Pattern Matching with Wildcards and Length Constraints

（带有通配符和长度约束的模式匹配）

Xindong Wu Fei Xie Jipeng Qiang Youxi Wu Dan Guo
Yingling Liu Qingren Wang Jun Gao Xuegang Hu

Science Press
Beijing

Supported by the National Fund for Academic Publication in Science and Technology

Pattern Matching with Wildcards and Length Constraints

(带有通配符和长度约束的模式匹配)

Xindong Wu, Fei Xie, Jipeng Qiang, Youxi Wu, Dan Guo, Yingling Liu, Qingren Wang, Jun Gao, Xuegang Hu

With Contributions from:

Abdullah N. Arslan, Gong Chen, Yuan Fang, Shuai Fu, Dan He, Yu He, Xiaoli Hong, He Jiang, Lei Guo, Peipei Li, Yan Li, Yawei Liu, Zhenyu Lu, Fan Min, Weidong Tian, Haiping Wang, Gongqing Wu, Xingquan Zhu

Science Press
Beijing

内 容 简 介

本书围绕带有通配符和长度约束的模式匹配这一科学前沿问题进行算法设计与分析，系统、深入地阐述带有通配符和长度约束模式匹配的问题描述、算法设计、理论和实验结果分析。主要内容包括：①形式化描述带有通配符和长度约束的模式匹配问题，设计基于在线处理方式的精确匹配算法，并对其扩展以解决近似模式匹配问题；②提出基于位并行技术的模式匹配算法，提高搜索效率；③提出“网树”结构，计算指数量级模式匹配解的个数；④提出“子网树”结构，解决一般通配符约束问题；⑤设计启发式算法，提高带有通配符约束模式匹配解的完备性。

本书可作为高等院校计算机科学与技术等专业研究生和高年级本科生的教材，也可作为数据挖掘、模式识别等相关领域研究人员的参考书。

图书在版编目（CIP）数据

带有通配符和长度约束的模式匹配= Pattern Matching with Wildcards and Length Constraints：英文/吴信东等著. —北京：科学出版社，2016
ISBN 978-7-03-047410-0

Ⅰ. ①带… Ⅱ. ①吴… Ⅲ. ①计算机网络-算法理论-英文
Ⅳ. ①TP393.02

中国版本图书馆 CIP 数据核字（2016）第 036351 号

责任编辑：裴 育 / 责任校对：胡小洁
责任印制：张 倩 / 封面设计：蓝正设计

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京通州皇家印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2016 年 3 月第 一 版 开本：720 × 1000 1/16
2016 年 3 月第一次印刷 印张：12 1/2
字数：328 000

定价：98.00 元
（如有印装质量问题，我社负责调换）

Preface

Pattern matching is a fundamental problem in computing. Along with the rapid development of information technology, it has been widely used in many fields, such as information retrieval, data stream mining, network intrusion detection, bioinformatics, and so on. In recent years, pattern matching with wildcard constraints as a new research area has attracted increasing attention. When a pattern contains wildcards, the matching problem becomes more complicated. Meanwhile, we also introduce the one-off condition, which requires that every character in the given sequence can be matched at most once. These two conditions, gap constraints and the one-off condition make the matching problem relevant and challenging for real-world applications, because gap requirements naturally consider the genetic insertion/deletion/mutation during the evolution and the one-off condition avoids counting the same characters for multiple times.

The objective of this book is to present state-of-the-art approaches for dealing with pattern matching with wildcards and length constraints. For each topic within this field, we will present the most representative algorithms, with sufficient details so that readers can gain a good knowledge without the necessity of referring to additional materials. To begin with, we introduce two online algorithms that focus on wildcards and length constraints under the one-off condition. As no polynomial solution currently exists to find the maximum number of non-overlapping occurrences of a given subsequence, and we regard it is an NP-hard problem, we present heuristic algorithms to resolve the conflict between different occurrences in the whole candidate solution space. Also, we study three related problems, pattern matching with arbitrary-length wildcards, pattern matching with general gaps and approximate pattern matching, each of which becomes more complicated than pattern matching with wildcards.

