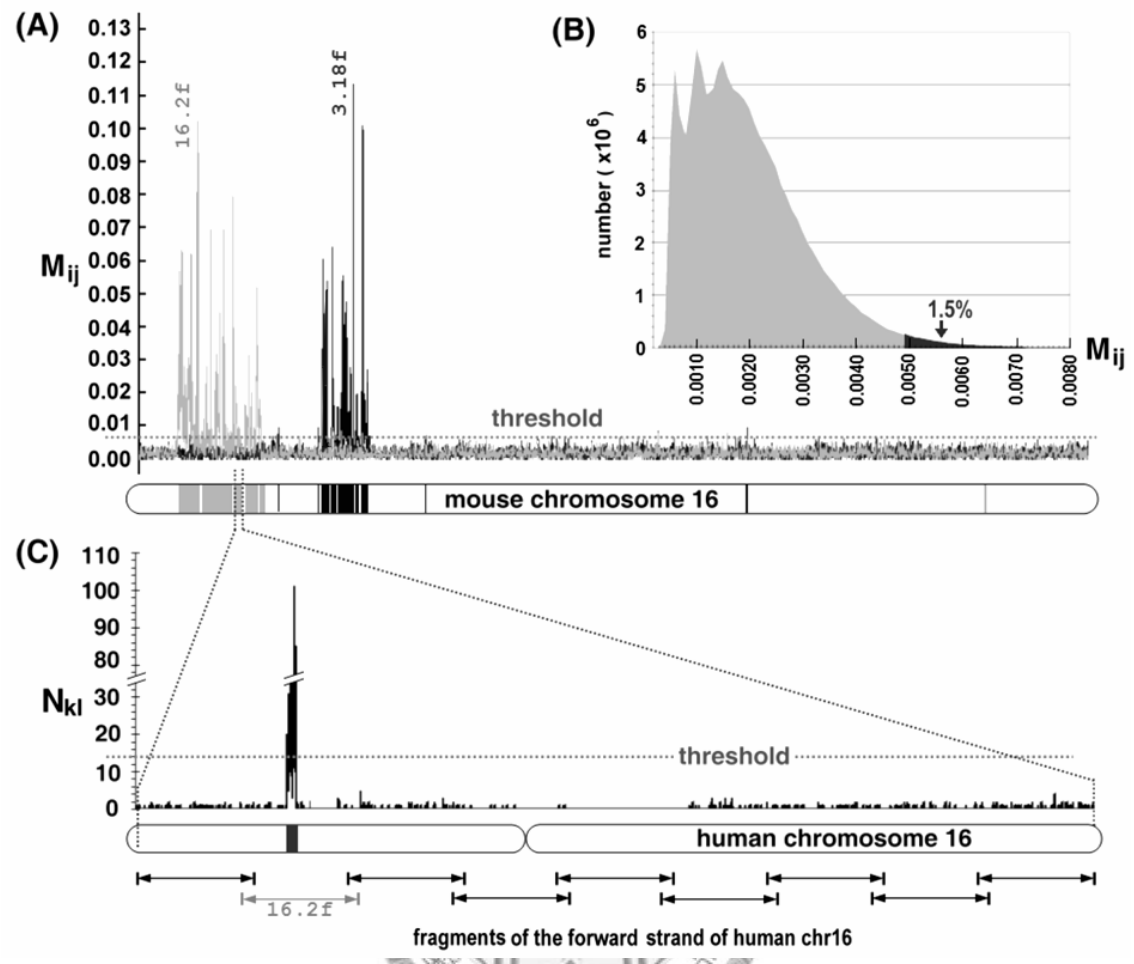


Fig. 3.2 Identification of the anchoring islands. (A) The M_{ij} spectrum (see text for definition) for mouse chromosome 16 computed from two human chromosomal fragments, denoted by 16.2f (the 2nd fragment on human chromosome 16 in the forward orientation) and 3.18f. The detected islands, regions containing at least four consecutive overlapping windows (each of 50 kb and with a M_{ij} value above threshold, see text) are labeled as vertical bars on the mouse chromosome shown below the x-axis. The boundaries for each island were set at the midpoint of the first and last of its consecutive windows. (B) The distribution of M_{ij} [for all windows (i) and all chromosomal fragments (j), see text]. The lower boundary of the top 1.5% of the distribution (dark area) was chosen as the M_{ij} threshold in the present work. (C) The N_{kl} spectrum for determining the matching island on the human chromosome, which, as indicated, was divided into minimally overlapped fragments with equal number of UMs, rather than base pairs (see text). For each mouse chromosome, such as chromosome 16 shown here, there were a total of 612 M_{ij} spectra, as the human genome was divided into 612 chromosomal fragments (half forward and half backward); for clarity, only two are shown in (A).

As illustrated in the example in Figure 3.2, the M_{ij} spectrum allowed us to find orthologous regions, hereafter referred to as anchoring islands, without doing sequence alignment. For a segment to qualify as an anchoring island, at this stage in genome A

only (Figure 3.2A), we specified that at least four consecutive windows must have a M_{ij} value in the top 1.5% of all M_{ij} (see Figure 3.2B) to suggest the presence of a pUMp, or orthologous relationship, between these windows of genome A and a chromosomal fragment of genome B. To pin down the region in this chromosomal fragment of genome B with which the anchoring island of genome A was orthologous, we moved the sliding window to genome B, and operated it on the fragment-containing chromosome to compute N_{kl}, the number of UMps shared by the *k*th window (on the chromosome of genome B) and the *l*th island (on genome A). The N_{kl} spectrum (Figure 3.2C) allowed us to delimit the matching anchoring island on genome B, which was specified as containing at least two consecutive windows with (i) N_{kl} values of at least 25 or (ii) N_{kl} values of at least 10 and within the top 3% of all N_{kl} for that particular *l*th island of genome A. Note that, for this stage, there was no need to compute N_k, or N_{kl}/N_k (i.e., M_{kl}), and the reason for the expansion to include the whole chromosome, instead of just the fragment, in the computation of N_{kl} was to provide sufficient background noise (hUMps) to distinguish the signal (pUMps). For multiple matches, i.e., when two or more matching anchoring islands were found on the fragment of genome B, the procedure for computing N_{kl} was repeated after switching the sliding window back to operate on the anchoring island-containing chromosome of genome A. This procedure was repeated until all anchoring islands were uniquely matched between the two genomes. For the present work on the human and mouse genomes, we found that multiple matches occurred in about 30% of cases; most of these could be resolved after N_{kl} was calculated for the second time, and all could be resolved after the fourth calculation.

3.2.3 Overlapped anchoring islands

A few (500-800, or 4-7%, depending on the version of genome assembly used) of the resulting anchoring islands overlapped; this was due to the pUMp signal being independently detected in overlapping windows. There were four types of such overlaps (Figure 3.3). For the first type, of partial overlaps, which accounted for ~60-75% of overlaps, we simply set the boundary of the anchoring island at the midpoint of the overlap. The second and third types (accounting for 20-40% of overlaps) occurred when a small island (usually < 100 kb) was embedded in a large island. Further analysis indicated that embedded islands of the second type, which comprised ~80% of the embedded cases, probably resulted from lineage-specific duplication, while those of the third type resulted from micro-rearrangements. Accordingly, we discarded embedded islands of the second type, but kept those of the third type and split their encompassing island into three, as illustrated in Figure 3.3. The fourth type occurred when a very small island (~40 kb) of one genome contained two separable clusters of UMps, each of which was mapped to one of two distinct, usually even smaller, islands of the other genome. The fourth type was rare, accounting for less than 2% of the overlaps. For sake of computational convenience and automation, we kept the first of the two pairings and discarded the other.

Although the use of a smaller window and moving step can eliminate most of the overlaps, particularly those of the first type, this would force the method to operate on

fewer UMs, which could decrease the signal/noise ratio, especially for regions containing a lower density of UMs (e.g., < 1,000 UMs/50 kb).

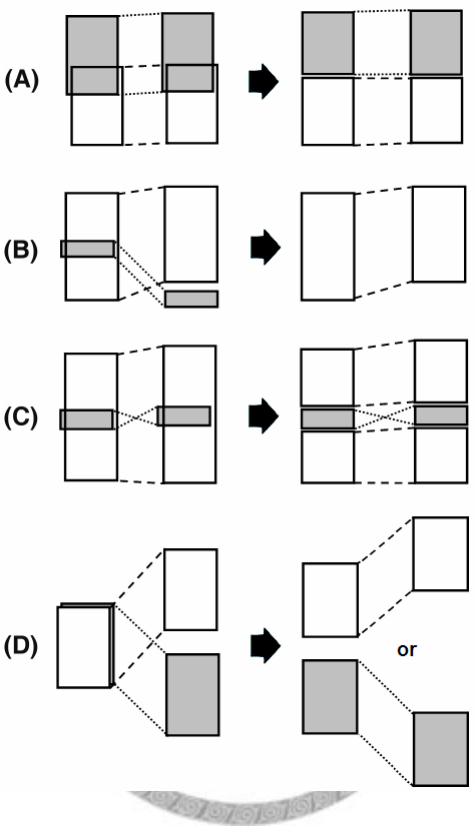


Fig. 3.3 Schematic illustration of the rules applied to resolve overlaps in anchoring islands. (A) Partially overlapped islands. (B) Embedded islands due to lineage-specific duplication. (C) Embedded islands due to micro-rearrangement. (D) Islands with identical boundaries, but distinct pairing partners.

3.2.4 Bidirectional mapping

At this stage, we had a set of non-overlapping, one-to-one matched, anchoring islands for genomes A and B. We called this set the A->B set, since the M_{ij} for this set was computed on windows of genome A. To further reduce the likelihood of the identified anchors being false positives, we also computed the B->A set, using identical procedures and parameters to those described above, and extracted the overlaps of the

two sets. The bi-directional mapping helped us set the thresholds for M_{ij} and N_{kl} (see above), using which more than 95% of the mapped anchoring islands were either identical or substantially overlapped between the two directions.

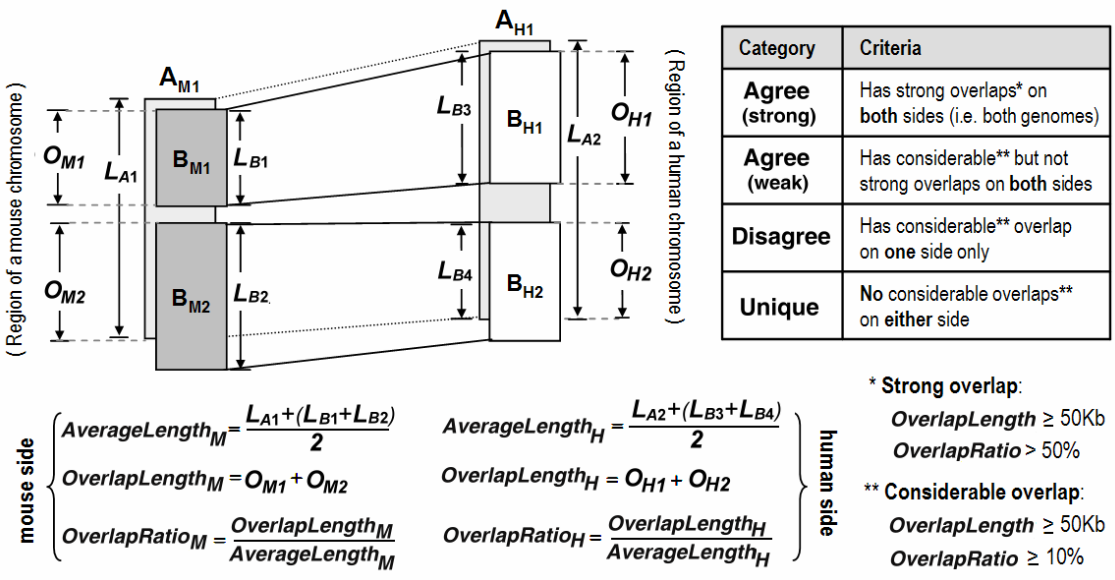
3.2.5 Conserved segments and syntenic blocks

The bi-directionally mapped and non-overlapping anchoring islands were then merged into conserved segments for any two adjacent islands in one genome that were also adjacent, as well as in the same orientation, in the other genome (see Nadeau and Sankoff (1998) for definitions of "conserved segment" and "syntenic block" (aka "conserved synteny")). Finally, the resulting conserved segments were grouped into syntenic blocks, each of which consisted of conserved segments that were contiguously matched, irrespective of the order and the orientation of their matching, in both genomes and on a single chromosome.

3.2.6 Comparison with other maps

It is not a trivial process to compare two different synteny maps, because different degrees of concordance may arise for conserved segments that are equivalent between the two maps on either of the two genomes. We therefore devised a set of parameters to assign equivalent (i.e., overlapped) conserved segments to four categories (see Figure 3.4): 'Agree (strong)', 'Agree (weak)', 'Disagree', and 'Unique', with decreasing degrees of overlap. The main distinction between the 'Agree' and 'Disagree' category was whether a substantial overlap in the segments was shared in both, or just one, of the two genomes; those that were not substantially overlapped in either genome, or were overlapped, but not in the same orientation, were assigned to 'Unique'. For the comparison with the MGSC and Ensembl maps, the same versions of the genome assembly for either human or mouse used in those maps were used to produce the

corresponding UM maps. These genome assemblies were retrieved from ftp://ftp.ncbi.gov/genomes/H_sapiens/ and ftp://ftp.ncbi.nih.gov/genomes/M_musculus/ at the National Center for Biotechnology Information (NCBI). The MGSC map, i.e., the genomic start and end positions and the orientation of mapped conserved segments, was provided by Michael Kamal (Whitehead Institute, MIT). The Ensembl map was downloaded from http://www.ensembl.org/Homo_sapiens/syntenyview/ and its segments parsed.



Category	Criteria
Agree (strong)	Has strong overlaps* on **both** sides (i.e. both genomes)
Agree (weak)	Has considerable** but not strong overlaps on **both** sides
Disagree	Has considerable** overlap on **one** side only
Unique	**No** considerable overlaps** on **either** side

Fig. 3.4 Parameters and criteria used to compare two human-mouse synteny maps. The letter notations are as follows: A for map A, B for map B, H for human, M for mouse, O for overlap, and L for length. In principle, the number of segments from one map to overlap with one segment of the other map on either side of the two genomes is not limited to two, but, for the purpose of illustration, two are used here.

3.2.7 BLASTZ evaluation

To evaluate the segments classed as 'Disagree' or 'Unique' between two maps, we subjected them to BLASTZ (Schwartz *et al.*, 2003) sequence alignment, using parameters B=2, C=0, T=1, and K=5000, 9000, or 12000. Each of the resulting alignments was displayed as a dot plot using the alignment viewer, Laj (Wilson *et al.*,

2001), inspected, and assigned to one of five outcomes (see Figure 3.5 for illustrative examples), “Concordant”, “Shifted”, “Multiple”, “Reversed”, and “Unsupported”. Those that showed no clear evidence of homology were considered “unsupported” by sequence alignment and were probably false positives. All the assignments could be made without much ambiguity, although, for a few segments with few and very small patches of matches in the dot plot, their assignment to one of the last four outcomes could be subjective.

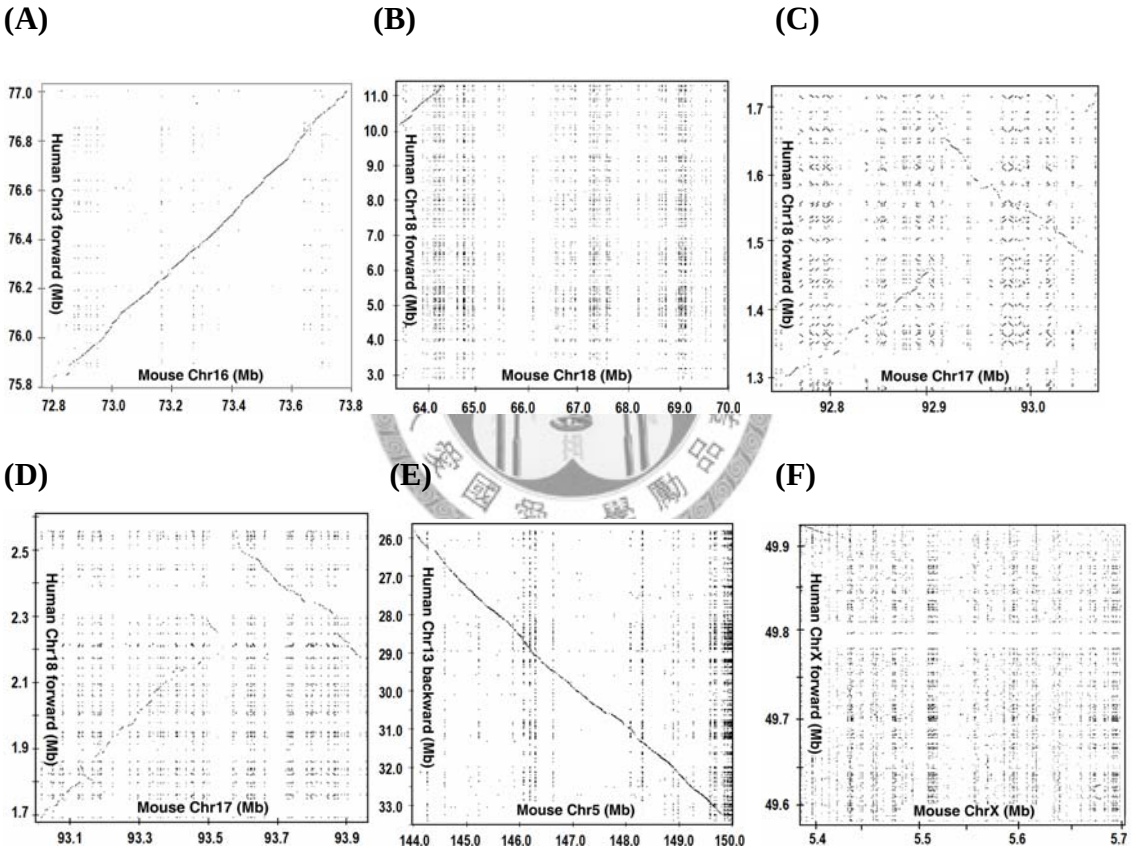


Fig. 3.5 Examples of BLASTZ alignment, shown as a dot plot, of conserved segments assigned as “Disagree” or “Unique”. (A) Concordant, (B) Shifted, (C) and (D) Multiple, (E) Reversed, (F) Unsupported. (A) and (E) segments are from the UM map, (B) and (E) segments from the Ensembl map, and (D) and (F) segments from the MGSC map. For visual clarity, BLASTZ parameter K (threshold for the maximal segment pair score) was set at 12000 in cases (B) and (D), 9000 in cases (A), (C), and (F), and 5000 in case (E).

3.2.8 Software

Computer modules for the UM method and synteny map visualization were written in C\C++, and Delphi/Object Pascal. The run-time to produce a human-mouse map, which included both the bi-directional mapping and the merging of anchoring islands into conserved segments and syntenic blocks, was ~7 hours on one personal computer (2.8 GHz Pentium IV, 2GB memory). The software is freely available at the Web site http://synteny.iis.sinica.edu.tw/um/.

3.3 Results

3.3.1 Maps from various versions of the human genome

The speed of the UM method for producing a whole-genome synteny map allowed us to produce multiple maps resulting from different versions of genome assembly. Maps using different human genome assemblies differ mainly in the number of small conserved segments which decreased with each update of the genome (Supplement Figure 3.6). This corroborates the argument that errors in sequence assembly are more likely to produce artifactual micro-rearrangements than to affect large (e.g., > 1Mb) synteny blocks (Pevzner and Tesler 2003). Given the results shown in Figure 3.6, we can expect a further reduction in the number of small conserved segments when a 'finished' mouse genome becomes available.

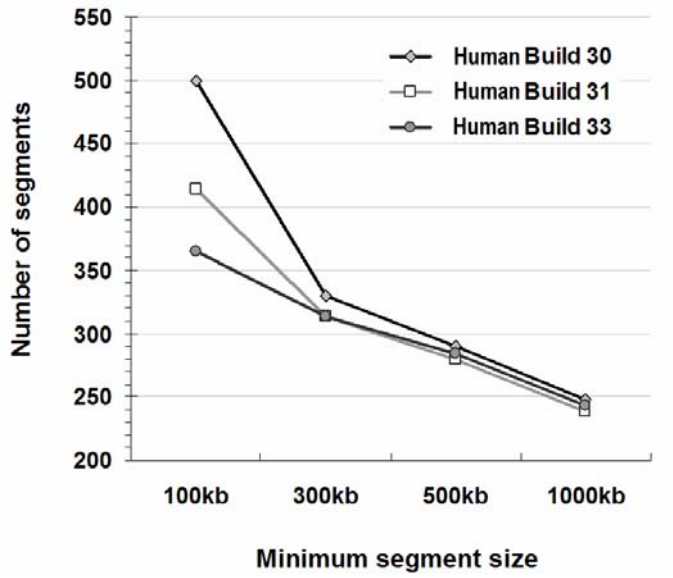


Fig. 3.6 Number of conserved segments identified by the UM method using different versions of the human genome (all mapped against NCBI mouse Build 30).

Some parameters for the UM map using the 'essentially complete' human genome (NCBI build 33) and the mouse genome NCBI build 30 (the only NCBI build for mouse available at the time of this work) are summarized in Table 3.1. Maps using human builds 30 and 31 gave quite similar results (data not shown). For the conserved segments and synteny blocks, these data, except for those for N50, are quite comparable with those reported by MGSC (Waterston *et al.*, 2002); in contrast, the 10,999 anchoring islands are only a fraction of the 558,000 'landmarks' (high scoring and bidirectional best sequence matches) identified by MGSC. Since the two sets of syntenic anchors eventually produced very similar maps (details below), our much larger 'islands' (846.9 Mb total length covering 33.9% of the mouse genome; Table 3.1) are, in effect, clusters of the 'landmarks' obtained by sequence alignment using PatternHunter (Ma *et al.*, 2002) (188 Mb total length and 7.5% mouse genome coverage (Waterston *et al.*, 2002)).