The book has two characteristics: ① we focus on solving specific problems of pattern matching with wildcards and length constraints, and ② we combine theory with practical examples. The book is suitable for graduate students and researchers who want to understand pattern matching with wildcards and length constraints in depth.

The research presented in this book has been supported by the National 973 Program of China under grant 2013CB329604, the US National Science Foundation (NSF) under grants CCF-0514819 and CCF-0905337, the National Natural Science Foundation of China (NSFC) under grants 60828005 and 61229301, and the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grant IRT13059. This book is also supported by the National Fund for Academic Publication in Science and Technology.

We are deeply grateful to the original authors of the different book chapters for their innovative work and the people who contributed to the revisions of the book. They are Abdullah N. Arslan, Gong Chen, Dan Guo, Yuan Fang, Shuai Fu, Dan He, Yu He, Xiaoli Hong, He Jiang, Lei Guo, Peipei Li, Yan Li, Yawei Liu, Yingling Liu, Zhenyu Lu, Fan Min, Jipeng Qiang, Weidong Tian, Haiping Wang, Qingren Wang, Gongqing Wu, Xindong Wu, Youxi Wu, Fei Xie, Xingquan Zhu.

Authors
May 2015

Contents

Chapter 1
Introduction

1.1 The Aim and Focus of This Book

In recent years, pattern matching with wildcards has become a challenging issue in many application fields, such as bioinformatics, information retrieval, and pattern mining. In bioinformatics, for example, the DNA subsequence TATA is a common promoter that often occurs after the sequence CAATCT within 30—50 wildcards[1, 2]. In information retrieval, pattern combinations with useless or "don't care" character intervals are more meaningful than the general patterns.

Given a pattern P and a sequence S, some classical matching algorithms return the positions of P in S. This matching can be exact or approximate. However, if wildcards are considered in the process of matching, this question will become complicated. The problem of pattern matching with wildcards was first studied by Fischer and Paterson[3]. A wildcard ε can match any character in a given alphabet. There are three types of patterns with wildcards: "a fixed number of wildcards", "allowing an unbounded number of wildcards", and "the number range of wildcards can be specified separately, namely flexible wildcards". As the first two types limit flexibilities for the user's queries, pattern matching with flexible wildcards draws more researchers' attention.

It is very difficult to solve the problem of pattern matching with flexible wildcards. Due to the flexibility of wildcards, not only the number of all matches is exponential with respect to the maximal gap size and the pattern length, but also the matching positions are hard to choose. We first studied the problem of pattern matching with wildcards and length constraints[4]. Both local and global length constraints were integrated into the problem for effective pattern matching. Meanwhile, to avoid useless information, the one-off condition was introduced that requires each character to be used at most once for pattern matching in a sequence. Considering some special research backgrounds, we also present the algorithms of computing the number of all matches of a pattern in a sequence.

We performed a systematic investigation on the pattern matching problem with wildcards aiming to design efficient matching algorithms, analyze the computational complexity, and apply them to practical applications[4-20]. The problem has also been extended to sequential pattern mining to discover all frequent sequential patterns with wildcards from a given sequence[21-25]. To keep the integrity of problem and make it easy for readers to understand, we focus on pattern matching with wildcards and length constraints in this book. To begin with, we present an efficient SAIL algorithm that returns each pattern occurrence in an online manner[4]. The SAIL algorithm is also extended to find approximate pattern occurrences where some errors may be allowed[7].

To improve time efficiency, bit-parallelism technologies are utilized to simulate the match process[5, 8, 13, 14]. The BPBM algorithm adopts an effective transition window mechanism with two nondeterministic finite automatons (NFAs) to accelerate the scan process[5, 8]. The BWW and FWW algorithms handle multiple-pattern matching simultaneously[13]. We also design efficient bit-parallelism algorithms to support pattern matching with arbitrary-length wildcards where the number range of wildcards may vary between two integer bounds or from a lower integer bound to infinity[14].

Since the number of pattern occurrences with wildcards is exponential with respect to the maximal gap flexibility and the pattern length, counting matches one by one is computationally impossible. Therefore, we present the PAIG algorithm and its two enhanced versions to compute the number of occurrences of a pattern[12]. Furthermore, an efficient algorithm named NAMEIC is presented based on a new nonlinear data structure Nettree[19].