Table 3.1 Size and Genome Coverage of Anchoring Islands, Conserved Segments and Syntenic Blocks *.

		mouse	human
10,999 anchoring islands	• Average	77.0 kb	81.8 kb
	• N50	50.0 kb	50.0 kb
	• Largest	1.27 Mb	1.30 Mb
	• Total Length (% genome) **	846.9 Mb (33.9%)	899.9 Mb (31.8%)
	• Spacing Ave.	150.1 kb	182.2 kb
	• Spacing N50	70 kb	80 kb
365 (≥100kb) conserved segments	• Average	6.33 Mb	7.08 Mb
	• N50	2.46 Mb	2.94 Mb
	• Largest	64.49 Mb	79.65 Mb
	• Total Length (% genome)	2309.3 Mb (92.3%)	2585.3 Mb (91.3%)
224 syntenic blocks	• Average	10.55 Mb	12.01 Mb
	• N50	4.78 Mb	5.58 Mb
	• Largest	146.01 Mb	143.27 Mb
	• Total Length (% genome)	2363.8 Mb (94.5%)	2689.1 Mb (94.9%)

* These data are for the UM human-mouse synteny map using the 'essentially complete' human genome (NCBI build 33) and the draft mouse genome (NCBI build 30).

** Genome size was calculated by omitting the telomeres, centromeres, and gaps between supercontigs. (Mouse: 2.501Gb; Human: 2.832Gb)

3.3.2 Comparison with maps produced by MGSC and Ensembl

As the key component of a synteny map is a list of conserved segments, the easiest way to compare two synteny maps is to compare two corresponding lists of conserved segments. Using the criteria for comparing two maps described in section 3.2.6, the comparison of the results for UM vs. MGSC and UM vs. Ensembl is presented in Tables 3.2 and 3.3, respectively. A graphical overview of these results is also presented in

Figure 3.7. As can be seen, the UM map agreed well with both the MGSC and the Ensembl maps, having ~99% of the mapped regions cross-covered with the former (Table 3.2) and up to 95% with the latter (Table 3.3). Furthermore, the vast majority of the 'Agree' segments were in strong agreement (i.e. high degree of overlap; see Figure 3.4), and the 'Disagree' or 'Unique' segments were mainly relatively small segments (see also Table 3.4), the largest being a few Mb in the comparison with the MGSC map and 24 Mb in the comparison with the Ensembl map. The somewhat smaller genome coverage and the smaller conserved segments obtained using the UM map were probably due to the fact that, unlike in the other two maps, the anchoring islands were not extended to include as much alignable sequence as possible.

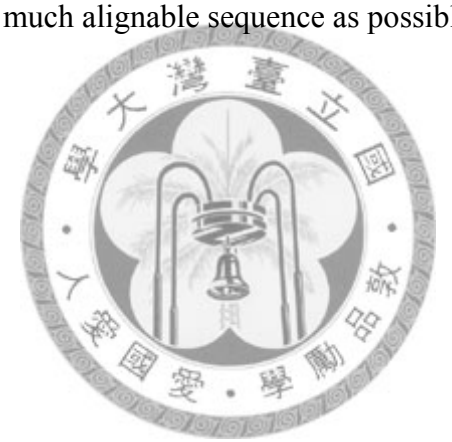


Table 3.2 Comparison between the UM map and the MGSC map on conserved segments*.

		Agree			Disagree	Unique	Total	
		Strong		Weak				
UM		310		8	0	12	330	
MGSC		308		8	0	26	342	
		size (Mb)	%mapped	size (largest)	size (largest)	size (largest)	size	%genome
UM	mouse	2260.6	99.2%	9.5 (3.1)	0.0 (—)	9.4 (2.6)	2279.5	91.7 %
	human	2512.2	99.0%	7.6 (2.8)	0.0 (—)	19.1 (3.2)	2539.0	90.3 %
MGSC	mouse	2321.7	98.7%	11.6 (0.8)	0.0 (—)	19.7 (4.2)	2353.0	94.6 %
	human	2583.8	98.5%	11.7 (0.5)	0.0 (—)	28.2 (3.9)	2623.6	93.3 %

* Human assembly NCBI build 30 vs. mouse assembly MGSCv3, with the minimum segment size cut at 300kb

Table 3.3 Comparison between the UM map and the Ensembl map on conserved segments*.

		Agree			Disagree	Unique	Total	
		Strong		Weak				
UM		261		23	10	71	365	
Ensembl		277		21	5	35	338	
		size (Mb)	%mapped	size (largest)	size (largest)	size (largest)	size	%genome
UM	mouse	2148.0	93.0%	17.9 (3.3)	6.7 (1.7)	136.8 (18.9)	2309.3	92.3 %
	human	2387.7	92.4%	32.2 (4.6)	7.6 (2.0)	157.7 (24.0)	2585.3	91.3 %
Ensembl	mouse	2274.1	94.5%	59.5 (15.1)	34.9 (21.3)	37.8 (11.5)	2406.3	96.2 %
	human	2514.2	93.9%	72.9 (17.3)	46.6 (7.2)	43.5 (12.0)	2677.2	94.5 %

* Human assembly NCBI build 33 vs. mouse assembly NCBI build 30, with the minimum segment size cut at 100kb

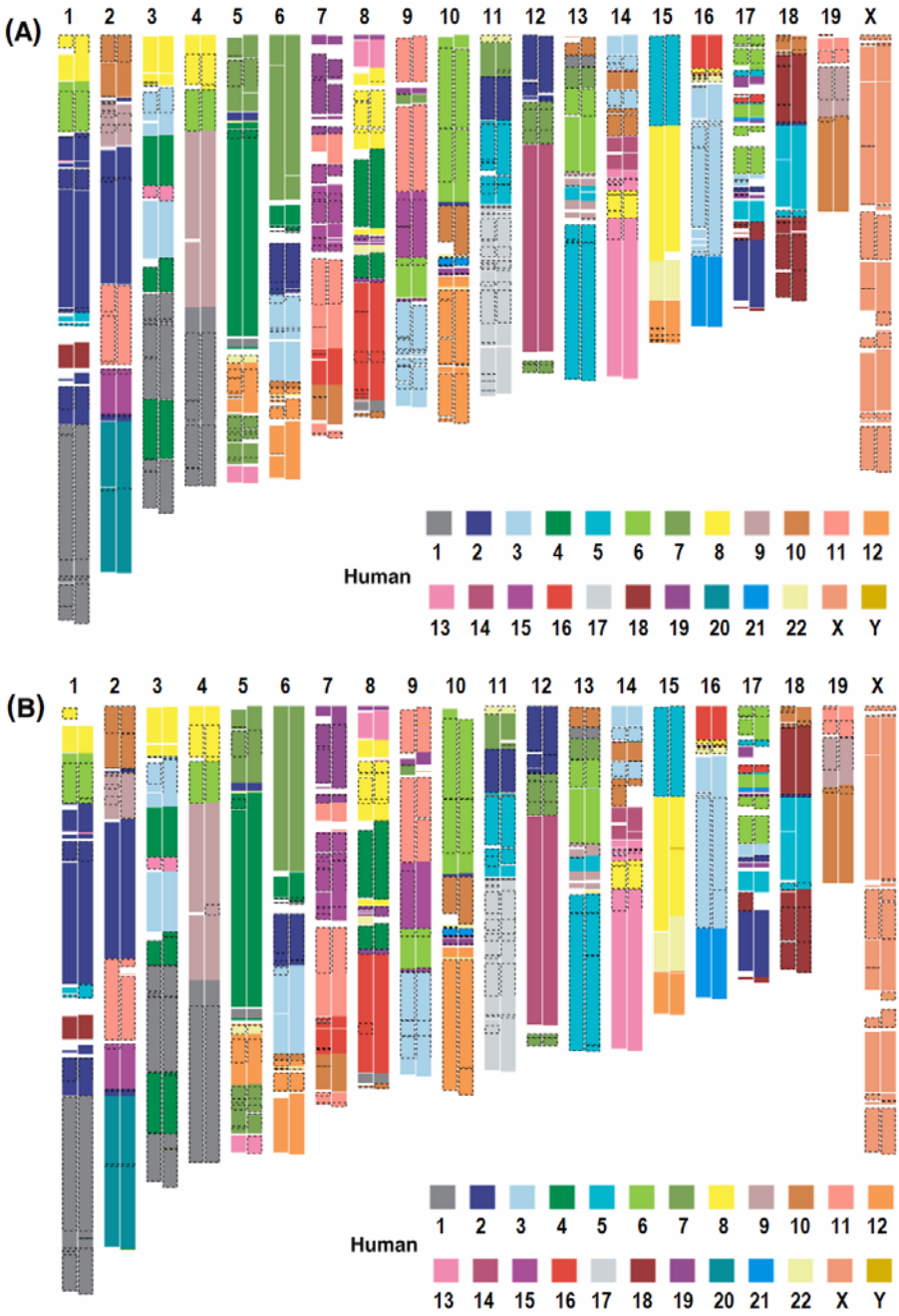


Fig. 3.7 A graphical overview of the comparisons of the human-mouse synteny maps obtained by the UM method and the corresponding map of either MGSC (A) or Ensembl (B). The UM map is shown in the left chromosomes. Each color corresponds to a particular human chromosome. Regions within a dashed box indicate that the human orthologous regions are in the backward strand.

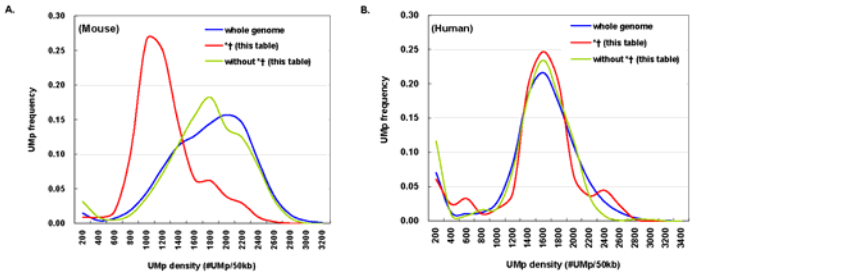
Tables 3.2 and 3.3 also show that, for all categories, the agreement between UM and MGSC was significantly better than that between UM and Ensembl. This is attributable in part to the smaller minimal conserved segments used in the Ensembl map

(100 kb vs. 300 kb for the MGSC map) and to the fact that, unlike the UM and MGSC maps, the Ensembl map is not cleanly resolved, in that some of its segments are substantially overlapping with, or entirely embedded in, other segments. The MGSC and Ensembl maps could not be precisely compared, because they were generated using different genome versions.

3.3.3 Evaluation with sequence alignment

Although a good sequence alignment, i.e., one resulting in a clear diagonal in the dot plot, does not necessary mean a pair of conserved segments are orthologous, the converse usually holds. Table 3.4 gives the results of sequence alignment, using BLASTZ (Schwartz *et al.*, 2003), for the 'Disagree' and 'Unique' segments from Tables 3.2 and 3.3. The results showed that all but 2 of the total 93 (12+71+10) UM 'Unique' or 'Disagree' pairs of segments were concordant with BLASTZ alignment, and the two exceptions were neither in the wrong orientation ("Reversed") nor without clear evidence of sequence similarity ("Unsupported"). In comparison, 2 of the 26 MGSC "Unique" and 10 of the 35 Ensembl "Unique" segment pairs were "unsupported" by BLASTZ alignment. Further examination (Figures 3.8 and 3.9) showed that 17 of the 23 MGSC "Unique", BLASTZ-concordant pairs, and 8 of the 11 Ensembl "Unique", BLASTZ-concordant pairs, were actually detected by the UM method, but were not included in the comparison because the corresponding UM segments were too small (<300 kb or <100 kb for the comparison with the MGSC or Ensembl map, respectively). These relatively small UM segments could probably be brought into agreement with the corresponding MGSC and Ensembl segments, if they were allowed to extend by sequence alignment, as discussed above. The remaining 6 (23-17) MGSC and 3 (11-8) Ensembl pairs not detected by UM were all small (most < 1 Mb), and, interestingly, the

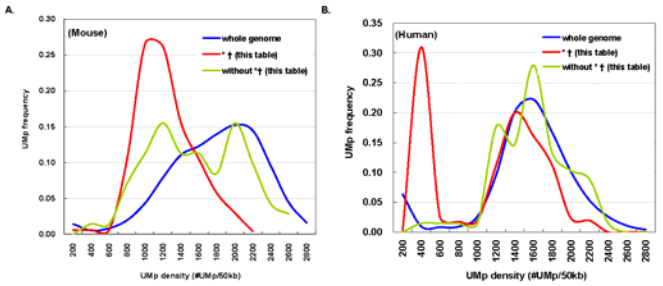
density of their UMps was significantly smaller than typical (Figures 3.8 and 3.9). We did not carry out the same evaluation on the 'Agree' segments due to limited computing resources, but, given the consensus of the results using two very different approaches (UM vs. MGSC or UM vs. Ensembl), together with the results presented below of the Largest Increasing Subsequence (LIS) analysis (Gusfield 1997) of UMps, it is unlikely that they would be BLASTZ-unsupported.



	The 23 MGSC unique-conserved segments						Cooresponding segments mapped by UM method (if available)					
	Mouse		Human		size(Mb)		Mouse		Human		size(Mb)	
1)	1	35357744 – 36678626	2 (-)	126212316 – 127968297	1.32/1.76		1	36545001 – 36685000	2 (-)	126215000 – 126395000	0.14/0.18	
2)	2	23825227 – 24310819	2 (-)	136387087 – 136954951	0.49/0.57	*						
3)	2	24448370 – 24811272	2 (+)	110941146 – 111302444	0.36/0.36		2	24545002 – 24835001	2 (+)	111015001 – 111325001	0.29/0.31	
4)	4	19523484 – 20502654	8 (-)	62369148 – 63377802	0.98/1.01	†	4	19535000 – 19575000	8 (-)	63305001 – 63385001	0.04/0.08	
5)	5	22290352 – 22645163	7 (+)	22494102 – 22892777	0.35/0.40		5	22395001 – 22635001	7 (+)	22605001 – 22875001	0.24/0.27	
6)	6	66180481 – 67067773	4 (-)	121746315 – 122955067	0.89/1.21		6	66195000 – 66455000	4 (-)	122585001 – 122935000	0.26/0.35	
7)	6	67409768 – 68020499	1 (-)	67592809 – 68276984	0.61/0.68		6	67545000 – 67825000	1 (-)	67835000 – 68135000	0.28/0.30	
8)	6	100466631 – 100773687	3 (-)	70276751 – 70604974	0.31/0.33		6	100470000 – 100765000	3 (-)	70305001 – 70615000	0.30/0.31	
9)	8	18418508 – 18753122	8 (-)	5457009 – 5806462	0.33/0.35	*						
10)	10	111646334 – 111983828	12 (-)	73209725 – 73496588	0.34/0.29		10	111595000 – 111695000	12 (-)	73425000 – 73575000	0.10/0.15	
11)	11	14707639 – 15422255	7 (-)	52755906 – 53388712	0.71/0.63	*						
12)	12	24196933 – 25154049	2 (-)	65621 – 1465094	0.96/1.40	*						
13)	14	14799816 – 15158548	14 (-)	49595453 – 49943494	0.36/0.35		14	14845001 – 15015000	14 (-)	49725000 – 49875000	0.17/0.15	
14)	15	104166368 – 104544877	12 (+)	54745979 – 55131873	0.38/0.39		15	104160000 – 104385001	12 (+)	54745001 – 55125001	0.23/0.38	
15)	17	6384013 – 6822353	6 (-)	158862622 – 159463064	0.44/0.60		17	6595001 – 6695000	6 (-)	159040001 – 159145000	0.10/0.10	
16)	17	12672381 – 14614856	6 (+)	167281001 – 170066733	1.94/2.79		17	12745001 – 12945001	6 (+)	167410001 – 167590001	0.20/0.18	
17)	18	9485148 – 9895379	18 (-)	442615 – 861485	0.41/0.42		18	9675001 – 9915000	18 (-)	445000 – 665000	0.24/0.22	
18)	18	64742635 – 65152771	18 (-)	55211604 – 55653091	0.41/0.44		18	64745000 – 64905000	18 (-)	55495000 – 55635000	0.16/0.14	
19)	X	5380273 – 5899667	X (+)	49570638 – 49925306	0.32/0.35	*						
20)	X	58782411 – 59174072	X (-)	109591823 – 109898161	0.39/0.31		X	58755001 – 58855000	X (-)	109825000 – 109925000	0.10/0.10	
21)	X	59696254 – 60647992	X (+)	7672246 – 8653245	0.95/0.98	†	X	60505001 – 60545001	X (+)	8435001 – 8485001	0.04/0.05	
22)	X	94469817 – 98701923	X (-)	84604369 – 88524592	4.23/3.92	†	X	97335000 – 97375000	X (-)	85945001 – 85985000	0.04/0.04	
23)	X	100255217 – 100722569	X (+)	83357161 – 83641557	0.47/0.28	*						

* No corresponding segments mapped by UM method
† The segment mapped by UM method is smaller than 50kb on the mouse genome

Fig. 3.8 Frequency distribution of the UMp densities for the whole genome and for regions covered by the 23 MGSC-unique BLASTZ-concordant segments (see Table 3.4). The table below shows that the UM method actually detected orthologous signals for 17 of the 23, but these were not used in the comparison because their size in the UM map was lower than 300 kb; 3 of these were very small (< 50 kb on the mouse genome; labeled †). The six segments that were not detected by the UM method are labeled *. (A) Distribution on the mouse genome. (B) Distribution on the human genome. "*†" denotes results using only the segments marked * or † in the Table, and "without *†" those using all segments apart from these.



The 11 Ensembl-unique conserved segments							Cooresponding conserved segments mapped by UM method (if available)					
	Mouse		Human		size(Mb)		Mouse		Human		size(Mb)	
1)	1	44351958 – 44664673	13 (+)	102013010 – 102389723	0.31/0.38	†	1	44560000 – 44595001	13 (+)	102235001 – 102345001	0.04/0.11	
2)	2	23598258 – 24297850	2 (-)	138749894 – 139523442	0.70/0.77	†	2	23765002 – 23785000	2 (-)	139325000 – 139365000	0.02/0.04	
3)	3	148081021 – 148316406	7 (-)	75879761 – 76128208	0.24/0.25	*						
4)	5	28081501 – 28258979	7 (+)	22334786 – 22489850	0.18/0.16	*						
5)	5	30801861 – 30921063	18 (-)	680527 – 837942	0.12/0.16	†	5	30815000 – 30855000	18 (-)	785001 – 825000	0.04/0.04	
6)	7	43488047 – 43959991	11 (+)	24552232 – 25129610	0.47/0.58	*						
7)	13	63705656 – 63822119	8 (+)	96916458 – 97044086	0.12/0.13		13	63715001 – 63775001	8 (+)	96925001 – 96985001	0.06/0.06	
8)	14	20834880 – 20935068	10 (-)	81903800 – 82093834	0.10/0.19		14	20845001 – 20895000	10 (-)	82025001 – 82085000	0.05/0.06	
9)	X	59770257 – 60256544	X (-)	112572298 – 112944771	0.49/0.37		X	59815001 – 59915000	X (-)	112845000 – 112935000	0.10/0.09	
10)	X	96741111 – 99596849	X (-)	88626799 – 91196265	2.86/2.57	†	X	99555001 – 99595000	X (-)	88710001 – 88745000	0.04/0.03	
11)	X	132498130 – 132702044	X (-)	9033068 – 9263979	0.20/0.23	†	X	1.32E+08 – 132525000	X (-)	9235000 – 9275000	0.04/0.04	

* No corresponding segments mapped by UM method
† The segment mapped by UM method is smaller than 50kb on the mouse genome

Fig. 3.9 Frequency distribution of the UMp densities for the whole genome and for regions covered by the 11 Ensembl-unique BLASTZ-concordant segments (see Table 3.4). The table below shows that the UM method actually detected orthologous signals for 8 of the 11, but these were not used in the comparison because their size in the UM map was lower than 100 kb; 5 of these were very small (< 50 kb on the mouse genome; labeled †). The three segments that were not detected by the UM method are labeled *. (A) Distribution on the mouse genome. (B) Distribution on the human genome. "*†" denotes results using only the segments marked * or † in the Table, and "without *†" those using all segments apart from these.

Table 3.4 BLASTZ-evaluation on the "Unique" and "Disagree" conserved segments from UM vs. MGSC (Table 3.2) and UM vs. Ensembl (Table 3.3) comparisons.

	Concordant *	Shifted	Multiple	Reversed	Unsupported	Total
Unique						
UM	11 (3)	0	1	0	0	12
MGSC	23 (2)	0	1	0	2	26
Unique						
UM	70 (33)	0	1	0	0	71
Ensembl	11 (1)	6	5	3	10	35
Disagree						
UM	10 (3)	0	0	0	0	10
Ensembl	0	1	4	0	0	5

* in parentheses are the number of conserved segments with size of the mouse segment ≥ 1Mb

3.3.4 Evaluation with LIS analysis of UMps

For a pair of conserved segments or anchoring islands, one expects the largest subset of UMps matched in the same direction (Figure 3.1), or LIS UMp, to be composed mainly of pUMps. An LIS analysis of UMps can, therefore, be used instead of sequence alignment to detect questionable segment or island pairs. Remarkably, the results of such an analysis (Figure 3.10) showed that, for 91% (10014/10999) of the UM anchoring islands, the LIS UMp ratio was 1.0, i.e. all the UMps matched within paired islands were ordered in the same forward or backward orientation, and only 7 (out of 10,999) pairs had a LIS UMp ratio smaller than 0.8. Furthermore, all of these 7 pairs with a low LIS UMp ratio, including two in regions full of repetitive elements, showed evidence of homology as assessed by BLASTZ alignment (Figure 3.10). As the islands were merged into segments (Methods), the percentage of ordered UMps would decrease (Figure 3.11); however, the sequence similarity of several less promising pairs,

as suggested by the LIS analysis (Figure 3.11), was validated by BLASTZ alignment (data not shown).

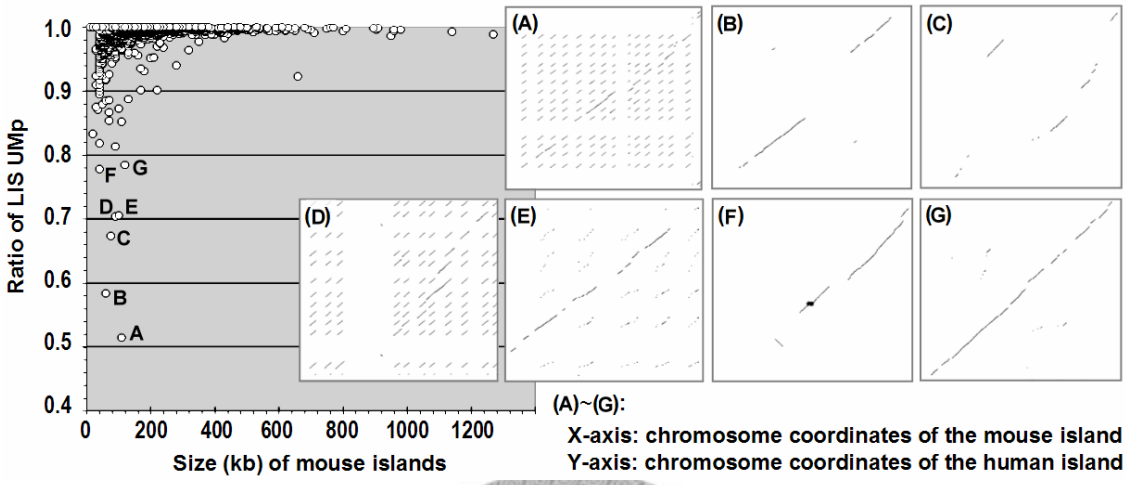


Fig. 3.10 LIS analysis of UMps in anchoring islands from the UM map of NCBI human build 33 vs. mouse build 30. (A)-(G) are dot-plots of the BLASTZ alignment for the seven indicated island pairs, for each of which the ratio of LIS UMps to all UMps common to the island pair was less than 0.8.

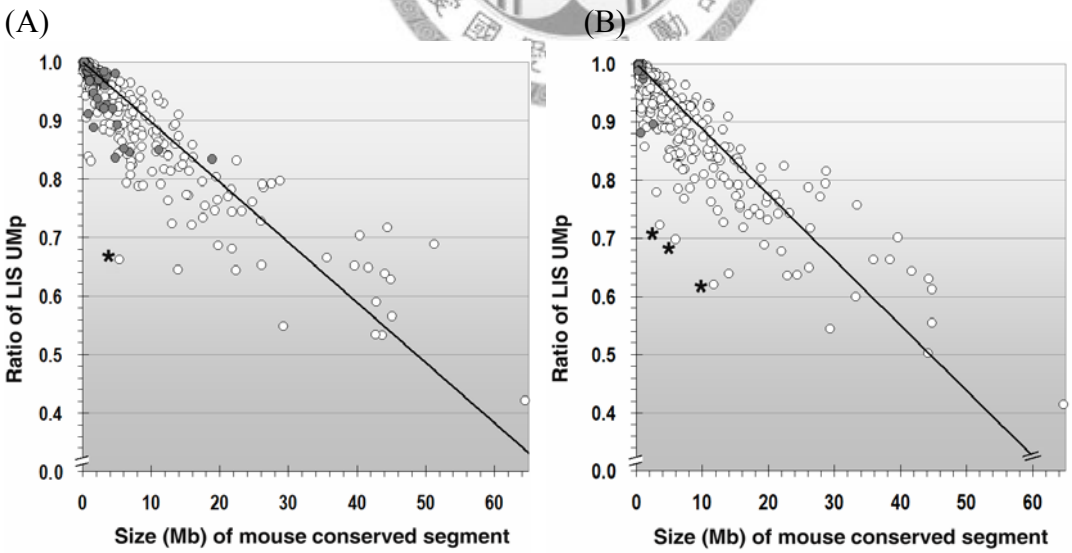


Fig. 3.11 LIS analysis of UMps in conserved segments of the UM map, using (A) mouse MGSCv3 and human NCBI build 30, and (B) NCBI mouse build 30 and human build 33. The shaded circles are segments found in the UM map, but not in the MGSC map (A) or the Ensembl map (B). Circles marked by * were evaluated by BLASTZ alignment because they had a low LIS UMp ratio for relatively small segments (see text). The line in the figure resulted from a linear regression of the data, with the constraint that it passed through a ratio of 1.0 at zero segment size.

Chapter 4

Copy number-based orthology seeding using contiguous matches

4.1 Introduction

Identifying orthologous and co-orthologous relationships between genomes is an important facet of comparative genomics (Koonin 2005). As the number of completely sequenced genomes continues to increase rapidly, orthology identification at the nucleotide level in both coding and noncoding regions of genomes is becoming an indispensable approach for studying genome evolution and for genome annotation (Deway and Pachter 2006). Essential to this approach is whole genome alignment, an approach that is computationally demanding, especially for large genomes. To achieve computational efficiency, various heuristic algorithms for large-scale sequence alignment, particularly those using index-based strategies, have been developed. Index-based alignment methods first build indices for one or all of the compared sequences and then retrieve seeds—often word matches or transformed word matches—from these indices to derive alignments to infer orthology, paralogy, and/or xenology (Fitch 2000). Since finding seeds (seeding) is necessarily the first step in all

index-based genome alignment methods (Ureta-Vidal *et al.* 2003), the strategy employed for selecting seeds and their retrieval is fundamental to the performance of genome alignment methods (Brown *et al.* 2004).

With few exceptions, most current seeding strategies are length-based, i.e., seeds are selected using fixed-length or variable-length constraints. The *k*-mer (or *k*-tuple) strategy using exact matches of words of a fixed length as seeds is perhaps the most popular and is used in general-purpose sequence comparison methods, such as FASTA (Lipman and Pearson 1985) and BLAST (Altschul *et al.* 1990), and in various genome comparison programs, which are well reviewed in Chain *et al.* (2003), Ureta-Vidal *et al.* (2003), Brown *et al.* (2004), and Batzoglou *et al.* (2005). Another length-based seeding strategy employed in genomic sequence comparison uses maximal exact matches (MEMs) (Höhl *et al.* 2002), also known as maximal pairs (Gusfield 1997), which include all exact matches with maximal lengths equal to or greater than *k* (see section 4.2.1). By excluding numerous redundant matches, which are particularly abundant in short-length words, MEM methods can acquire a better efficiency of seeding than *k*-mer methods for large-scale sequence comparison (Chain *et al.* 2003).

To further increase seeding efficiency, Delcher *et al.* (1999) proposed the use of a subset of MEMs, using only the maximal unique matches (MUMs) to align two genomes, where a MUM is a maximal substring which occurs exactly once in each of

the two compared genomes and cannot be extended without introducing mismatches. The MUMmer system works well for closely related genomes (Chain *et al.* 2003). In addition, we have presented a fixed-length seeding method, called UniMarker, with a one-to-one mapping constraint (Liao *et al.* 2004).

Generally speaking, all seeding strategies are a trade-off between sensitivity and specificity. Thus, at one extreme a typical *k*-mer method (e.g., using *k*=11, the default setting in BLAST for nucleotide comparison) can be highly sensitive, but must deal with numerous non-orthologous local matches in comparing genomes, while, at the other, most non-orthologous local matches can be automatically masked by methods such as MUMmer and UniMarker, which use a unique occurrence constraint to obtain high specificity, but suffer from limited sensitivity in detecting orthologous regions lacking the unique markers owing to sequence divergence or other evolutionary events.

Herein, we explore the possibility of devising a new seeding model that lies between these two extremes, while focusing on expanding the capability of the high-specificity methods to compare not very closely related genomes. Specifically, we generalized the seeding models of MUMs and UniMarkers by relaxing the constraint of both uniqueness and length. First, we capture all substrings of any length for which the total copy number (i.e., total number of copies in the compared genomes) is no larger than a given threshold α, and extend them to maximal length while preserving the copy

number. We call these substrings of maximal length with variable copy numbers α-markers. We then retrieve the maximal pairs, pairs of identical substrings in S_1 and S_2 that cannot be extended to longer exact matches, that contain α-markers as their substrings as seeds for orthology detection. We call these maximal pairs of α-markers α-pairs. For example, if α=4 and with x:y denoting x copies in the first genome and y copies in the second, we consider seeds of maximal length with 1:1, 1:2, 2:1, 1:3, 2:2, and 3:1 copies in the two compared genomes. Note that, in this generalization, MUMmer (Delcher *et al.* 1999) and UniMarker (Liao *et al*., 2004) both only consider 1:1 mapping and also have a constraint on word length of, respectively, $\geq k$ (k =20 is usually the default) or k=16.

In the next sections of this presentation, we first give a formal definition of α-markers and α-pairs, along with an illustrative example, then describe a linear-time algorithm to retrieve α-pairs, a prerequisite for achieving computational efficiency in genome-scale comparisons. Our algorithm is based on enhanced suffix arrays, which are efficient in comparing large genomes (Abouelhoda *et al*. 2004). Finally, we compare our seeds to several length-based seeds for their ability to detect orthologues. We use two datasets of genomes or chromosomes. The first dataset contains genomic or chromosomal sequences from human, mouse, chicken, and pufferfish and was used to compare the ability of different types of seed to detect orthologues in human versus mouse, chicken, or pufferfish. The second dataset consists of seven prokaryote genomes and was used to compare orthologues in *Mycoplasma pneumoniae* with those in another six genomes from Eubacteria and Archaebacteria. Ensembl (Hubbard *et al.* 2007) and

COG (Clusters of Orthologous Groups of proteins; Tatusov *et al.* 2003) orthologues were used to benchmark the comparisons. The results for the vertebrate dataset showed that significantly fewer seeds were required for α-pairs to achieve superior sensitivity; in addition, a denser set of colinear identical matches in these orthologues was obtained using seeding of α-pairs than using a length-based method, such as MEM or *k*-mer. Similar trends, but with less profound differences, were found in the prokaryote dataset.

4.2 Methods

In this section, we present definitions and an algorithm to compute α-pairs. We present a new algorithm based on the MEM-enumeration algorithm of Abouelhoda *et al.* (2004) which can handle the enumeration of a new type of seed.

4.2.1 α-markers and α-pairs

Suppose that S_1 and S_2 are the two genome sequences to be compared. Let $Word_c$ denote the set of all substrings such that each member of $Word_c$ has a copy number $x>0$ in S_1, $y>0$ in S_2, and $x+y = c$ ($c\geq 2$ by definition). We also denote each member in $Word_c$ as a *c*-copy word of S_1 and S_2. For example, if S_1 = "accgtttgag" and S_2 = "acccgtatgagcaccgtatgg", $Word_3$ = {"ac", "acc", "ccg", "ccgt", "cg", "cgt", "gt", "tg"}, where each word has a total of three instances in S_1 and S_2 combined. For example, word "ac" occurs three times at position 1 in S_1 and positions 1 and 13 in S_2, respectively (Figure 4.1). Our focus is on less frequent words, that is, the set union of $Word_2$, $Word_3$, …, $Word_\alpha$, where α is a user-specified integer. In order to give a compact presentation of these less frequent words, we define *α-markers* as follows.

Definition 4.1: We say a *c*-copy word of sequences S_1 and S_2 is *right maximal* if we cannot extend any of its instances in the right direction to obtain a longer *c*-copy word; likewise, a *c*-copy word is *left maximal* if we cannot extend it in the left direction to

obtain a longer *c*-copy word. We also denote a *maximal c*-copy word (a *c*-copy word that is both left and right maximal) of S_1 and S_2 as an *α-marker* of S_1 and S_2, for c = 2, 3, …, α.

In Figure 4.1, column 3 shows the maximal *c*-copy words that could be extended from the *c*-copy words in column 2 while retaining the same copy number.

(A)

Pos 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

S_1=" a c c g t t t g a g "

S_2=" a c c c g t a t g a g c a c c g t a t g g "

(B)

c	$Word_c$: (*genome,position,len*)	*α*-marker (*α*=3)	Pairs of instances {*pos1,pos2,len*}	Is an *α*-pair?
2	accg: (S_1,1,4) (S_2,13,4) accgt: (S_1,1,5) (S_2,13,5)	accgt	accgt {1,13,5}	Yes
	ag: (S_1,9,2) (S_2,10,2) ga: (S_1,8,2) (S_2,9,2) gag: (S_1,8,3) (S_2,9,3) tga: (S_1,7,3) (S_2,8,3) tgag: (S_1,7,4) (S_2,8,4)	tgag	tgag {7,8,4}	Yes
3	ac: (S_1,1,2) (S_2,1,2) (S_2,13,2) acc: (S_1,1,3) (S_2,1,3) (S_2,13,3)	acc	acc {1,1,3} acc {1,13,3}	Yes No
	ccg: (S_1,2,3) (S_2,3,3) (S_2,14,3) ccgt: (S_1,2,4) (S_2,3,4) (S_2,14,4) cg: (S_1,3,2) (S_2,4,2) (S_2,15,2) cgt: (S_1,3,3) (S_2,4,3) (S_2,15,3) gt: (S_1,4,2) (S_2,5,2) (S_2,16,2)	ccgt	ccgt {2,3,4} ccgt {2,14,4}	Yes No
	tg: (S_1,7,2) (S_2,8,2) (S_2,19,2)	tg	tg {7,8,2} tg {7,19,2}	No Yes

Fig. 4.1 An example of *α*-markers and *α*-pairs. (A) Two compared sequences S_1 and S_2 with the sequence positions indicated above the sequences. (B) For S_1 and S_2, there are seven 2-copy words ("accgt", …, "tgag") and eight 3-copy words ("ac", …, "tg"); these are listed in column 2. The round brackets in column 2 denote the positions of *c*-copy words in the form (*genome, position, len*). By extending each *c*-copy word in column 2 while preserving the same copy number, we have five maximal *c*-copy words for *c*=2,3, these being "accgt", "tgag", "acc", "ccgt", and "tg" in column 3. These are the *α*-markers for *α*=3 (Def.4.1). Column 4 shows the pairs of instances of column 3 *α*-markers in the form of {position in S_1, position in S_2, string length}. Column 5 checks whether a pair of instances in column 4 is an *α*-pair. For example, the instance pair {7,8,2} of "tg" is not an MEM and thus not an *α*-pair (Def.4.2).

In this chapter, we refer to an exact match as a pair of identical substrings in S_1 and S_2. In the following, we are interested in those matches referred to as maximal exact matches (MEMs) (Höhl *et al*. 2002) or maximal pairs (Gusfield 1997). The notation MEM of S_1 and S_2 refers to a pair of identical substrings in S_1 and S_2 that cannot be

extended to a longer exact match. Now, let us consider those MEMs of S_1 and S_2 that have α-markers as their strings. It should be emphasized that, by definition, the matches generated from an α-marker m are constrained by the copy number of m and not by their lengths.

Definition 4.2: An MEM e of sequences S_1 and S_2 is said to be an α-pair of an α-marker m of S_1 and S_2 if the two strings in e are instances of m in S_1 and S_2, respectively.

Let (*genome, position, len*) denote the position and length of a string instance in a specified genome and $\{p_1, p_2, len\}$ denote an exact match composed of string instances at (S_1, p_1, *len*) and (S_2, p_2, *len*) with string length *len*. In Figure 4.1, the three instances of "tg" are at (S_1,7,2), (S_2,8,2), and (S_2,19,2), and there are two pairs of instances of "tg" of S_1 and S_2: {7,8,2} and {7,19,2}, where only {7,19,2} is a MEM.

Lemma 4.3: Denote e as a MEM of S_1 and S_2. String ω of e is a c-copy word of S_1 and S_2 and $c \leq \alpha$ if, and only if, e is an α-pair.

Note that Lemma 4.3 follows directly from Definitions 4.1 and 4.2.

4.2.2 A linear time α-pair retrieval algorithm

According to Lemma 4.3, we can generate all α-pairs by computing the copy number c of each MEM's string and by reporting a MEM if $c \leq \alpha$. Note also that MEMs may be enumerated using the linear-time algorithm based on enhanced suffix arrays proposed by Abouelhoda *et al.* (2004). However, since the algorithm presented by Abouelhoda *et al.* deals with a single genome, we have modified it slightly to enumerate α-pairs of two genomes by borrowing the position-set technique from Höhl *et al*. (2002). Note that an enhanced suffix array refers to a data structure consisting of a suffix array (Manber and Myers 1993) and its augmented arrays, such as the longest common prefix (**lcp**) array (Kasai *et al.* 2001) and the Burrows-Wheeler transformation (**bwt**) array (Burrows and

Wheeler 1994), and these arrays can be constructed in linear time (Abouelhoda *et al.* 2004).

First, using two special symbols '#' and '$', we concatenate S_1 and S_2 into a new string $S = S_1\#S_2\$$, then build the enhanced suffix array of S as a virtual suffix tree Tv for the bottom-up traversal of all lcp-intervals (Kasai *et al.* 2001, Abouelhoda *et al.* 2004). For convenience, we say a word is a c-copy word of S if the number of times it occurs in S is exactly c , without paying any attention to its copy number in S_1 and S_2. Note that an lcp-interval σ is an interval of the suffix array that contains all suffixes of S prefixed by a right maximal c-copy word of S, say ω, for $c > 0$, where the size of σ is exactly the copy number of ω. Let n denote the size of S. Since each suffix of S is unique, there are n leaf nodes in Tv, where each leaf corresponds to a suffix of S. To simplify notations, in this paper, the notation of the right maximal c-copy word of S of σ will be referred to as the string of σ. For any two right maximal words of S, ω_1 and ω_2, we define the partial order relation $\omega_1 < \omega_2$ if ω_1 is a prefix of ω_2. In Tv, each node corresponds to an lcp-interval and a node σ_1 is an ancestor of a node σ_2 if, and only if, their corresponding right maximal words of S satisfy the partial order relation $\omega_1 < \omega_2$. Obviously, the string ω of σ is the longest common prefix of its children.

In the following, we present the algorithm to enumerate all α-pairs of genomes S_1 and S_2.

Let Σ be the set of letters of S_1 and S_2 and let $|\Sigma|$ denote the size of Σ. During the bottom-up traversal of Tv, for each node, σ, we maintain $2\times|\Sigma|$ position sets $P(g,x,\sigma)$, where g=1 or 2 and $x\in\Sigma$. Each element of $P(g,x,\sigma)$ is a suffix of S prefixed by the string ω of σ in genome S_g and x is the character immediately to the left of this suffix. Note that only the starting position of each suffix is recorded in the position set $P(g,x,\sigma)$.

Depending on whether σ is a leaf node or an internal node, the computation of the position sets is different. For each leaf node of *Tv*, the size of its lcp-interval is one and its $P(g,x,\sigma)$ is obtained simply by looking up the **bwt** array. For each internal node σ of *Tv*, $P(g,x,\sigma)$ is the set union of its children's position sets. This position-set technique is adopted from Kurtz and Lonardi (2004), who showed that the position sets can be computed in $O(|\Sigma|n)$ time.

For each internal node σ of *Tv*, if we let ω be its string and let σ_a and σ_b be any two distinct children of σ, we obtain starting positions of MEMs of S_1 and S_2 with string ω by computing $P(1,x,\sigma_a) \times P(2,y,\sigma_b)$ for all $x \neq y$ and for all σ's children $\sigma_a \neq \sigma_b$. Note that it is not difficult to show that, for each MEM *e* reported at node σ of *Tv*, the string of *e* is exactly the string of σ (Kurtz and Lonardi 2004, Abouelhoda *et al.* 2004), and thus each MEM enumerated in this procedure is unique. It is also known that the above MEM-enumeration algorithm produces all MEMs of S_1 and S_2 (Kurtz and Lonardi 2004, Abouelhoda *et al.* 2004), and we have modified it by performing the MEM-reporting procedure at node σ of *Tv* if the size of σ is no greater than α. According to Lemma 3, each produced MEM is an α-pair. Since each α-pair can be enumerated in constant time, the entire enumeration procedure runs in $O(|\Sigma|n + z)$ time, where *z* is the number of α-pairs.

From the above, we know that α-pairs can be retrieved by first generating position sets of each node of *Tv* in $O(|\Sigma|n)$ time, then enumerating α-pairs by traversing *Tv* in another $O(|\Sigma|n + z)$ time. Thus, the total complexity of the α-pair retrieval algorithm runs in $O(|\Sigma|n + z)$ time. Additionally, theorem 4 below ensures the completeness of the α-pair retrieval algorithm, while the proof of the soundness is trivial.

Theorem 4: Given two sequences S_1 and S_2, the α-pair retrieval algorithm reports every α-pair.

Proof: Let us assume the contrary, i.e., there exists an α-pair p which is not reported by the α-pair retrieval algorithm. Denote u_1 and u_2 as the two suffixes prefixed by the two string instances of p, with u_1 belonging to genome S_1 and u_2 to S_2. In other words, p is the longest common prefix of u_1 and u_2. Let us also denote the two nodes in the virtual suffix tree containing u_1 and u_2 as σ_1 and σ_2, respectively. We also denote the character immediately to the left of u_1 as a_1 and the one immediately to the left of u_2 as a_2. Let us consider the marker ω of the closest common ancestor σ of σ_1 and σ_2. We can show that the strings of ω and p are the same (if we assume otherwise, then there must exist another common ancestor of σ_1 and σ_2 and this contradicts the fact that σ is the closest common ancestor of σ_1 and σ_2. The details can be easily derived by interested readers). Let σ_1' and σ_2' denote the two children of σ which are ancestors of σ_1 and σ_2, respectively. Then we can see that σ_1 belongs to $P(1,a_1,\sigma_1')$ and σ_2 belongs to $P(2,a_2,\sigma_2')$. Since p is not reported by the algorithm, thus the copy number of ω must be greater than α. However, as we mentioned earlier, the strings of ω and p are the same, which contradicts the assumption that p is an α-pair.