To solve the problem of pattern matching with general gaps where the gaps can possibly be negative, the Subnettree structure is introduced based on Nettree. Two algorithms named SETS[16] and SETA[17] are respectively designed for strict and approximate pattern matching with general gaps.

Finally, two heuristic algorithms are described to improve the completeness of pattern matching algorithms with wildcards. The WOW algorithm introduces a weighted centralization measure based on Nettrees to get the optimal solution[6]. In the RBCT algorithm, a CluTree structure that is a cluster of trees with red and black nodes is designed to find more occurrences[10].

1.2 Overview

Chapter 2—SAIL and SAIL-APPROX Algorithms: Pattern matching with both gap constraints and one-off condition is a challenging topic. We first proposed SAIL to solve the problem that returns each occurrence of a pattern in an online manner[4]. In this pattern matching problem, user can specify the number of wildcards between each two consecutive characters of a pattern P and the length of each matching substring in the sequence S. SAIL returns each matched substring of P in S as soon as it appears in S in an $O(n+kG_M mW)$ time with an $O(G_M m)$ space overhead, where n is the length of S, k is the frequency of P's last character occurring in S, G_M is the user-specified maximum length for each matching substring, m is the length of P, and W is the maximal gap flexibility.

We extend the SAIL algorithm to deal with approximate pattern matching, where some errors may be allowed for the matching of a pattern[7]. Given a pattern P and a sequence S, SAIL-APPROX is to find an approximate occurrence of P in S such that it satisfies both local and global constraints and the distance between this approximate occurrence and P is less than a user defined threshold.

Chapter 3—Pattern Matching Based on Bit-parallelism: In this chapter, a series of bit-parallelism algorithms are designed to accelerate the search process. The BPBM algorithm adopts an effective transition window mechanism with two nondeterministic finite state automatons (NFAs) to drop a useless scan window[5, 8]. It identifies gap constraints automatically and just scans once to output occurrences with exact match positions. Meanwhile, BPBM inherits the advantage of BM algorithm[26] to skip some characters in a sequence. Theoretical analysis and experimental results show that the BPBM algorithm is more competitive on the search process. It also has better stability that decreases operation costs with the size increase of a sequence alphabet or the length of a pattern.

To handle multiple-pattern matching simultaneously, the BWW and FWW algorithms are designed based on a nondeterministic finite state automaton and bit-parallelism technology[13]. Experimental results show that they are more efficient than other peers, and are especially dozens of times faster on average than SAIL. FWW is twice faster than BPBM in DNA sequences, when the length of multiple patterns increases.

The number range of wildcards may vary between two integer bounds or from a lower integer bound to infinity. That is, the gap size can be specified by the user or arbitrarily large. Therefore, the MOTW and MWTO algorithms are presented to solve the problem of pattern matching with arbitrary-length wildcards[14]. MOTW first searches for occurrences of a pattern from left to right, and then decides the location of a window. MWTO first locates the start position of a window, and then decides whether an occurrence exists within the window. Experimental results validate the correctness of the proposed algorithms, and show that MOTW outperforms MWOT in the case of pattern matching without length constraints while MWOT outperforms MOTW in the case of length constraints.

Chapter 4—Efficient Algorithms for Counting the Number of Occurrences: In this chapter, we present two efficient algorithms, named PAIG[12] and NAMEIC[19], to compute the number of all matches that is exponential with respect to maximal gap flexibility and pattern length. The PAIG (pattern matching with wildcards and length constraints) algorithm has three versions: PAIG-S (simple), PAIG-RS (reduced space), and PAIG-RST (reduces space and time). For the PAIG-RST version, the time complexity is $O(nm^2W^2)$, where n is the sequence length, m is the pattern length, and W is the maximal gap flexibility. The space complexity is $O(mW)$. PAIG meets global length constraints with a minor modification. It can also be employed to compute the pattern frequency, which is the focus of pattern mining.

The NAMEIC algorithm is based on a new data structure Nettree that is a kind of directed acyclic graph (DAG) with edge labels. A Nettree is different from a regular tree in that a node may have more than one parent. The time complexity of NAMEIC is $O(Wmn)$, where n is the length of sequence S, m is length of pattern P, and W is the maximal gap flexibility.

Chapter 5—Pattern Matching with General Gaps: In this chapter, the problem of pattern matching with general gaps is studied where the gaps can be negative. Two algorithms, named SETS[16] and SETA[17], are respectively designed for strict and approximate pattern matching. In the SETS algorithm, an instance of SPANGLO (Strict Pattern mAtching with geNeral Gaps and Length cOnstraints) is transformed into an exponential number of occurrences with non-negative gaps in the worst case. The space and time complexities of SETS are $O(mG_MW)$ and $O(G_MWm^2n)$ respectively, where n, m, G_M and W are the sequence length, the pattern length, the maximal length of the occurrence and the maximal gap of the pattern respectively.

In the SEAT algorithm, a SAP (strict approximate pattern matching with general gaps and length constraints) instance is transformed into an exponential amount of the exact pattern matching with general gap instances. The space and time complexities of SETA are $O(mG_MWd)$ and $O(G_MWm^2nd)$, respectively, where m, G_M, W, and d are the length of pattern P, the maximal length constraint, the maximal gap length of pattern P and the approximate threshold.

Chapter 6—WOW Algorithm: As online algorithms are not complete algorithms for pattern matching with wildcards, an offline algorithm WOW is designed that uses a weighted centralization measure based on Nettree to represent occurrences of a pattern[6]. The idea of WOW is that the smaller the occurrence's weighted centralization is, the smaller the overlap with other occurrences is to get the optimal solution. Experimental results demonstrate that WOW can obtain more occurrences than SAIL and BPBM in most cases.

Chapter 7—RBCT Algorithm: The traditional left-most strategy leads to incomplete final matching results when selecting among multiple candidate matching positions. Therefore, a new matching algorithm RBCT is proposed based on a new data structure CluTree[10]. A CluTree is composed of a cluster of trees with red and black nodes according to a pattern P and a sequence S. The RBCT algorithm uses the sharing degree, correlation degree and mixed information entropy of each node in the CluTree for path selection and dynamic pruning. Theoretical analysis and experimental results show that the RBCT algorithm outperforms other peers in retrieval precision and matching efficiency.

1.3 Basic Concepts

Definition 1.1 Let Σ be an **alphabet**. $S=s_0s_1\cdots s_{n-1}\in\Sigma^*$ is called a string or sequence of Σ, where $n = |S|$ is the length of S.

For example, $\Sigma=\{A, C, G, T\}$ is the alphabet of DNA sequences. RNA has a slightly different alphabet $\{A, U, C, G\}$, and a protein has a larger alphabet with 20 characters.

Definition 1.2 A **wildcard** (denoted as ε) is a special symbol that can match any character in Σ. A gap $\varepsilon(N, M)$ is a sequence of wildcards with a minimal size N and a maximal size M. $W=M-N+1$ is the gap flexibility of $\varepsilon(N, M)$.

$\varepsilon(N, M)$ is often represented by $[N, M]$ for brevity.

Definition 1.3 A **pattern** is a tuple $P = (p, \varepsilon)$, where $p = p_0p_1\cdots p_{m-1}$ is a sequence of Σ, and $\varepsilon = \varepsilon_0\varepsilon_1\cdots\varepsilon_{m-2}$ is a sequence of gaps. $\varepsilon_i = \varepsilon(N_i, M_i)$ $(0\leqslant i\leqslant m-2)$ is the gap between p_i and p_{i+1}. $m = |p|$ is the length of P and denoted as $|P|$.

A pattern can also be viewed as a mixture of characters and gaps, and expressed as $P=p_0\varepsilon_0(N_0, M_0)p_1\cdots\varepsilon_{m-2}(N_{m-2}, M_{m-2})p_{m-1}$. Note that $|p| = |\varepsilon|+1$ is mandatory. However, this requirement does not limit the generality of the pattern. If there is no wildcard between two adjacent characters in P, $g(0, 0)$ will be used.