4.2.3 Evaluation of orthology seeding

To evaluate the ability of different types of seeds to detect orthologues, we used two quantitative measures: seeding sensitivity and colinear identities per orthologue. Seeding sensitivity is defined as

$$Sn = 100\% \times N_{\text{Seeded}} / N ,$$

where N denotes the total number of orthologues annotated in a reference benchmark,

such as the Ensembl orthology (Hubbard *et al.* 2007) or COG (Tatusov *et al.* 2003), and N_{Seeded} denotes the total number of annotated orthologues containing at least one seed. Obviously, it is impossible to detect an orthologue if no seeds are found within the orthologue. Furthermore, because the sensitivity measure *Sn* does not gauge how likely the seeded orthologues will be detected, we define a second measure, the colinear identities per orthologue, as

$$\bar{I}c = \sum_{i=1}^{N} I_i \Big/ N ,$$

where I_i denotes the maximal number of colinearly identical base pairs decomposed from the seeds mapping the two sequences of the *i*-th orthologue. I_i can be computed using an algorithm for finding longest increasing subsequences (Gusfield 1997).

* Steps for computing colinear identities

1 Collect seeds that fall inside the *i*-th orthologue as a set ***X***.

2 For each seed $\{p_1, p_2, \ell\}$[a] in ***X***:

  2.1 Decompose it into ℓ letter matches: $\{p_1, p_2, 1\}$, $\{p_1+1, p_2+1, 1\}$, …, $\{p_1+\ell-1, p_2+\ell-1, 1\}$.

  2.2 Store the letter matches (i.e., identical base pairs) to an array ***Y***.

3 Sort ***Y*** by ascending order of the positions in S_1 and descending order of the positions in S_2.

4 For each record in ***Y***, store the positions in S_2 to an integer sequence ***Z***.

5 Compute I_i = the length of LIS[b] (Longest Increasing Subsequence) of ***Z***.

[a] A seed is a substring match in the form {position p_1 in S_1, position p_2 in S_2, string length ℓ}.

[b] We implemented an O(n log p) LIS algorithm (Gusfield 1997), where n is the length of the input and p the length of the LIS.

Generally speaking, the larger *I* for a candidate orthologue, the easier it is to

identify the orthologue in a post-seeding process such as the ungapped/gapped extension (Altschul *et al.* 1997). Using the Ensembl- and COG-annotated orthologues for a variety of vertebrate and prokaryote species as benchmarks, we computed *Sn* and $\bar{I}c$ for several different seeding models to compare, respectively, their sensitivity to seed the annotated orthologues and their relative potential to detect the annotated orthologues.

4.2.4 Datasets and software

Two datasets of genomic sequences were used to evaluate the different seeding models. Dataset A (Table 4.1A) consisted of human chromosome 15, mouse chromosome 7, chicken chromosome 10, and the freshwater pufferfish genome; the orthologues between human and the various species as annotated by Ensembl (Hubbard *et al.* 2007) were used as the reference answer-set for evaluation. These vertebrate genomic sequences were retrieved from ftp://ftp.ensembl.org/pub/release-41/, and the orthologues as annotated in Ensembl v.41 were obtained by querying BioMart at http://oct2006.archive.ense-mbl.org/Multi/martview. Dataset B (Table 4.1B) consisted of seven small prokaryote genomes, and their COG orthologues (Tatusov *et al.* 2003) were used as the reference answer-set. For dataset B, we retrieved genomes from ftp://ftp.ncbi.nih.gov/genomes/Bacteria/ and COG orthologues from ftp://ftp.ncbi.nih.gov/pub/COG/COG/. Both the Ensembl and COG orthologues are determined based on protein sequence comparisons. Computer modules for our method were written in C/C++ and are freely downloadable from the website http://synteny.iis.sinica.edu.tw/am/ .

Table 4.1 The two datasets used in this study.

(A) Vertebrate dataset (reference sequence: HS chr.15, 100Mb)

Sequence compared[a,b] ID name (size)	Divergence time[c]	#Orthologues[d] (1-to-1, *m*-to-*m*)
MM chr.7 (145 Mb)	91 Mya	124 (124, 0)
GG chr.10 (21 Mb)	310 Mya	314 (296, 18)
TN genome (217 Mb)	450 Mya	347 (245, 102)

(B) Prokaryote dataset (reference sequence: Mpn genome, 816Kb)

Sequence compared[a,e] ID name (size)	Divergence time[c]	#Orthologues[d] (1-to-1, *m*-to-*m*)
Mge genome (580 Kb)	< 2600 Mya	472 (319, 153)
Rpr genome (1112 Kb)	~ 2600 Mya	315 (191, 124)
Buc genome (641 Kb)	~ 2600 Mya	274 (209, 65)
Bbu genome (911 Kb)	~ 2600 Mya	333 (227, 106)
Ctr genome (1043 Kb)	~ 2600 Mya	297 (200, 97)
Tac genome (1565 Kb)	> 4000 Mya	320 (120, 200)

[a] HS: *Homo sapiens*; MM: *Mus musculus*; GG: *Gallus gallus*; TN: *Tetraodon nigroviridis*. Mpn: *Mycoplasma pneumoniae*; Mge: *Mycoplasma genitalium*; Rpr: *Rickettsia prowazekii*; Buc: *Buchnera sp. APS*; Bbu: *Borrelia burgdorferi*; Ctr: *Chlamydia trachomatis*; Tac: *Thermoplasma acidophilum*. HS chromosome 15 and the Mpn genome were used as the reference sequence for the comparisons in dataset A and B, respectively. The figures in parentheses are the size of the chromosomes or genomes compared.

[b] MM chr.7 was chosen because it has a larger number of, compared to the other chromosomes, orthologues to HS chr.15. GG chr.10 was chosen by the same criterion.

[c] Mya denotes millions of years ago. Data are from Hedges (2002).

[d] The orthologues (i.e., orthologous gene pairs) in (A) are from Ensembl v.41 and those in (B) from the COG database (see Methods). *m*-to-*m* denotes many-to-many relationships, which include one-to-many and many-to-one mappings.

[e] Mpn was used as the query genome to compare against five genomes from four phyla of Eubacteria (Firmcutes: Mpn, Mge; Proteobacteria: Rpr, Buc; Spirochaetes: Bbu; Chlamydiales: Ctra) and one genome from Euryarchaeota of Archaebacteria (Taci).

4.3 Results

4.3.1 α-pairs vs. MEM or *k*-mer in vertebrate sequences

There are two common ways to parameterize exact matches of DNA sequences based on sequence length: 1) a *k*-mer pair of S_1 and S_2 is a pair of identical words in S_1 and S_2, where the length of the words is *k*, and 2) a maximal exact match (MEM) (Höhl *et al*. 2002) of S_1 and S_2 (see Methods 2.1). We let MEM_k denote the set of all MEMs for which the lengths are equal to, or greater than, *k*. For convenience, we denote the set of α-pairs at a specified α value as AP_α. We were interested in knowing whether the copy-number-based AP_α seeds conferred any advantage over the length-based MEM_k

and *k*-mer seeds in detecting orthologues, and how their performances changed when the parameters α and *k* changed. Using dataset A (Table 4.1A) and the two measures *Sn* and $\bar{I}c$ described above, we compared the orthology seeding results for human chr.15 versus mouse chr.7, chicken chr.10, and the pufferfish genome using seeds of AP_α, MEM_k, or *k*-mer pairs. The results are presented in Figures 4.2 and 4.3, and Table 4.2.

Figure 4.2 shows that, for the two mammals human and mouse, a minimal α (α=2) was sufficient for AP_α to seed all the orthologues of HS chr.15 vs MM chr.7 (Figure 4.2A), and, for the more distant pair human and chicken, 99% of the orthologues of HS chr.15 vs GG chr.10 could still be seeded by α=3, but, to cover the last 1% (3 orthologues, see Table 4.1A), the cost, i.e. size of α and total number of seeds, escalated (Figure 4.2C). In comparison, a much larger, but still small, α (~10) was needed to seed 90% of the very distant human-fish orthologues (Figure 4.2E), while, to cover the last orthologue, α increased to more than a thousand (Figure 4.3A). Similarly, as the evolutionary distance from human increased on going from mouse to chicken to fish, the difficulty in actually mapping these orthologues increased accordingly, as evidenced by the decreasing $\bar{I}c$, which decreased from thousands to hundreds to scores for these species at relatively low α copies (Figures 4.2B, D, and F).

The seeding results showed that the data for the AP_α seeds were all much closer to the upper left corner of the plot than those for the MEM_k or *k*-mer seeds (Figure 4.2A-F), indicating a superior performance for AP_α in both the *Sn* and $\bar{I}c$ measures. That is, using the same amount of seeds, AP_α achieved a better *Sn* and $\bar{I}c$ than MEM_k or *k*-mer; conversely, to achieve the same *Sn* or $\bar{I}c$, a much larger number of seeds were required for MEM_k or *k*-mer (especially the latter) than for AP_α. Quantitatively, depending on the species compared, to achieve 100% *Sn,* between 2 and 62 times as many seeds were

required for MEM_k than for AP_α, while the *k*-mer/AP_α seed ratio was between 5 and 377. To achieve a nearly equal $\bar{I}c$, the $MEM_{k/}$ AP_α seed ratio ranged from ~10 to ~30 and the *k*-mer/ AP_α seed ratio ranged from 28 to 159 (for k =14, 15, and 16; Table 4.2). The values of these ratios appear to depend on the values of α and k, the evolutionary distance, and the sizes of the sequences compared (for example, pufferfish is much more distant from human than is chicken, but the size of its whole genome sequence (217 Mb) is much larger that that of chicken chromosome 10 (21 Mb) (Table 4.1A). Other factors, such as the number of times a genome had been wholly or segmentally duplicated, might also have an effect.

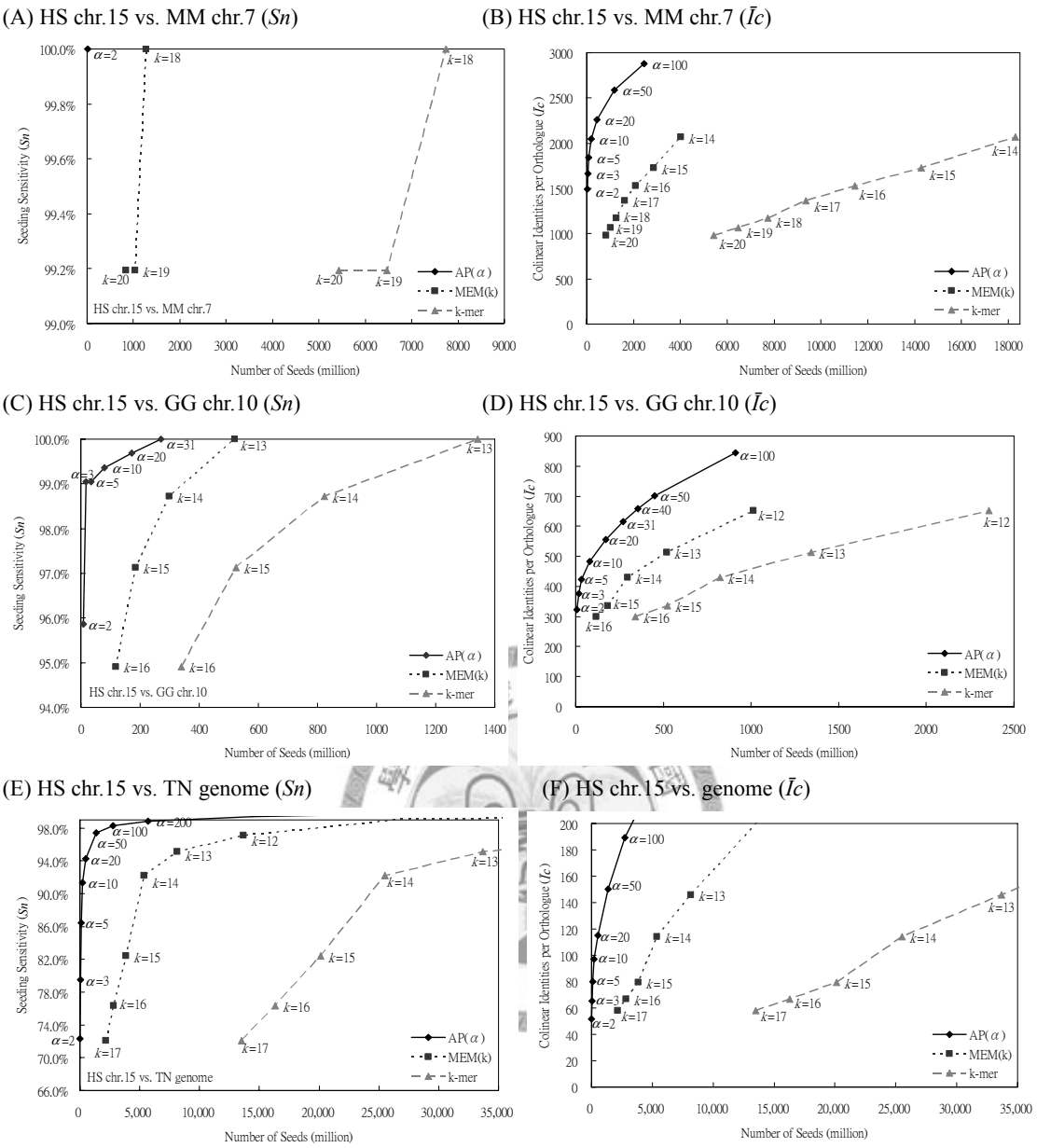


Fig. 4.2 *Sn* or $\bar{I}c$ vs. total number of seeds generated using AP_α, MEM_k, or *k*-mer in the comparison of vertebrate sequences (Table 4.1A). (A) and (B) are the results for human chr.15 vs. mouse chr.7, (C) and (D) are the results for human chr.15 vs. chicken chr.10, and (E) and (F) are the results for human chr.15 vs. the pufferfish genome. The data for the larger α and smaller *k* values needed to reach 100% *Sn* or a higher $\bar{I}c$ for the human vs. pufferfish comparison are presented in Figure 4.3.

(A) HS chr.15 vs TN genome: *Sn* (B) HS chr.15 vs TN genome: $\bar{I}c$

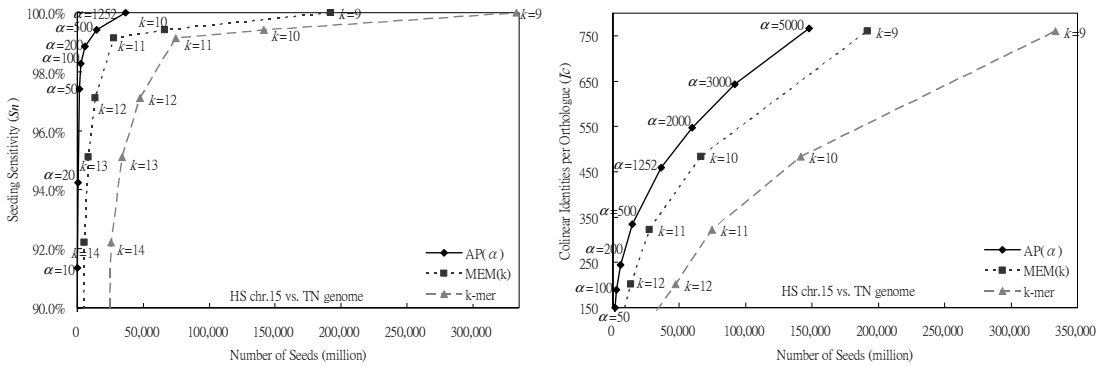


Fig. 4.3 (A) and (B) are, respectively, the extension of Figures 4.2E and 4.2F for larger α and smaller k values.

Table 4.2 MEM_k or k-mer seed to AP_α seed ratio at 100% *Sn* or an nearly equal $\bar{I}c$ for detecting vertebrate orthologues (dataset A).

(i) *Sn*

Sequences compared	Parameters and seed# generated at 100% *Sn*			**A**. Seed#(k-mer)/Seed#(AP_α) **B**. Seed#(MEM_k)/Seed#(AP_α)
	k-mer	MEM_k	AP_α	
HS chr.15, MM chr.7	**k** = 18 Seed# = 7,734,559,863	**k** = 18 Seed# = 1,270,153,339	**α**= 2 Seed# = 20,503,556	**A= 377.23** **B= 61.95**
HS chr.15, GG chr.10	**k** = 13 Seed# = 1,342,581,347	**k** = 13 Seed# = 519,637,186	**α**= 31 Seed# = 272,011,977	**A= 4.94** **B= 1.91**
HS chr.15, TN genome	**k** = 9 Seed# = 333,160,672,645	**k** = 9 Seed# = 191,831,428,705	**α**= 1252 Seed# = 36,522,353,493	**A= 9.12** **B= 5.25**

(ii) $\bar{I}c$

(A) HS chr.15 vs. MM chr.7

k	Seed#(k-mer)	$\bar{I}c$	k	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(k-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
16	11,441,794,971	1527.04	16	2,091,308,171	1527.04	3	42,223,038	1663.10	270.98	49.53
15	14,291,762,410	1724.56	15	2,849,967,439	1724.56	5	87,128,594	1843.71	164.03	32.71
14	18,307,564,831	2069.69	14	4,015,802,421	2069.69	20	443,129,366	2259.95	41.31	9.06
Average for k=14~16									158.78	30.43

(B) HS chr.15 vs. GG chr.10

k	Seed#(k-mer)	$\bar{I}c$	k	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(k-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
16	340,355,453	299.02	16	117,602,462	299.02	2	8,079,214	322.98	42.13	14.56
15	524,956,370	334.97	15	184,600,917	334.97	3	16,868,192	377.30	31.12	10.94
14	822,944,161	430.32	14	297,987,791	430.32	10	80,059,161	483.21	10.28	3.72
Average for k=14~16									27.84	9.74

(C) HS chr.15 vs. TN genome

k	Seed#(k-mer)	$\bar{I}c$	k	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(k-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
16	16,307,700,263	66.73	16	2,821,865,633	66.73	5	101,623,422	80.08	160.47	27.77
15	20,139,059,134	79.75	15	3,831,358,871	79.75	10	235,753,891	97.16	85.42	16.25
14	25,526,195,868	114.09	14	5,387,136,734	114.09	20	509,153,134	115.14	50.13	10.58
Average for k=14~16									98.68	18.20

(iii) Summary

Sequences compared	Achieving *Sn*=100%[a]			Achieving equal $\bar{I}c$[b]		
	k-mer	MEM_k	AP_α	k-mer	MEM_k	AP_α
HS chr.15, MM chr.7	377.2 :	62.0 :	1	158.8 :	30.4 :	1
HS chr.15, GG chr.10	4.9 :	1.9 :	1	27.8 :	9.7 :	1
HS chr.15, TN genome	9.1 :	5.3 :	1	98.7 :	18.2 :	1

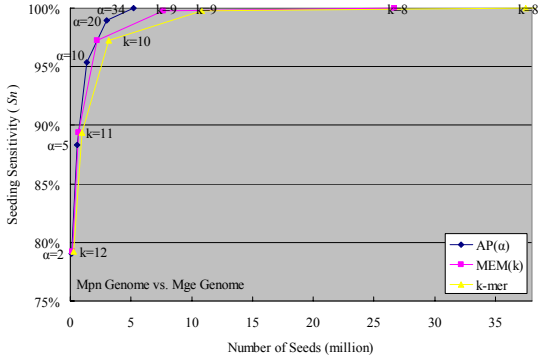
[a] Details of the data are provided in the above table (i)

[b] Ratios for $\bar{I}c$ were estimated using MEM_{14}, MEM_{15}, and MEM_{16} (see Figure 2 and the above table (ii)).

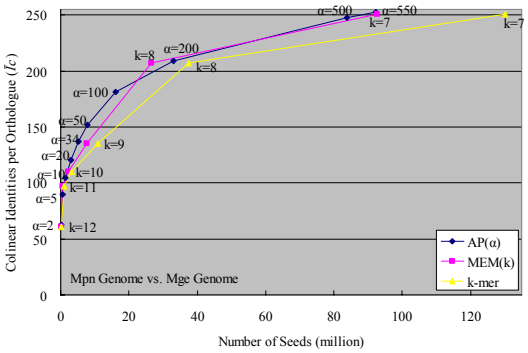
4.3.2 α-pairs vs. MEM or *k*-mer in prokaryote sequences

The marked difference between AP_α and MEM_k or *k*-mer seeds seen for the vertebrate orthologues above was much reduced when comparing prokaryote genomes. To achieve an *Sn* of 100% and a nearly equal $\bar{I}c$, the MEM_k/AP_α and *k*-mer/AP_α (for k =7, 8, and 9) seed ratios were often just slightly above unity and all were less than 10 (Table 4.3). The small $\bar{I}c$, usually of the order of 10-100 (see Figure 4.4), also indicated a difficulty for all the three seed models in mapping these distant prokaryote orthologues, and that increasing copy numbers (α) would not result in as great an advantage over MEM_k and *k*-mer as in the case of vertebrate orthologues (cf. Figures 4.2 and 4.4). Because these prokaryote genomes are 10 to several hundreds times smaller than the vertebrate sequences compared (Table 4.1), the results may suggest that, while AP_α seeds are more efficient than MEM_k and *k*-mer seeds in detecting both vertebrate and prokaryote orthologues, this efficiency gain is considerably greater in larger-scale comparisons.

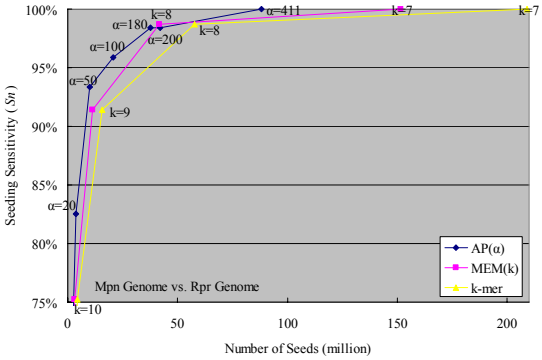
(A) Mpn vs. Mge: *Sn*



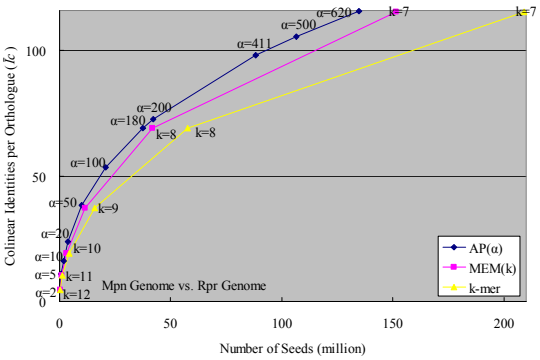
(B) Mpn vs. Mge: *Īc*



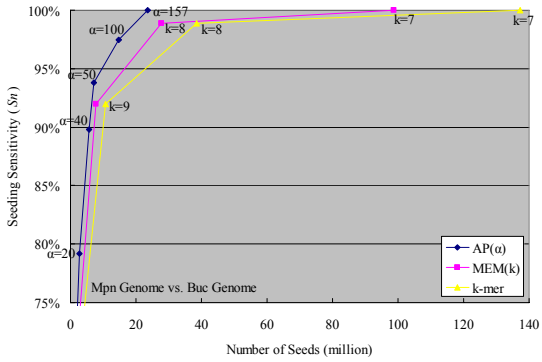
(C) Mpn vs. Rpr: *Sn*



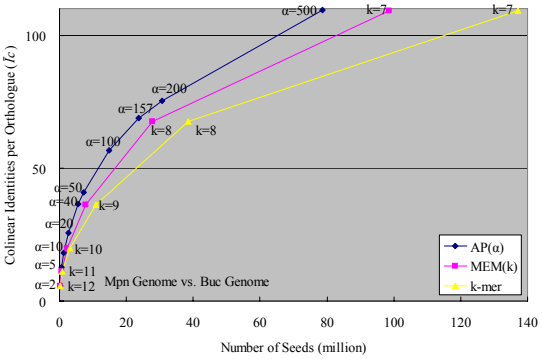
(D) Mpn vs. Rpr: *Īc*



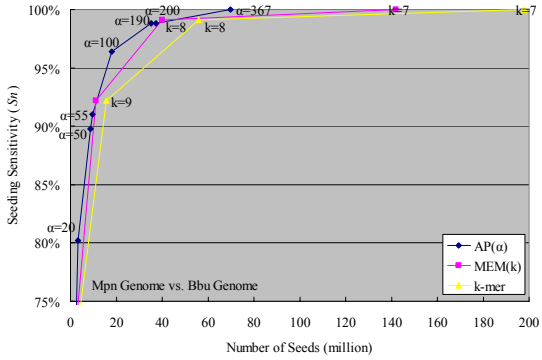
(E) Mpn vs. Buc: *Sn*



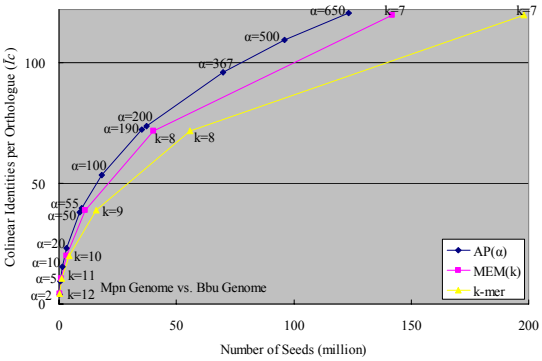
(F) Mpn vs. Buc: *Īc*



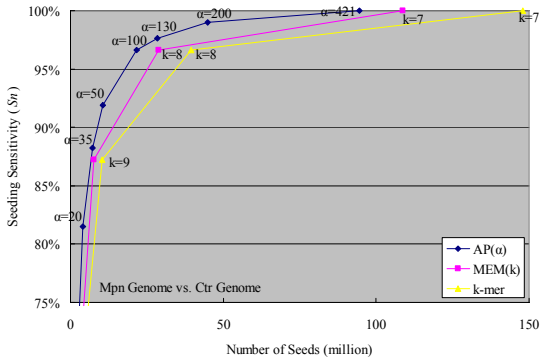
(G) Mpn vs. Bbu: *Sn*



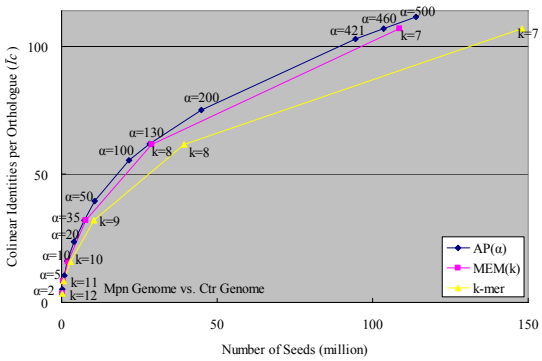
(H) Mpn vs. Bbu: $\bar{I}c$



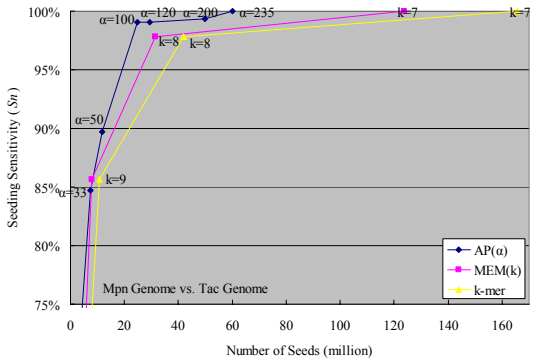
(I) Mpn vs. Ctr: *Sn*



(J) Mpn vs. Ctr: $\bar{I}c$



(K) Mpn vs. Tac: *Sn*



(L) Mpn vs. Tac: $\bar{I}c$

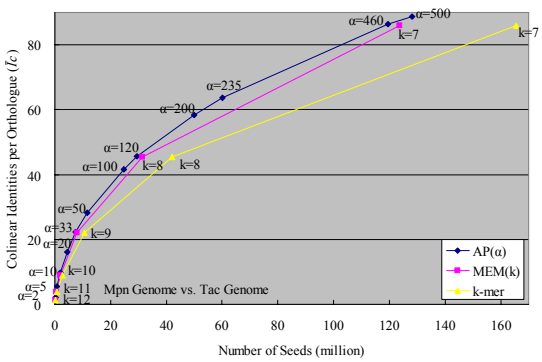


Fig. 4.4 *Sn* or $\bar{I}c$ vs. total number of seeds generated using AP_a, MEM_k, or *k*-mer in the comparison of prokaryote genomes (dataset B).

Table 4.3 MEM_k or *k*-mer seed to AP_α seed ratio at 100% *Sn* or an nearly equal $\bar{I}c$ for detecting prokaryote orthologues (dataset B).

(i) *Sn*

Sequences compared	Parameters and seed# generated at 100% *Sn*			**A**. Seed#(*k*-mer)/Seed#(AP_α) **B**. Seed#(MEM_k)/Seed#(AP_α)
	k-mer	MEM_k	AP_α	
Mpn, Mge genomes	***k*** = 8 Seed# = 37,465,508	***k*** = 8 Seed# = 26,656,835	***α***= 34 Seed# = 5,232,900	**A= 7.16** **B= 5.09**
Mpn, Rpr genomes	***k*** = 7 Seed# = 209,202,042	***k*** = 7 Seed# = 151,581,922	***α***= 411 Seed# = 88,221,457	**A= 2.37** **B= 1.72**
Mpn, Buc genomes	***k*** = 7 Seed# = 137,217,332	***k*** = 7 Seed# = 98,641,496	***α***= 157 Seed# = 23,652,853	**A= 5.80** **B= 4.17**
Mpn, Bbu genomes	***k*** = 7 Seed# = 197,838,930	***k*** = 7 Seed# = 141,975,543	***α***= 367 Seed# = 69,785,157	**A= 2.83** **B= 2.03**
Mpn, Ctr genomes	***k*** = 7 Seed# = 148,041,782	***k*** = 7 Seed# = 108,666,713	***α***= 420 Seed# = 94,306,140	**A= 1.57** **B= 1.15**
Mpn, Tac genomes	***k*** = 7 Seed# = 165,652,229	***k*** = 7 Seed# = 123,668,040	***α***= 235 Seed# = 60,013,345	**A= 2.76** **B= 2.06**

(ii) $\bar{I}c$

(A) Mpn genome vs. Mge genome

k	Seed#(*k*-mer)	$\bar{I}c$	*k*	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(*k*-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
9	10,808,673	135.45	9	7,646,186	135.45	34	5,232,900	136.98	2.07	1.46
8	37,465,508	206.84	8	26,656,835	206.84	200	32,954,863	208.80	1.14	0.81
7	130,118,311	250.46	7	92,652,803	250.46	550	92,234,567	252.25	1.41	1.00
Average for *k*=7-9									1.54	1.09

(B) Mpn genome vs. Rpr genome

k	Seed#(*k*-mer)	$\bar{I}c$	*k*	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(*k*-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
9	15,850,543	37.35	9	11,439,656	37.35	50	9,947,265	38.47	1.59	1.15
8	57,620,120	69.24	8	41,769,577	69.24	180	37,606,998	69.22	1.53	1.11
7	209,202,042	115.50	7	151,581,922	115.50	620	134,718,171	115.60	1.55	1.13
Average for *k*=7-9									1.56	1.13

(C) Mpn genome vs. Buc genome

k	Seed#(*k*-mer)	$\bar{I}c$	*k*	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(*k*-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
9	10,830,804	36.15	9	7,743,848	36.15	40	5,620,685	36.43	1.93	1.38
8	38,575,836	67.50	8	27,745,032	67.50	157	23,652,853	68.94	1.63	1.17
7	137,217,332	109.15	7	98,641,496	109.15	500	78,596,947	109.36	1.75	1.26
Average for *k*=7-9									1.77	1.27

(D) Mpn genome vs. Ctr genome

k	Seed#(*k*-mer)	$\bar{I}c$	*k*	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(*k*-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
9	15,755,379	38.73	9	11,262,885	38.73	55	9,716,666	39.70	1.62	1.16
8	55,863,387	71.76	8	40,108,008	71.76	190	35,263,107	72.23	1.58	1.14
7	197,838,930	119.85	7	141,975,543	119.85	650	123,433,031	120.60	1.60	1.15
Average for *k*=7-9									1.60	1.15

(E) Mpn genome vs. Bbu genome

k	Seed#(*k*-mer)	$\bar{I}c$	*k*	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(*k*-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
9	10,476,247	31.95	9	7,674,858	31.95	35	7,257,329	32.12	1.44	1.06
8	39,375,069	61.74	8	28,898,822	61.74	130	28,371,621	61.93	1.39	1.02
7	148,041,782	106.89	7	108,666,713	106.89	460	103,489,389	106.91	1.43	1.05
Average for *k*=7-9									1.42	1.04

(F) Mpn genome vs. Tac genome

k	Seed#(*k*-mer)	$\bar{I}c$	*k*	Seed#(MEM_k)	$\bar{I}c$	α	Seed#(AP_α)	$\bar{I}c$	Seed#(*k*-mer)/Seed#(AP_α)	Seed#(MEM_k)/Seed#(AP_α)
9	10,658,274	22.00	9	7,931,466	22.00	33	7,531,808	22.19	1.42	1.05
8	41,984,189	45.40	8	31,325,915	45.40	120	29,473,289	45.69	1.42	1.06
7	165,652,229	85.93	7	123,668,040	85.93	460	119,581,616	86.43	1.39	1.03
Average for *k*=7-9									1.41	1.05

(iii) Summary

Sequences compared	Achieving *Sn*=100%[a] *k*-mer : MEM_k : AP_α	Achieving equal $\bar{I}c$ [b] *k*-mer : MEM_k : AP_α
Mpn genome, Mge genome	7.16 : 5.09 : 1	1.54 : 1.09 : 1
Mpn genome, Rpr genome	2.37 : 1.72 : 1	1.56 : 1.13 : 1
Mpn genome, Buc genome	5.80 : 4.17 : 1	1.77 : 1.27 : 1
Mpn genome, Bbu genome	2.83 : 2.03 : 1	1.60 : 1.15 : 1
Mpn genome, Ctr genome	1.57 : 1.15 : 1	1.42 : 1.04 : 1
Mpn genome, Tac genome	2.76 : 2.06 : 1	1.41 : 1.05 : 1

[a] Details of the data are provided in the above table (i).

[b] Ratios for $\bar{I}c$ were estimated using MEM_7, MEM_8, and MEM_9 (see Figure 4.4 and the above table (ii)).

4.3.3 α-pairs vs. MUM or MAM

As alluded to earlier, α-pairs can be considered as a conceptual extension of MUMs with the removal of the uniqueness constraint. It was therefore of interest to compare AP_α with MUM_k and with MAM_k (the maximal almost-unique match), which extends the one-to-one mapping of MUM_k to one-to-many mapping (i.e., one-side uniqueness, Delcher *et al.* 2002). Note that, besides one-side or two-side uniqueness, both MUM_k and MAM_k, like MEM_k, also impose a length constraint (k).

Table 4.4 shows the best *Sn* for the two datasets (Table 4.1) that could be achieved with MUM_k and MAM_k, i.e., with MUM_1 and MAM_1 where matches of all lengths (i.e. $k \geq 1$) were considered. The results showed that, as the evolutionary distance from human increased from mouse to chicken to pufferfish, the best possible *Sn* for MUM_k decreased, and for the highly diverged bacteria genomes, it dropped to below 40%. Removing the unique mapping on one side, as in MAM_1, resulted in a considerable improvement in *Sn*, but not to an extent that would be useful in practice, especially for distant genomes. In comparison, AP_α achieved a similar *Sn* and $\bar{I}c$ to MAM_1 (Table 4.5) at a very small α (3-5), and, unlike MUM_k and MAM_k, could reach 100% *Sn* with a moderate or manageable α (see below for the scalability of α-pairs), even for prokaryote species that have diverged for more than 4000 million years (Table 4.1B). Thus, by tuning copy number instead of length, the potential to map moderately, or even very, distant genomes seems much greater with α-pairs than with length-based seeds, although it remains to be determined what $\bar{I}c$ value would be large enough to map highly distant orthologues in the post-seeding processes without the aid of protein sequence comparisons.

Table 4.4 From MUM/MAM to α-pairs: improving sensitivity by increasing α.

Sequences compared	$Sn(MUM_1)$[a] (=$Sn(AP_2)$)	$Sn(MAM_1)$[b]	$Sn, \alpha\ (AP_\alpha)$[c]	
			~$Sn(MAM_1)$	Sn=100%
HS chr.15, MM chr.7	100.0%	100.0%	100.0%, α=2	α=2
HS chr.15, GG chr.10	95.9%	98.1%	99.0%, α=3	α=31
HS chr.15, TN genome	72.3%	81.6%	83.9%, α=4	α=1252
Mpn, Mge genomes	79.0%	87.5%	88.4%, α=5	α=34
Mpn, Rpr genomes	25.1%	35.9%	37.8%, α=3	α=411
Mpn, Buc genomes	33.2%	48.5%	49.3%, α=4	α=157
Mpn, Bbu genomes	25.5%	33.0%	33.9%, α=3	α=367
Mpn, Ctr genomes	29.0%	36.4%	39.1%, α=3	α=420
Mpn, Tac genomes	16.6%	30.6%	31.9%, α=4	α=235

[a] *Sn* achieved using MUM_k ($k \geq 1$) seeds, which are equivalent to AP_2 (the default value of *k* in MUMmer is 20).

[b] *Sn* achieved using MAM_k ($k \geq 1$) seeds. For the one-to-many mappings, the reference sequences (HS chr 15 and the Mpn genome, respectively, for Dataset A and B) were treated as the unique side for the comparisons.

[c] *Sn* achieved using AP_α at the specified α value.

Table 4.5 From MUM/MAM to α-pairs: improving $\bar{I}c$ by increasing α.

Sequences compared	$\bar{I}c(MUM_1)$[a] (=$\bar{I}c(AP_2)$)	$\bar{I}c(MAM_1)$[b]	$\bar{I}c$, α (AP_α)[c]	
			~$\bar{I}c(MAM_1)$[d]	α=20
HS chr.15, MM chr.7	1493.5	1603.0	1663.1, α=3	2260.0
HS chr.15, GG chr.10	323.0	357.3	377.3, α=3	555.7
HS chr.15, TN genome	51.7	68.8	73.9, α=4	115.1
Mpn, Mge genomes	62.7	78.7	83.9, α=3	120.3
Mpn, Rpr genomes	4.7	6.6	7.4, α=3	24.0
Mpn, Buc genomes	5.8	10.0	11.0, α=4	25.6
Mpn, Bbu genomes	4.0	5.4	5.9, α=3	23.1
Mpn, Ctr genomes	5.1	6.8	7.5, α=3	23.8
Mpn, Tac genomes	2.1	4.1	4.5, α=4	16.1

[a] $\bar{I}c$ achieved using MUM_k ($k \geq 1$) seeds, which are equivalent to AP_2.

[b] $\bar{I}c$ achieved using MAM_k ($k \geq 1$) seeds. For the one-to-many mappings, the reference sequences HS chr.15 (dataset A) and the Mpn genome (database B) were treated as the unique side for the comparisons.

[c] $\bar{I}c$ achieved using AP_α at the specified α value.

[c] At an $\bar{I}c$ value close to $\bar{I}c(MAM_1)$.

4.3.4 The number of α-pairs increases linearly with α

A striking property, and a great practical advantage, of α-pairs is that their number increases linearly as α increases (Figure 4.5). This linear relationship holds for all the comparisons we made, including those made on the vertebrate dataset (Figure 4.5), those on the prokaryote genomes (Figure 4.6 and Table 4.6), and those made on several self-comparisons of diverse genomic sequences (data not shown). In fact, the R^2 regression coefficient was so high (>0.99; Table 4.6) for these comparisons that we can reliably estimate the number of added seeds when we increase the copy number by one: i.e., if $|AP_\alpha|$ is known, where $|AP_\alpha|$ denotes the number of α-pairs at a specified α value, we can estimate $|AP_{\alpha+1}|$ to be roughly $(\alpha/(\alpha\text{-}1))\times|AP_\alpha|$ (derivation in Table 4.6). Thus,

the cost of enhancing the sensitivity by increasing the copy number in the α-pairs seeds is considerably smaller than that of enhancing the sensitivity by decreasing k in the length-based seeds, since the number of MEM_k or k-mer pairs grows exponentially as the word length k decreases (Kurtz 2001, Kent 2002).

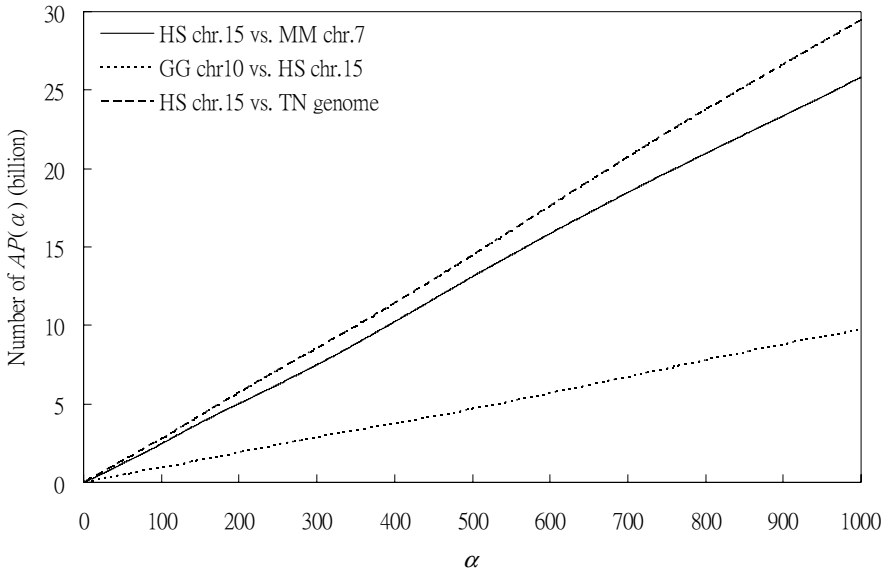


Fig. 4.5 Number of α-pair seeds as a function of α for the comparisons of vertebrate genomic sequences.

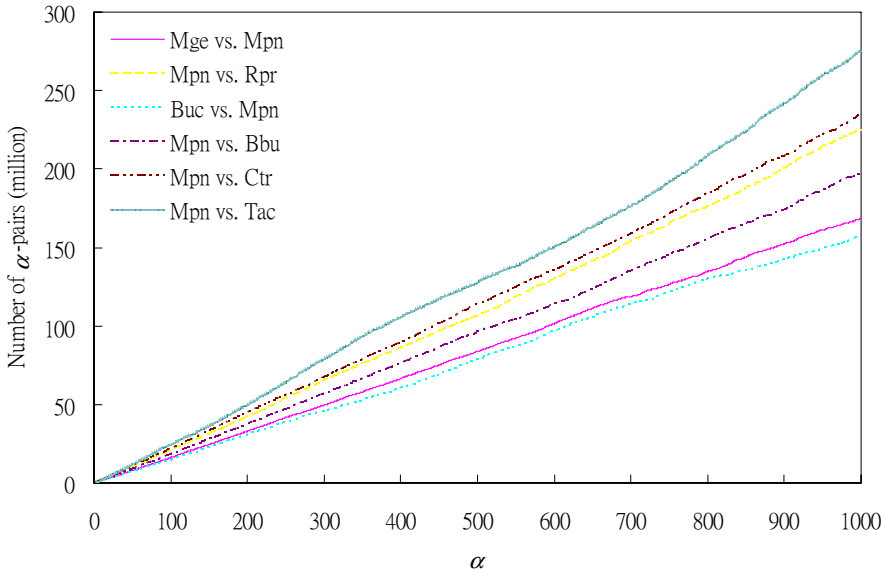


Fig. 4.6 Number of α-pair seeds as a function of α for the comparisons of prokaryote genomic sequences. The results of the linear regression analysis for each comparison are presented in Table 4.6.

Table 4.6. Results of linear regression analysis for the number of AP_α vs. α.

Sequences compared [a]	R^2	**m**[b]
HS chr.15, MM chr.7	0.9994	26099093
HS chr.15, GG chr.10	0.9990	9620470
HS chr.15, TN genome	0.9995	29431377
Mpn genome, Mge genome	0.9998	169095
Mpn genome, Rpr genome	0.9989	220301
Mpn genome, Buc genome	0.9985	158978
Mpn genome, Bbu genome	0.9993	193531
Mpn genome, Ctr genome	0.9993	229764
Mpn genome, Tac genome	0.9962	262727

[a] Double-strand DNA comparisons were performed. For each comparison, both strands of the smaller sequence were compared to the forward strand of the longer sequence, e.g., both strands of HS chr.15 (100 Mb) were compared to the forward strand of MM chr.7 (145 Mb).