Throughout the book, we use notations as follows: $p[j] = p_j$ and $s[i] = s_i$ indicating characters of the pattern and sequence, respectively.

Definition 1.4 Given a pattern P with length m, $P_i^j=p_i\varepsilon_i(N_i, M_i)p_{i+1}\cdots\varepsilon_{j-1}(N_{j-1}, M_{j-1})\,p_j$ $(0\leqslant i\leqslant j\leqslant m-1)$ is called a **subpattern** of P.

We are interested in two kinds of subpatterns defined as follows.

Definition 1.5 Given a pattern P with length m, $P_i=p_0\varepsilon_0(N_0, M_0)p_1\cdots\varepsilon_{i-1}(N_{i-1}, M_{i-1})\,p_i$ $(i\leqslant m-1)$ is called a **prefix** of P.

Specifically, $P_0 = p_0$ and $P_{m-1} = P$.

Definition 1.6 Given a pattern P with length m, $P_i^{i+1} = p_i\varepsilon_i(N_i, M_i)p_{i+1}$ $(0 \leqslant i \leqslant m-2)$ is called a **pattern component** of P.

Pattern components are the shortest subpatterns with both characters and gaps.

Here, $H = [G_N, G_M]$ $(G_N \leqslant G_M)$ is used to represent the global length constraints of P with flexible gaps. The length constraints satisfy $m + \sum_{i=0}^{m-2} N_i \leqslant G_N \leqslant G_M \leqslant m + \sum_{i=0}^{m-2} M_i$.

Definition 1.7 An occurrence $C=(c_0, \cdots, c_i, \cdots, c_{m-1})$ is a list of position indices of P in S. If there exists an offset sequence $C=(c_0, \cdots, c_i, \cdots, c_{m-1})$ $(0 \leqslant c_i \leqslant n-1, 0 \leqslant i \leqslant m-1)$ such that

(1) $s_{c_i} = p_i$;

(2) $N_i \leqslant c_{i+1} - c_i - 1 \leqslant M_i$.

C is an **occurrence** of P in S for all $0 \leqslant i \leqslant m-1$. A set of occurrences $C_1, C_2, \cdots, C_t$ constitute an occurrence set U where t is the number of occurrences, and named matching number in this book.

Definition 1.8 Considering the **global length constraints** $[G_N, G_M]$, if $C = (c_0, \cdots, c_i, \cdots, c_{m-1})$ is an occurrence of P in S, it meets an additional condition: $G_N \leqslant c_m - c_0 + 1 \leqslant G_M$.

Definition 1.9 Suppose $occ_1=(k_0, \cdots, k_{m-1})$ and $occ_2=(h_0, \cdots, h_{m-1})$ are two occurrences of P. If $k_p \neq h_q$, for all $0 \leqslant p, q \leqslant m-1$, we say occ_1 and occ_2 satisfy the **one-off condition**. The one-off condition means that each character can be used at most once in all occurrences of a pattern in the sequence.

Example 1.1 Assume $S=ACACTTCA$ and $P=A\varepsilon(0, 2)C$, (0, 1), (0, 3), (2, 3) are three occurrences of P. Occurrences (0, 1) and (2, 3) satisfy the one-off condition while (0, 1) and (0, 3) do not because they share the same position 0.

Throughout this book, the problem of pattern matching with gap constraints is called the PMG problem, and the problem of pattern matching with both gap constraints and the one-off condition is called the PMGO problem.

For reference, the major notations used in this book are listed in Table 1.1.

Table 1.1 Notations

Symbol	Definition
S	A subject sequence
N	Length of S, namely $n=\|S\|$
$P=(p, \varepsilon)$	A pattern

Continued

Symbol	Definition
M	Length of P without considering gaps, namely $m=\|P\|$
N_i	The minimum local length constraint between P_i and P_{i+1}
M_i	The maximum local length constraint between P_i and P_{i+1}
G_N	The minimum global length constraint
G_M	The maximum global length constraint
W_i	Flexibility of ε_i; $W_i=M_i-N_i+1$
W	$\max\limits_{0\leqslant i\leqslant m-2} W_i$
L	The maximum length sum of P; $L=m+\sum\limits_{i=0}^{m-2} M_i$