[b] The regression model used is $|\boldsymbol{AP_\alpha}| = \mathbf{m}\cdot(\alpha\text{-}1)$ for α=2..1000. From $|AP_\alpha| = m\ (\alpha\text{-}1)$, we derived that $|AP_{\alpha+1}| = m\ (\alpha\text{+}1\text{-}1) = m\alpha = (m\alpha / m\ (\alpha\text{-}1))\times m\ (\alpha\text{-}1) = \alpha/(\alpha\text{-}1))\times|AP_\alpha|$. Thus, we can estimate $|AP_{\alpha+1}|$ if $|AP_\alpha|$ and α are known.

Chapter 5

Extending α-markers/α-pairs to discontiguous seeding models

5.1 Introduction

Recently, there have been advances in the *k*-mer method (Brown *et al.* 2004; Batzoglou 2005). One notable advance was the use of discontiguous seed, which computes only k' letter matches of each *k*-mer seed where $k'<k$. The idea of using discontiguous patterns of matching bases has been explored in order to enhance the sensitivity and/or speed of homology detection, such as detecting coding regions by ignoring wobble base pairs, (Kent and Zahler 2000), finding ungapped alignments with frequent substitutions by randomized seeding (Buhler 2001), and searching for homology by the PatternHunter method (Ma *et al.* 2002), which allows seed optimization and multiple seed models (Li *et al.* 2004). In principle, the α-marker method, like the *k*-mer method, can be extended to use discontiguous seeds. In this chapter, we will present a method to implement discontiguous seeding models for the α-marker method and then present the experiment results of orthology seeding using the same datasets mentioned in chapter 4. According to the results, some discontiguous seeding models, such as the wobble-aware model (Kent and Zahler 2000), achieved significant improvements in sensitivity/specificity

trade-off.

5.2 Methods

5.2.1 Discontiguous α-markers and α-pairs

5.2.1.1 Notations of discontiguous seeds of maximal length

Instead of using fixed-length seeds, the α-marker method uses maximal-length seeds. So we have to define notations for describing discontiguous seeds of maximal length. In Ma *et al.* (2002), they used binary strings to denote fixed-length discontiguous seeds (or called spaced seeds). In the binary strings, '1' denotes a required letter match and '0' denotes a "don't-care" letter position. In addition, the numbers of ones in the binary strings are defined as weights (Ma *et al.* 2002) in contrast to the lengths (or spanning lengths) of the binary strings. For example, a 5-mer exact match is represented as 11111, where both the weight and length of 11111 are five. A spaced seed, which requires five letter matches and two "don't care" letters of positions 4 and 6, is represented as 1110101, where its weight is five and its length is seven.

To describe discontiguous seeds of maximal length, we borrow the aforementioned notations and add some symbols from regular expressions. First we can add parentheses "()" in the seed model strings to mark substrings. Then we can add the superscript star symbol '*' following the parentheses to denote the string in the preceding parentheses can repeat zero, one, or many times. For example, $1(1)^*$ denotes exact matches of length

≥ 1 like 1, 11, 111, 1111, etc., and $1(011)^*$ denotes discontiguous matches of length ≥ 1 like 1, 1011, 1011011, etc.. Now the new notations are sufficient to describe discontiguous seeds of maximal length.

5.2.1.2 Reusing the α-marker method to generate discontiguous wobble seeds

In the WABA (Wobble-Aware Bulk Aligner) program of Kent and Zahler (2000), they proposed the use of seed model 11011011 of weight 6 to search homologous coding sequences owing to rapid divergence in the third, "wobble" positions of most codons, i.e., the 110 pattern. Following this idea, we can design many discontiguous seeds of fixed-lengths based on the 110 pattern, such as 101101101 of weight 6, 1101101101 and 1101101101 of weight 7, 11011011011 and 101101101101 of weight 8, etc.. Instead of listing a lot of such fixed-length seeds, we can summarize them as 110-based seeds of maximal length. In the following, we will reuse the α-marker method in chapter 4 with slight modifications to generate these 110-based seeds of maximal length with copy number constraint.

Since the α-marker method only cope with the exact matching scheme, first we need to transform the 110-discontiguous matching scheme into all possible reading frames of the exact matching scheme by ignoring the positions with '0' label. For example, given a sequence S = "123456789abcadb", if we read S from the first position using the pattern read-read-ignore that corresponds to 110, we will have S^1 =

"124578abad"; if we read *S* from the second position using the pattern read-read-ignore, we will have S^2 = "235689bcdb"; if we read *S* from the zero position using the pattern read-read-ignore, we will have S^0 = "134679acab". Obviously, any discontiguous subsequence of the pattern $(110)^*$ in *S* will constitute an exact match in either S^0, S^1, or S^2. However, the reverse will not always hold. For example, "ab" in S^1 exactly matches "ab" in S^0, but the corresponding subsequences in *S* are "ab" and "adb", respectively, which will not constitute a discontiguous match. To solve this problem, we assign a binary number *r* to each letter μ of S^0, S^1, and S^2, where r=0 indicates μ corresponds to the first letter of pattern 110 in *S*, r=1 indicates μ corresponds to the second letter of pattern 110 in *S*, and two letters with the same label *r* indicates the same position in pattern 110. Let $T = S^0\#S^1\#S^2$, where '#' is a special symbol to separate sequences. We can determine whether or not an exact match (*posT*1, *posT*2, *len*) of *T* corresponds to a discontiguous match of *S* by checking whether or not the *r* labels of *posT*1 and *posT*2 are equal. In practice, we design the following function to transform a position from the coordinate of *T* to the coordinate of *S* and the position's *r* label is also acquired in the function. Let f =0, 1, or 2 denote the number of reading frame, *posT* denote a position in *T* and *posS* denote the correspondent position of *posT* in *S*.

$$posS = 3 \times q + r + f, \tag{5.1}$$

where $q = (posT - \theta)/2$, $r = (posT - \theta) \bmod 2$, and $\theta = \begin{cases} 0 & \text{if } f = 0, \\ 1 & \text{if } f = 1, 2. \end{cases}$

Meanwhile, it is simple to generate S^f from S by skipping the (f+2 mod 3)th base periodically for f=0,1,2. Thus, the 110-discontiguous matching problem of S can be transformed into the contiguous matching problem of T. The positions of exact matches in T can trace back to the corresponding positions in S by using Equation 5.1.

The steps for generating wobble-aware α-pairs are almost the same as the steps for generating exact α-pairs in chapter 4, except the preprocessing step and the postprocessing filter by r label checking. In the preprocessing step, we need to transform each input sequence to the three reading frames and concatenate them into one sequence. Then, we build the enhanced suffix array for the concatenated sequence. Next, bottom-up traverse the enhanced suffix array and generate the MEMs constrained by copy number $c \leq \alpha$ as mentioned in section 4.2.2. In the postprocessing step after the MEM are generated, we only report MEMs whose two string instances starting from positions of the same r label to avoid the matches with gaps. The reported MEMs are exactly all the discontiguous wobble-aware seeds with a copy number constraint. If we want to generate discontiguous wobble-aware seeds with a weight constraint, the steps are the same as the above except we generate MEMs constrained by weights, which correspond to lengths in the transformed sequences and lcp values mentioned in section 4.2.2. Also, we can specify both an upper bound of copy numbers and a lower bound of weight/length to generate more complex α-pair seeds by checking both the sizes and lcp

values of lcp intervals in this step of MEM generation.

In addition, the seed models of the discontiguous wobble-aware seeds generated above will cover all variations based on pattern $(110)^*$, including $(110)^*11$ and $10(110)^*1$ for even weight, and $(110)^*1$ and $1(011)^*$ for odd weight. This is a different feature from discontiguous wobble-aware seeds of fixed length.

5.2.2 Evaluation of orthology seeding

To evaluate the ability of different types of seeds, including contiguous and discountiguous matching schemes, to detect orthologues, we extended the evaluation measure: seeding sensitivity (*Sn*) described in section 4.2.3 and added a new measure: seeding specificity (*Sp*) to make the evaluation more complete. First, we used orthologues (i.e., orthologus gene pairs) from Ensembl orthology (Hubbard *et al*. 2007) or COG (Tatusov *et al*. 2003) as the answers of orthology detection and the genes that occur in the orthologues are called test genes. Then we defined non-orthologues as the cross-species test gene pairs that are not orthologues. After that, we used a criterion, called *t*-seed test, to check whether a gene pair is predicted as an orthologue or not. A gene pair is said to be a positive prediction under *t*-seed test if this gene pair contains at least *t* nonoverlapped seeds, where a seed denotes a pair of two identical subsequences without indels and two seeds are said to be overlapped if they are overlapped at both sides of the two compared sequences and can be merged into a longer seed without

introducing any gap. A gene pair is said to be a negative prediction under *t*-seed test if this gene pair contains at most *t*-1 nonoverlapped seeds. Let TP denote the number of orthologues that contain at least *t* nonoverlapped seeds, FP denote the number of non-orthologues that contain at least *t* nonoverlapped seeds, TN denote the number of non-orthologues that contain at most *t*-1 nonoverlapped seeds, and FN denote the number of orthologues that contain at most *t*-1 nonoverlapped seeds, we define seeding sensitivity (*Sn*) and specificity (*Sp*)under *t*-seed test as

$$Sn = 100\% \times TP/(TP + FN)$$, and

$$Sp = 100\% \times TN/(TN + FP)$$.

To better understand the sensitivity-specificity trade-offs among parameters for different seeding methods, we plotted ROC (Receiver Operating Characteristic) curves (Fawcett 2004), which use *Sn* as the *x*-axis and 1-*Sp* as the *y*-axis, for each experiment in the results.

5.3 Results

In this section, we used ROC curves and figures of colinear identities vs. total number of seeds to compare three types of discontiguous and contiguous seeding methods, including discontiguous wobble-aware seeds of maximal length, spaced *k*-mer seeds, and contiguous seeds of maximal length. Discontiguous wobble-aware seeds of maximal length that we used consist of wobble-aware α-pairs and wobble-aware MEMs

mentioned in section 5.2. Spaced *k*-mer seeds that we used include WABA-like 110-based seeds (Kent and Zahler 2000), Choi's good spaced seeds for homology search (Choi *et al*. 2004), and the alternative pattern $(10)^x$ of fixed length. The detail patterns and weights of the spaced *k*-mer seeds we used are listed in Table 5.1. Contiguous seeds of maximal length that we used are composed of exact α-pairs and exact MEMs mentioned in section 4.2. The datasets we used in the following experiments are listed in Table 4.1.

Table 5.1 Spaced *k*-mer seeds used in this study.

(A) Choi's good spaced seeds

SeedPattern	Weight	SpanLen
111010110100110111	12	18
11101011001100101111	13	20
111011100101100101111	14	21
11110010101011001101111	15	23
11110011010101100110111 1	16	24

(B) Waba-like 110-based spaced seeds

SeedPattern	Weight	SpanLen
11011011011011011	12	17
1101101101101101101	13	19
11011011011011011011	14	20
11011011011011011011101	15	22
11011011011011011011011	16	23

(C) Spaced seeds of alternative pattern $(10)^x$

SeedPattern	Weight	SpanLen
101010101010101010101	12	23
1010101010101010101010101	13	25
10101010101010101010101010101	14	27
101010101010101010101010101010101	15	29
1010101010101010101010101010101010101	16	31

5.3.1 Comparisons of ROC curves for wobble-aware *α*-pairs/MEMs, spaced *k*-mer seeds and exact *α*-pairs/MEMs

In this section, we will compare the sensitivity-specificity trade-offs among parameters of different seeding methods for the vertebrate and prokaryote datasets described in Table 4.1 using ROC (Receiver Operating Characteristic) curves (Fawcett 2004). The

seven seeding methods used in this study are summarized in Table 5.2. In Table 5.2, we combine α-pairs (denoted as AP(α)) with maximal unique matches (denoted as MUM(k)) because they are both copy number-based, complement to each other in terms of the full range of specificity, and can be generated by traversing the enhanced suffix array once as mentioned in sections 5.2.1 and 4.2.2.

Table 5.2 Features of the seven seeding methods used in this study.

Seeding method	Matching scheme	Maximal or Fixed length	Weight/Length contraint	Copy number constraint
Exact MEM(k)	Contiguous	Maximal	Yes	No
Exact AP(α)+MUM(k)	Contiguous	Maximal	Yes (for α=2)	Yes
Waba spaced model	Discontiguous (110-based)	Fixed	Yes	No
Alt. spaced model	Discontiguous (10-based)	Fixed	Yes	No
Choi's spaced model	Discontiguous (Choi's selection)	Fixed	Yes	No
Wob. MEM(k)	Discontiguous (110-based)	Maximal	Yes	No
Wob. AP(α)+MUM(k)	Discontiguous (110-based)	Maximal	Yes (for α=2)	Yes

As shown in Figure 5.1, for human chr.15 vs. pufferfish genome, we plotted ROC curves of the seven seeding methods (Table 5.2). For the contiguous matching scheme, the ROC curve of exact AP(α)+MUM(k) were significantly closer to the upper left than the ROC curve of exact MEM(k) in 1,2,3,5,10,20-seed tests, which means incorporating

copy number constraints to contiguous seeding methods can achieve not only higher sensitivity but also higher specificity. To measure the quantity of differences for ROC curves, we computed the AUC (Area Under ROC Curve) values for ROC curves (Fawcett 2004) that have full ranges of specificity (i.e., from 0 to 1) of the two datasets (Table 4.1) and showed the results in Table 5.3. For example, in Figure 5.1A, the AUC values of exact AP(α)+MUM(k) and exact MEM(k) are 0.837 and 0.703 respectively, where the difference is 0.134.

For the discontiguous matching scheme, we first compared the three kinds of spaced k-mer seeds of weights 12, 13, 14, 15, and 16 (Table 5.1). In Figure 5.1A, we found WABA-like spaced seeds performed better than Choi's spaced seeds, and spaced seeds of alternative pattern $(10)^x$ performed the worst. This is related to that we used orthologous gene pairs as the benchmarks and they always contain coding sequences, where the pattern 110 is designed for coding sequences (Kent and Zahler 2000). Such observations are concordant to the results of finding optimal seeds for homologous coding regions of Brejova *et al*. (2004). As for Figure 5.1B-F, the results of spaced k-mer seeds are consistent with the corresponding results in Figure 5.1A.

For discontiguous seeds of maximal length, the ROC curve of wobble-aware AP(α)+MUM(k) outperformed all the other seeding methods in Figure 5.1 and the ROC curve of wobble-aware MEM(k) was highly overlapped with the ROC curve of

WABA-like spaced seeds. The AUC values of wobble-aware AP(α)+MUM(k) and wobble-aware MEM(k) are 0.961 and 0.930 respectively in Figure 5.1A, where the difference is 0.031. This reveals that incorporating copy number constraints to discontiguous wobble-aware seeding can achieve both higher sensitivity and higher specificity.

The ROC curves of the seven seeding methods (Table 5.2) using 1,2,3,5,10,20-seed tests for human chr.15 vs. chicken chr.10 and pufferfish genome are shown in Figure 5.2 and Figure 5.3, which showed similar advantages of incorporating copy number constraints to contiguous and discontiguous seeds of maximal length.

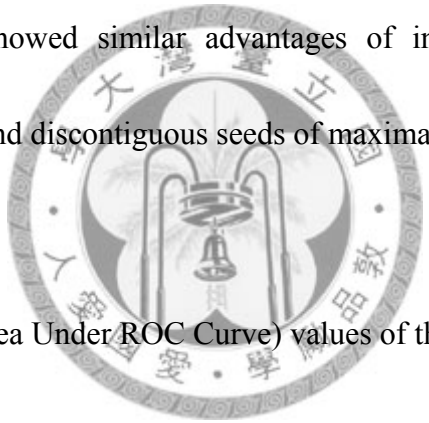


Table 5.3 List of AUC (Area Under ROC Curve) values of the exact and wobble-aware matching schemes

(A) Exact matching scheme

Sequences compared	1-seed test		2-seed test		3-seed test		5-seed test		10-seed test		20-seed test	
	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k
HS chr.15, MM chr.7	0.9949	0.8657	0.9985	0.8164	0.9974	0.7743	0.9868	0.7304	0.9609	0.6736	0.8740	0.6351
HS chr.15, GG chr.10	0.9272	0.8499	0.9061	0.8031	0.8763	0.7606	0.8243	0.7058	0.7494	0.6351	0.6671	0.5840
HS chr.15, TN genome	0.8368	0.7028	0.7898	0.6519	0.7601	0.6209	0.7107	0.5930	0.6477	0.5610	0.5924	0.5463
Mpn, Mge genomes	0.8891	0.8874	0.8803	0.8820	0.8729	0.8722	0.8479	0.8450	0.7782	0.7584	0.7052	0.6934
Mpn, Rpr genomes	0.6330	0.6232	0.5998	0.6052	0.5993	0.6009	0.5963	0.5930	0.5865	0.5893	0.5823	0.5841
Mpn, Buc genomes	0.6266	0.6220	0.6072	0.6152	0.5932	0.5845	0.5787	0.5801	0.5704	0.5559	0.5536	0.5467
Mpn, Bbu genomes	0.5934	0.5999	0.6076	0.6017	0.6013	0.5832	0.5852	0.5815	0.5775	0.5727	0.5745	0.5630
Mpn, Ctr genomes	0.6073	0.5897	0.6023	0.5878	0.6000	0.5871	0.5905	0.5797	0.5788	0.5738	0.5629	0.5569
Mpn, Tac genomes	0.5492	0.5497	0.5638	0.5571	0.5723	0.5677	0.5717	0.5553	0.5659	0.5512	0.5670	0.5654

(B) Wobble-aware matching scheme

Sequences compared	1-seed test		2-seed test		3-seed test		5-seed test		10-seed test		20-seed test	
	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k	AP_α	MEM_k
HS chr.15, MM chr.7	0.9994	0.9877	0.9997	0.9696	0.9999	0.9525	0.9969	0.9121	0.9713	0.8382	0.8982	0.7473
HS chr.15, GG chr.10	0.9698	0.9725	0.9571	0.9516	0.9417	0.9358	0.9231	0.9039	0.8549	0.8182	0.7557	0.7144
HS chr.15, TN genome	0.9612	0.9295	0.9343	0.8846	0.9202	0.8557	0.8755	0.8003	0.7901	0.7021	0.6932	0.6295
Mpn, Mge genomes	0.9346	0.9395	0.9291	0.9308	0.9192	0.9227	0.8969	0.8918	0.8325	0.8230	0.7257	0.7161
Mpn, Rpr genomes	0.7233	0.7158	0.6926	0.6857	0.6702	0.6659	0.6355	0.6358	0.6161	0.6055	0.6045	0.6061
Mpn, Buc genomes	0.7412	0.7290	0.7080	0.6887	0.6753	0.6653	0.6464	0.6257	0.6028	0.5900	0.5749	0.5750
Mpn, Bbu genomes	0.7201	0.7021	0.6991	0.6805	0.6722	0.6591	0.6480	0.6384	0.6195	0.6065	0.5996	0.5799
Mpn, Ctr genomes	0.7148	0.7291	0.6828	0.6823	0.6601	0.6601	0.6285	0.6347	0.6016	0.5888	0.5838	0.5780
Mpn, Tac genomes	0.6457	0.6422	0.6407	0.6320	0.6209	0.6242	0.6085	0.5998	0.5878	0.5875	0.5802	0.5715

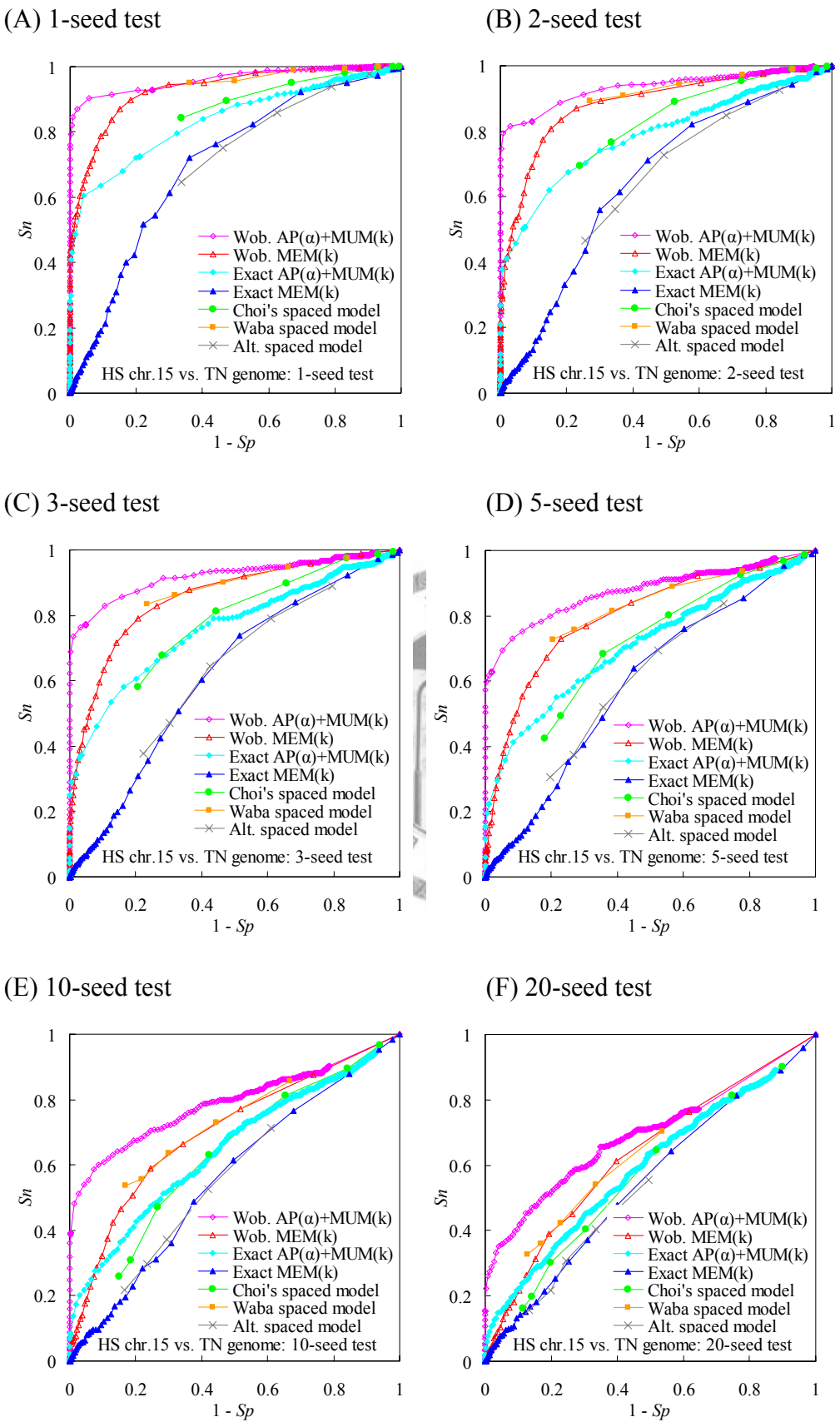


Fig. 5.1 ROC curves of the seven seeding methods (Table 5.2) using 1,2,3,5,10,20-seed tests for human chr.15 vs. pufferfish genome.

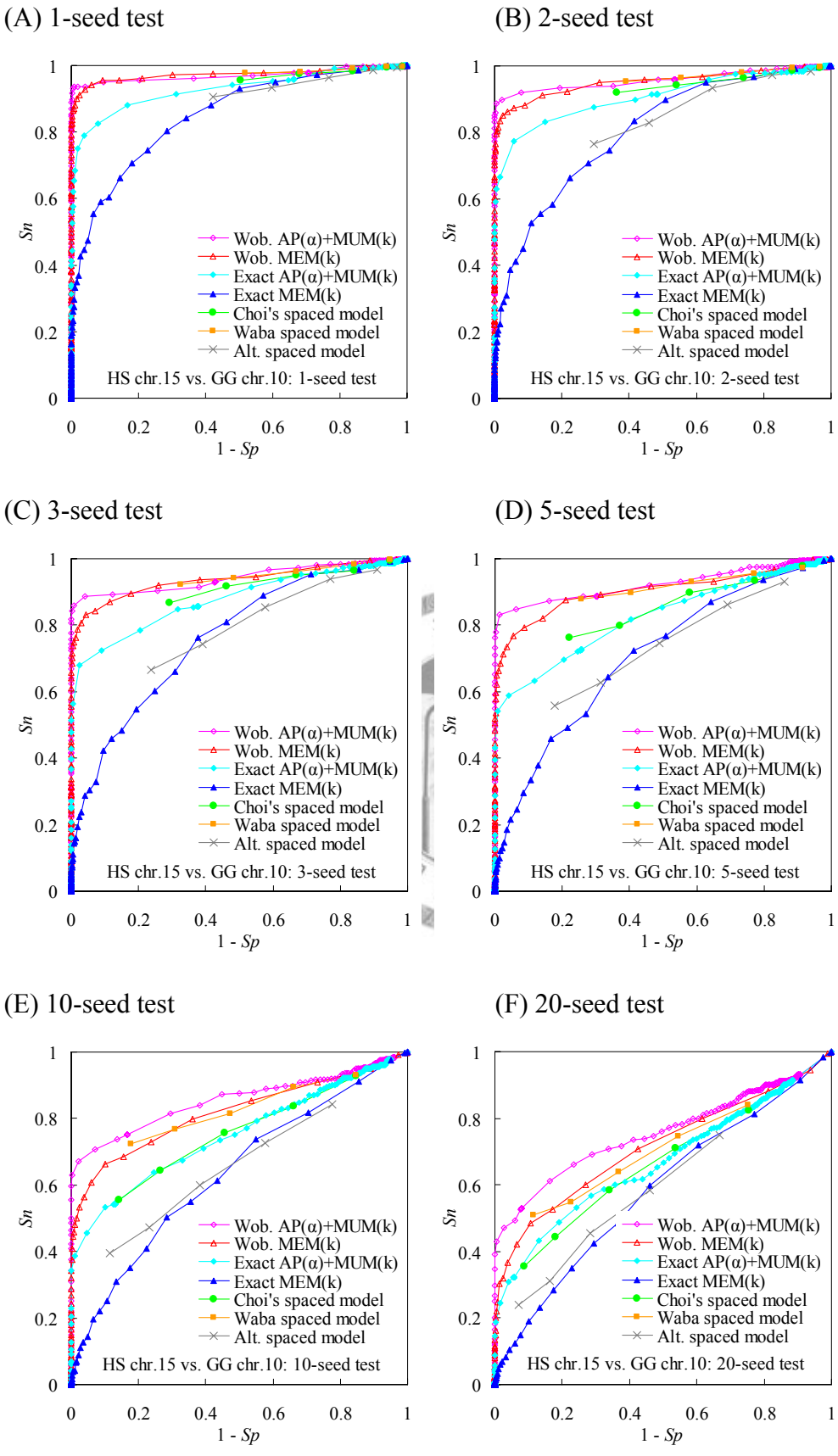


Fig. 5.2 ROC curves of the seven seeding methods (Table 5.2) using 1,2,3,5,10,20-seed tests for human chr.15 vs. chicken chr.10.

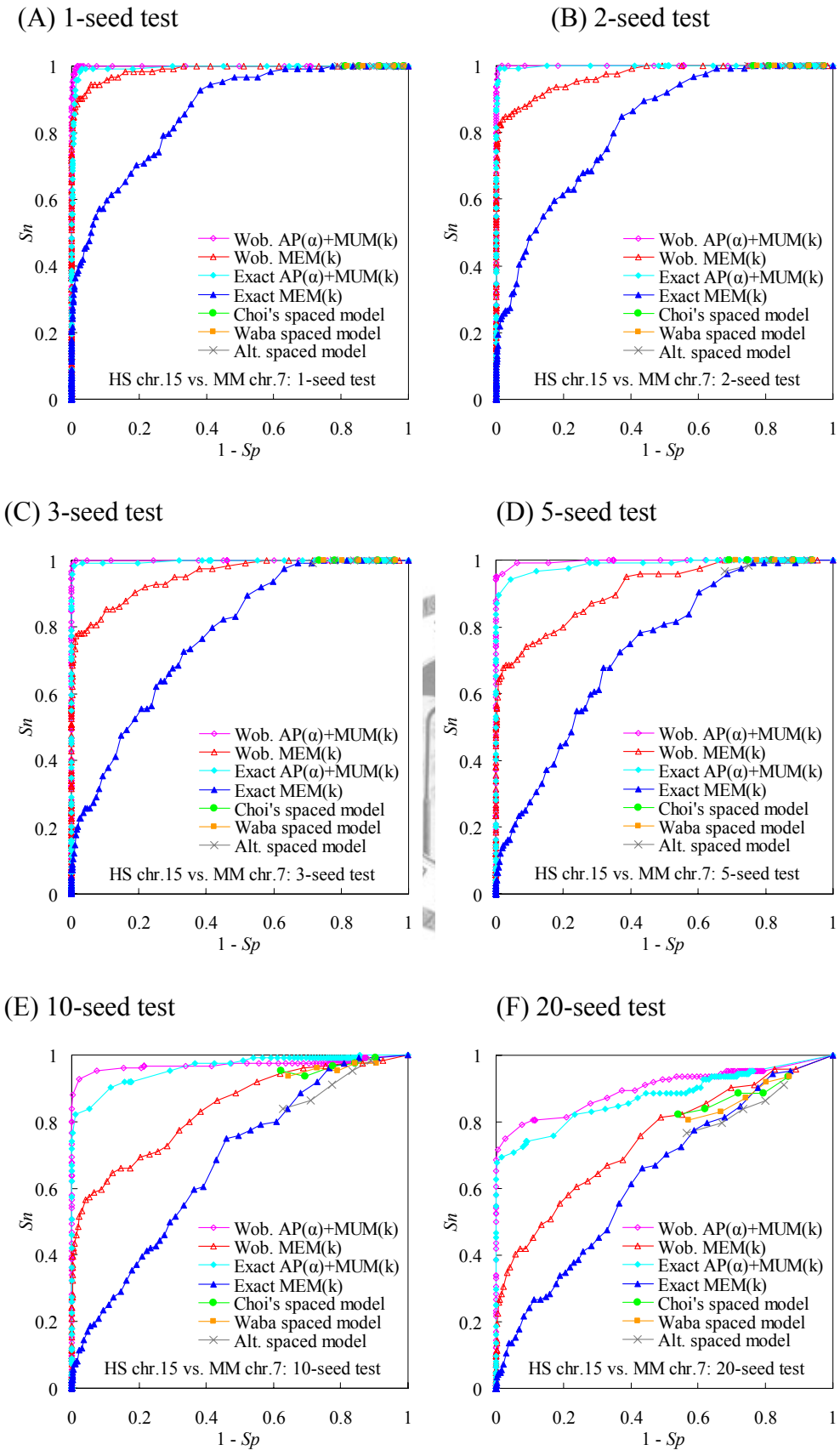


Fig. 5.3 ROC curves of the seven seeding methods (Table 5.2) using 1,2,3,5,10,20-seed tests for human chr.15 vs. mouse chr.7.

In Figure 5.4, we plotted AUC values vs. testing methods of the seven seeding methods (Table 5.2) for the vertebrate dataset (Table 4.1A) to visualize the differences between exact/wobble-aware AP(α)+MUM(k) and MEM(k).

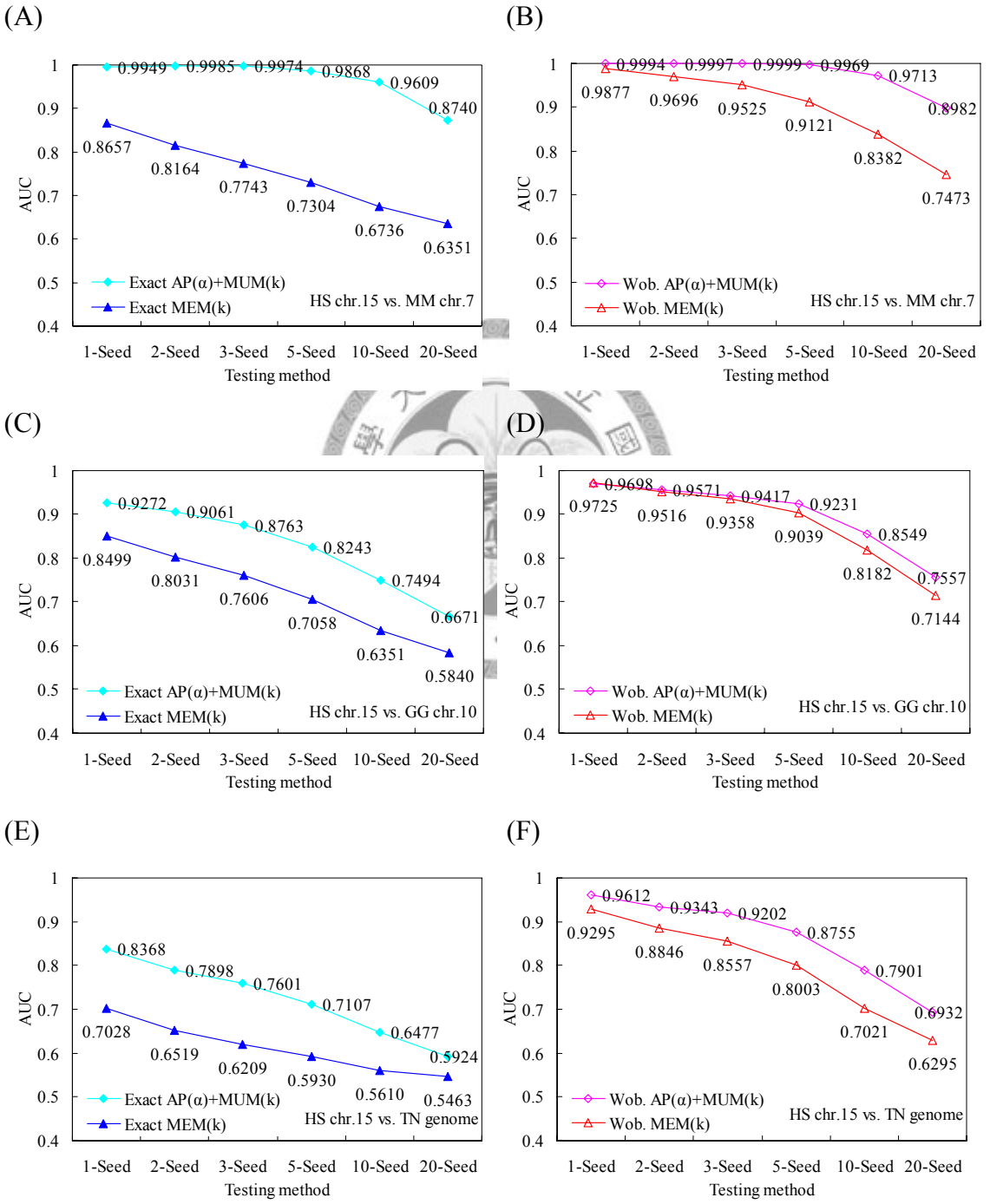


Fig. 5.4 AUC values vs. testing methods of the seven seeding methods (Table 5.2) for the vertebrate dataset (Table 4.1A).

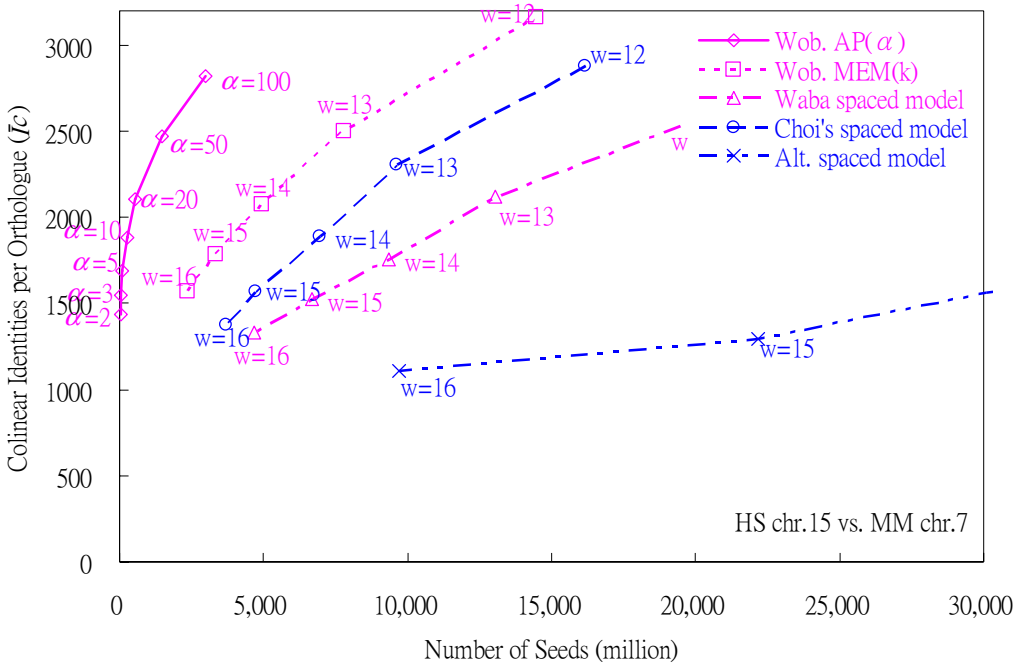
5.3.2 Comparisons of colinear identities vs. total number of seeds for wobble-aware α-pairs/MEMs, spaced k-mer seeds and exact α-pairs/MEMs

Here we use another viewpoint to compare different seeding methods used in this study. In Figures 5.5-7, we plotted $\bar{I}c$ (defined in section 4.2.3) vs. total number of seeds generated using the seven seeding methods in Table 5.2 and the exact k-mer seeding method in the comparison of human chr.15 vs. mouse chr.7, chicken chr.10 and pufferfish genome. For human chr.15 vs. mouse chr.7 in Figure 5.5, we found the curve of wobble-aware AP(α) was closer to the upper left than that of wobble-aware MEM(k), and the curve of wobble-aware MEM(k) was closer to the upper left than that of WABA spaced models. This means less seeds are required for wobble-aware AP(α) to achieve equal colinear identities per orthologues than that for wobble-aware MEM(k) and that for WABA spaced models. Similar trends are found in Figures 5.6-7.

For spaced k-mer seeds, Choi's spaced seeds performed better than WABA spaced seeds in Figure 5.5, similar to WABA spaced seeds in Figure 5.6, and less than WABA spaced seeds in Figure 5.7. But Choi's spaced seeds performed less than wobble-aware MEM(k) in Figures 5.5-7. As for spaced seeds based on pattern 10, they performed the worst among the three kinds of spaced k-mer seeds we used in Figures 5.5-7.

Comparing Figures 5.5-7A to Figure 5.5-7B, we found the curves of wobble-aware AP(α) and exact AP(α) for human chr.15 vs. mouse chr.7 are of similar heights. The curve of wobble-aware AP(α) is a little higher than that of exact AP(α) for human chr.15 vs. chicken chr.10. The curve of wobble-aware AP(α) is significantly higher than that of exact AP(α) for human chr.15 vs. puffersih. This reveals that for the comparison of distant genomes like human vs. fish, discontiguous wobble-aware seeds can perform much better than contiguous seeds in orthology seeding. For the comparison of closer genomes, the enhancement by discontiguous wobble-aware seeds is less profound.

(A)



(B)

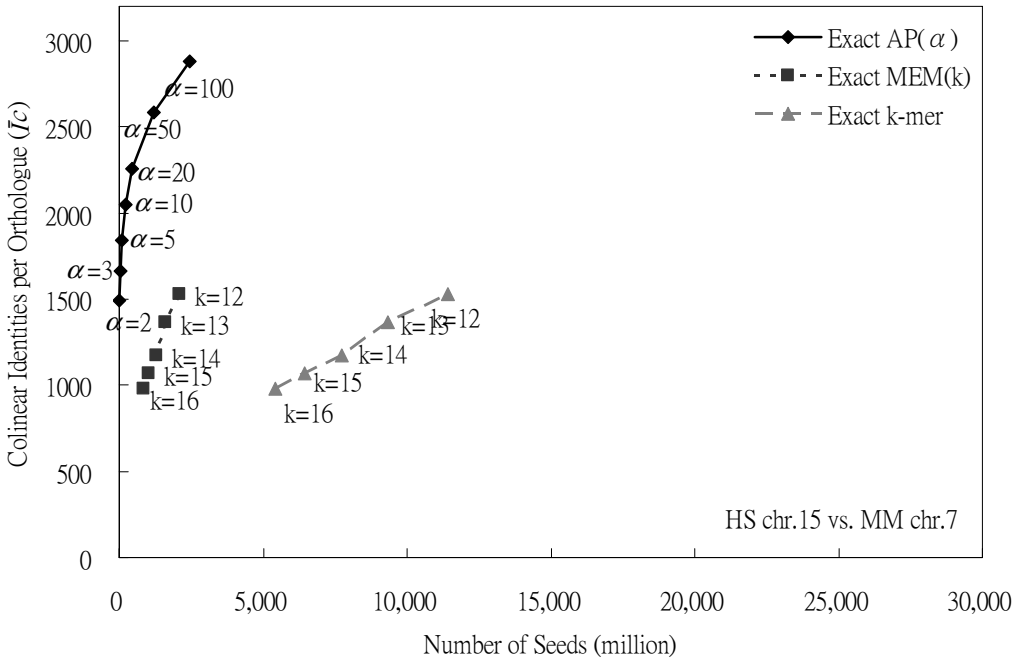
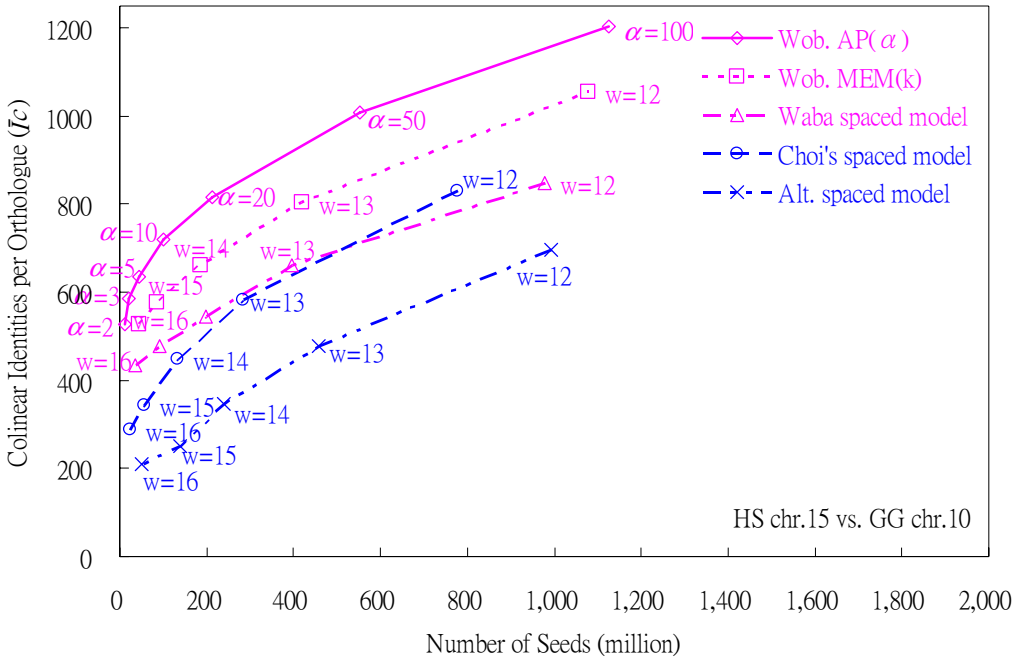


Fig. 5.5 $\bar{I}c$ vs. total number of seeds generated using (A) several discontiguous seed models and (B) several contiguous seed models in the comparison of human chr.15 vs. mouse chr.7.

(A)



(B)

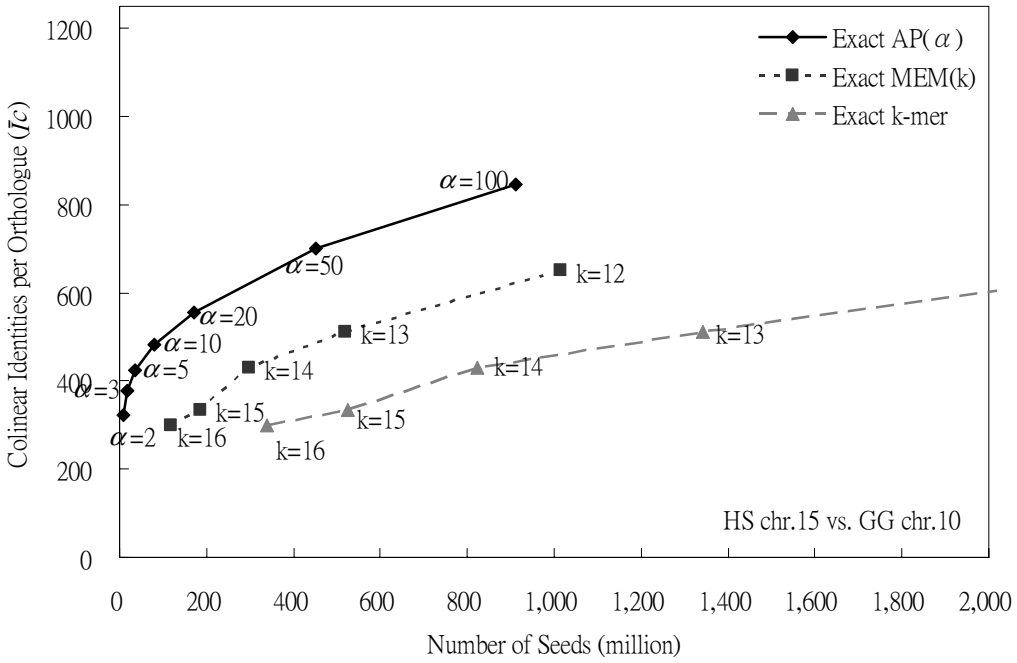
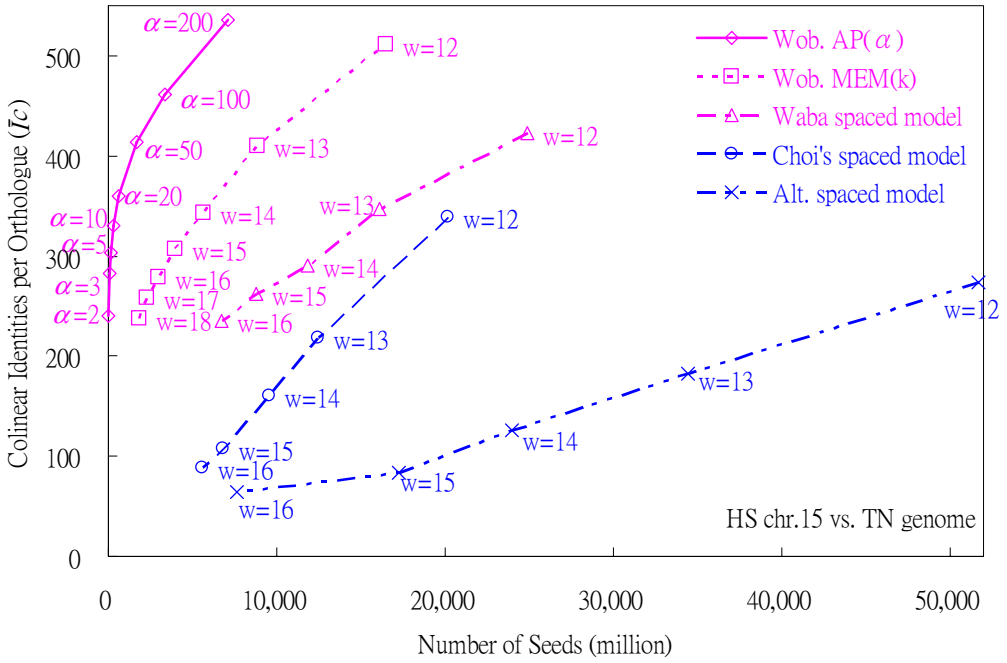


Fig. 5.6 $\bar{I}c$ vs. total number of seeds generated using (A) several discontiguous seed models and (B) several contiguous seed models in the comparison of human chr.15 vs. chicken chr.10.

(A)



(B)

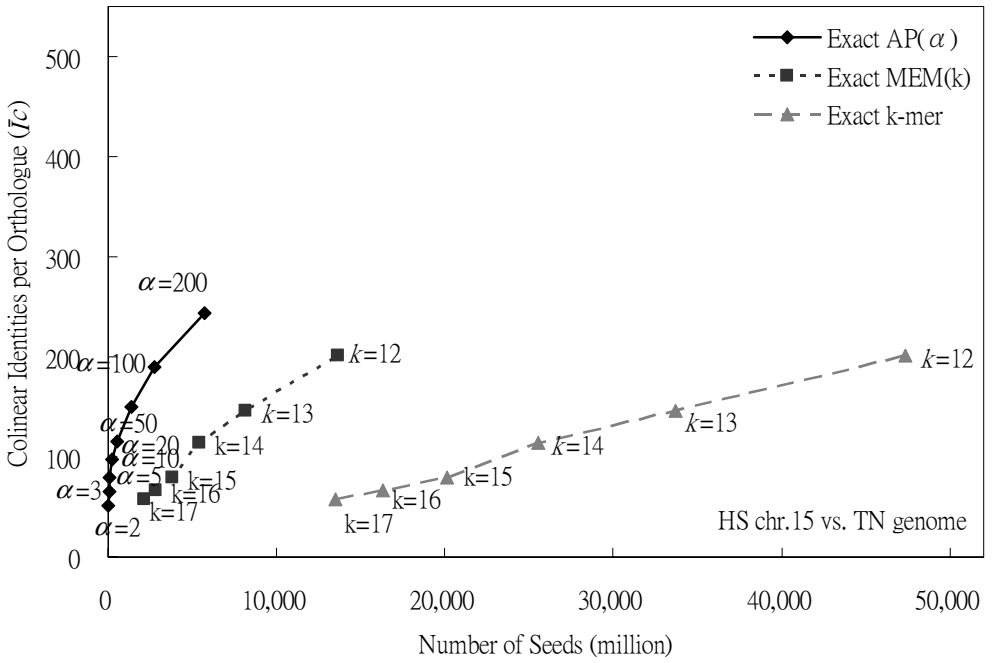


Fig. 5.7 $\bar{I}c$ vs. total number of seeds generated using (A) several discontiguous seed models and (B) several contiguous seed models in the comparison of human chr.15 vs. pufferfish genome.

Chapter 6

Discussion and conclusions

6.1 Discussion

Orthology seeding is the process used to locate pieces of sequence matches to detect orthologous regions among genomes. By definition, orthologous regions are homologous regions shared by two genomes from a speciation event, or, more specifically, regions that have originated from a single ancestral genomic region in the last common ancestor of the compared genomes (Kooin 2005). Because losses or duplications of genes or genomic regions can occur after speciation, orthologous relationships are not just one-to-one, but may become many-to-many or may even cease to exist (Theiβen 2002). Thus, complete orthology identification necessarily involves the consideration of co-orthologous regions, defined as two or more genomic regions in the same lineage that are collectively orthologous to one or more genomic regions in another lineage due to a lineage-specific duplication (Koonin 2005). It therefore seems, at least conceptually, that copy number-based seeding is intrinsically more capable of capturing the ramifications of evolutionary processes than length-based seeding. Furthermore, because orthologous relationships that have not yet experienced losses or duplications following speciation tend to involve one-to-one mapping and co-orthologous relationships tend to be only involve "several-to-several" mapping, seeds of lower copy numbers should be more relevant to orthology detection than seeds with higher copy-number. These considerations provided the basic ideas behind our development of the upper bounded α-marker method, which also underscores the

feasibility of using a highly streamlined method, such as UniMarker which is essentially a special case of α-pairs at α=2, for mapping relatively close genomes, such as human and mouse (Liao et al., 2004).

Although the aforementioned copy number-based seed model is conceptually simple, the complete, compact and efficient enumeration of the required, relatively low-copy, word matches of any length is not. In this contribution, we showed that this can be done in linear complexity (see Methods). Furthermore, we showed that copy number-based seeds compared favorably with length-based seeds in seeding vertebrate and prokaryote orthologues, although the extent of performance gain cannot be simply explained by evolutionary distance or genome/chromosome size alone (Figures 4.2 and 4.4 and Tables 4.2-5). It is also not clear whether there is a biological basis for the linear growth of α-pairs (Figures 4.5-6), but this hitherto unobserved property of genomic sequences nevertheless reveals an exciting potential for scaling up to map distant genomes and for investigating genome evolution.

Recently, there have been advances in the *k*-mer method (Brown *et al.* 2004; Batzoglou 2005). One notable advance was the use of discontiguous seed, which computes only k' letter matches of each *k*-mer seed where $k'<k$. The idea of using discontiguous patterns of matching bases has been explored in order to enhance the sensitivity and/or speed of homology detection. In chapter 5, we designed discontiguous wobble-aware seeds of maximal length to detect orthologues and demonstrated that we can fulfill the design using enhanced suffix arrays with copy number constraints and weight/length constraints. According to the results of ROC curves for the vertebrate dataset in section 5.3.1, the advantages of incorporating copy number constraints to contiguous or discontiguous wobble-aware seeds were profound. One challenging issue of using discontiguous seeds of maximal length is the pattern design problem, which is

more restricted than using discontiguous seeds of fixed length so far. It remains an open issue to design discontiguous seeds of maximal length for noncoding sequence comparison.

In addition, besides the pairwise comparison described above, it is straightforward to use the α-marker method for self and multiple comparisons. One only needs to modify the α-pairs generating step by selecting α-pairs from $P(1,x_1,\sigma)$ and $P(1,x_1',\sigma)$ where $x_1 \neq x_1'$ for self comparison and from $P(i,x_i,\sigma)$ and $P(j,x_j,\sigma)$, $x_i \neq x_j$, for multiple comparison, where $1 \leq i < j \leq g$ and g denotes the number of compared genomic sequences. Finally, it should be possible to incorporate copy number seeds into various post-seeding processes in programs such as the pairwise genome alignment tools MUMmer3 (Kurtz *et al.* 2004) and AVID (Bray *et al.* 2003), the genome rearrangement locator GRIL (Darling *et al.* 2004a), the multiple genome alignment tools MGA (Höhl *et al.* 2002) and Mauve (Darling *et al.* 2004b), and the synteny-mapping UniMarker method (Liao *et al.* 2004).

6.2 Conclusions

In the dissertation, we first proposed the UM method for synteny mapping of closely related genomes. The UM method is highly efficient by its alignment-free design and the whole synteny mapping process of giga-base genomes, such as human and mouse, can be completed in a few hours on single desktop computer with ordinary CPU and RAM. Second, we proposed the α-marker method for orthology seeding, generalized from MUM and UniMarker, suitable for from closely related genomes to not closely related genomes. Results from comparing to various length-based seeds in detecting the Ensembl and COG orthologues for several vertebrate genomes/chromosomes and prokaryote genomes of long evolutionary distances suggest that orthology seeding via

copy number can achieve higher sensitivity and better efficiency than orthology seeding via length. Furthermore, we extend the α-marker method to generate discontiguous wobble-aware seeds of maximal length with copy number constraints. The comparative results of ROC curves for human chr.15 vs. mouse chr.7, chicken chr.10, and pufferfish genome showed that discontiguous wobble-aware α-pairs achieved significantly better performances than spaced *k*-mer seeding methods tested.

Bibliography

Abouelhoda,M.I., Kurtz,S. and Ohlebusch,E. (2004) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*. **2**, 53-86.

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403-410.

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389-3402.

Batzoglou,S. *et al.* (2000) Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Res.*, **10**, 950-958.

Batzoglou,S. (2005) The many faces of sequence alignment. *Brief. Bioinformatics*, **6**, 6-22.

Bray,N. *et al.* (2003) AVID: a global alignment program. *Genome Res*., **13**, 97-102.

Brejova,B., Brown,D.G., and Vinar,T. (2004) Optimal spaced seeds for homologous coding regions. *J. Bioinf. and Comput. Biol.*, **1**, 595-610.

Brown,D.G., Li,M. and Ma,B. (2004) A tutorial of recent developments in the seeding of local alignment. *J. Bioinf. and Comput. Biol.*, **2**, 819-842.

Brudno,M. and Morgenstern,B. (2002) Fast and sensitive alignment of large genomic sequences. *Proc. IEEE Computer Society Bioinformatics Conference (CSB)*.

Buhler,J. (2001) Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, **17**, 419-428.

Burrows,M. and Wheeler,D.J. (1994) A block-sorting lossless data compression algorithm. Research Report 124, Digital Systems Research Center.

Chain,P. *et al.* (2003) An application-focused review of comparative genomics tools:

capabilities, limitations and future challenges. *Brief. Bioinformatics*, **4**, 105-123.

Chen,L.Y., Lu,S.H., Shih,E.S. and Hwang M.J. (2002) Single nucleotide polymorphism mapping using genome-wide unique sequences. *Genome Res.*, **12**, 1106-1111.

Choi,K.P., Zeng,F. and Zhang,L. (2004) Good spaced seeds for homology search. *Bioinformatics*, **20**, 1053 - 1059.

Clamp,M. *et al*. (2003) Ensembl 2002: accommodating comparative genomics. *Nucleic Acids Res.*, **31**, 38-42.

Darling,A.C.E. *et al.* (2004a) GRIL: genome rearrangement and inversion locator. *Bioinformatics*, **20**, 122-124.

Darling,A.C.E. *et al.* (2004b) Mauve: multiple alignment of conserved genomic sequence with rearrangement. *Genome Res.*, **14**, 1394-1403.

Delcher,A,L., Kasif,S., Fleischmann,R.D., Peterson,J., White,O. and Salzberg,S.L. (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369-2376.

Delcher,A,L., Phillippy,A., Carlton,J. and Salzberg,S.L. (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.*, **30**, 2478-2483.

Deway,C.N. and Pachter,L. (2006) Evolution at the nucleotide level: the problem of multiple whole-genome alignment. *Human Molecular Genet.*, **15**, R51-R56.

Ehrlich,J., Sankoff,D. and Nadeau,J.H. (1997) Synteny conservation and chromosome rearrangements during mammalian evolution. *Genetics*, **147**, 289-296.

Frazer,K.A. *et al*. (2003) Cross-species sequence comparisons: a review of methods and available resources. *Genome Res*., **13**, 1-12.

Fawcett,T. (2004). ROC Graphs: Notes and practical considerations for researchers. Technical report, Palo Alto, USA: HP Laboratories.

Fitch,W.M. (2000) Homology a personal view on some of the problems. *Trends Genet.*, **16**, 227-231.

Gregory,S.G. *et al*. (2002) A physical map of the mouse genome. *Nature*, **418**, 743–750.

Gusfield,G. (1997) Algorithms on strings, trees, and sequences. Cambridge University Press, New York.

Hedges,S.B. (2002) The origin and evolution of model organisms. *Nature Rev. Genet.*, **3**, 838-849.

Hubbard,T.J. *et al*. (2002) The Ensembl genome database project. *Nucleic Acids Res.*, **30**, 38-41.

Hubbard,T.J. *et al.* (2007) Ensembl 2007. *Nucleic Acids Res.*; **35**, D610-D617.

Höhl,M., Kurtz,S. and Ohlebusch,E. (2002) Efficient multiple genome alignment, *Bioinformatics*, **18**, S312-S320.

Jaillon,O. *et al*. (2004) Genome duplication in the teleost fish *Tetraodon nigroviridis* reveals the early vertebrate proto-karyotype. *Nature*, **431**, 946–957.

Kasai,T. *et al.* (2001) Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Proc. Annu. Symp. on Combinatorial Pattern Matching,* Lecture Notes in Comput. Sci., **2089**, Springer, 181-192.

Kent,W.J. and Zahler,A.M. (2000) Conservation, regulation, synteny, and introns in large-scale *C. briggsae-C. elegans* genomic alignment. *Genome Res*., **10**, 1115-1125.

Kent,W.J. (2002) BLAT: the BLAST-like alignment tool. *Genome Res*., **12**, 656-664.

Koonin,E.V. (2005) Orthologs, paralogs, and evolutionary genomics. *Annu. Rev. Genet.*, **39**, 309-338.

Kurtz,S. (1999) Reducing the Space Requirement of Suffix Trees. *Software Pract. Exper*, **29**, 1149-1171.

Kurtz,S. *et al.* (2001) REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Res.*, **29**, 4633-4642.

Kurtz,S. and Lonardi,S. (2004) Computational biology, handbook on data structures and applications. Mehta,D.P. and Sahni,S. (editors), Chapman and Hall/CRC computer and information science series 2004.

Kurtz,S. *et al.* (2004) Versatile and open software for comparing large genomes. *Genome Biology*, **5**: R12.

Lefebvre, A., Lecroq, T., Dauchel, H. and Alexandre, J. (2003) FORRepeats: detects repeats on entire chromosomes and between genomes. *Bioinformatics,* **19** (3): 319-326.

Li,M., Ma,B., Kisman,D. and Tromp,J. (2004) PatternHumter II: Highly sensitive and fast homology search, *J. Bioinf. and Comput. Biol.*, **2**, 417-439.

Liao,B.Y., Chang,Y.J., Ho,C.M. and Hwang,M.J. (2004) The UniMarker (UM) method for synteny mapping of large genomes. *Bioinformatics*, **20**, 3156-3165.

Lipman,D.J. and Pearson,W.R. (1985) Rapid and sensitive protein similarity searches. *Science*, **227**, 1435-1441.

Ma,B., Tromp,J. and Li,M. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440-445.

Manber,U. and Myers,E. (1993) Suffix arrays: a new method for on-line string matches. *SIAM J. Comput.*, **22**, 935-948.

Nadeau,J.H. and Sankoff,D. (1998) Counting on comparative maps. *Trends Genet.*, **14**,495-501.

Ning,Z., Cox,A.J. and Mullikin,J.C. (2001) SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11**, 1725-1729.

Pevzner,P. and Tesler,G. (2003) Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. USA.*, **100**, 7672-7677.

Schwartz,S. *et al*. (2003) Human-mouse alignments with BLASTZ. *Genome Res.*, **13**, 103-107.

Schuler,G.D. (1997) Sequence mapping by electronic PCR. *Genome Res.*, **7**, 541-550.

Shih,A. and Li,W.H. (2003) GS-aligner: a novel tool for aligning genomic sequences using bit-level operations. *Mol. Biol. Evol.*, **20**, 1299–1309.

Tatusov,R.L. *et al.* (2003) The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, **4**, 41.

Theiβen,G. (2002) Orthology: secret life of genes. *Nature*, **415**, 741.

Ureta-Vidal,A., Ettwiller,L. and Birney,E. (2003) Comparative genomics: genome-wide analysis in metazoan eukaryotes. *Nature Rev. Genet.*, **4**, 251-262.

Vinga,S. and Almeida,J. (2003) Alignment-free sequence comparison - a review. *Bioinformatic*s, **17**, 391-397.

Waterston,R.H. *et al*. (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature*, **420**, 520–562.

Wilson, M.D. *et al*. (2001) Comparative analysis of the gene-dense ACHE/TFR2 region on human chromosome 7q22 with the orthologous region on mouse chromosome 5. *Nucleic Acids Res.*, **29**, 1352–1365.

Zhang,Z., Schwartz,S., Wagner,L. and Miller,W. (2000) A greedy algorithm for aligning DNA sequences. *J. Comput. Biol.*, **7**, 203-214.

** Revision: 20080730*

List of Publications

- Journal papers
  - Liao, Ben-Yang, **Chang, Yu-Jung**, Ho, Jan-Ming, and Hwang, Ming-Jing (2004) The UniMarker (UM) method for synteny mapping of large genomes. *Bioinformatics*, **20**, 3156-3165. (Chang and Liao were joint First Authors in this work.)(**SCI**, IF=5.04)
  - Shih, Jen-Ying, **Chang, Yu-Jung**, Chen, Wun-Hwa (2008) Using GHSOM to construct legal maps for Taiwan's securities and futures markets. *Expert Systems with Applications*, **34**, 850-858. (**SCI**, IF=1.18)
  - Chen, Chih-Ming, Lee, Hahn-Ming, and **Chang, Yu-Jung** (2008) Two novel feature selection approaches for web page classification. *Expert Systems with Applications*, in press. (**SCI**, IF=1.18)
- Conference papers and posters
  - Liao, Ben-Yang, **Chang, Yu-Jung**, Ho, Jan-Ming, and Hwang, Ming-Jing (2003) Human and mouse genome comparison using genome-wide unique sequences. *Poster sessions of 11th International Conference on Intelligent Systems for Molecular Biology*, Brisbane, Australia, June 29 - July 3.
  - Shih, Jen-Ying, **Chang, Yu-Jung**, Chen, Wun-Hwa, Ho, Jan-Ming Ho and Kao, Cheng-Yan (2004) Constructing securities and futures markets legal maps of Taiwan using GHSOM. *Proceedings of. 2nd International Conference on Digital Archive Technologies*, Taipei, Taiwan, March 18-19.
  - Shih, Jen-Ying, **Chang, Yu-Jung** (2006) Constructing knowledge maps of a manager's managerial logic by a text mining approach. *Proceedings of 9th Joint Conf. Info. Sci. (JCIS)*, Kaohsiung, Taiwan, Oct. 8-11. (**EI**)
- Papers in submission
  - **Chang, Yu-Jung**, Kao, Cheng-Yan, Lin, Wen-Dar, Hwang, Ming-Jing, and Ho, Jan-Ming (in submission) Copy number-based seeds for orthology detection in genome comparisons. In submission to *Bioinformatics*.