带通配符的模式匹配问题及其解空间特征分析

项泰宁 郭 丹 王海平 胡学钢
(合肥工业大学计算机与信息学院 合肥 230009)

摘 要 随着生物信息学、信息检索等领域的发展,带有通配符和长度约束的模式匹配问题引起了广泛关注。该问题扩展了精确模式匹配问题,使匹配更加灵活,同时也增加了匹配的复杂性,极大地提高了非线性匹配算法的复杂度。求解该问题的匹配算法的效率与问题的解空间密切相关,而目前针对该问题的解空间及其特征尚缺乏系统的研究。鉴于此,描述了该问题的解空间,并分析了解空间的可分性。之后,提出解空间划分算法SPLIT,并分析了SPLIT的时间复杂性。实验部分以3个匹配算法为对照,在真实DNA数据集下,使用了5109组模式。实验结果表明,SPLIT不影响匹配解的结构,且可以有效降低非线性匹配算法的时间消耗。

关键词 解空间,分割,模式匹配,通配符
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.09.051

Characteristic Analysis of Pattern Matching with Wildcards Problem and its Solution Space

XIANG Tai-ning GUO Dan WANG Hai-ping HU Xue-gang
(School of Computer Science and Information, Hefei University of Technology, Hefei 230009, China)

Abstract With the developments in bioinformatics and information retrieval, pattern matching with wildcards and length constraints has attracted wild attention. This problem extends the exact pattern matching, so it brings more flexibility as well as complexity. Meanwhile, the time complexity of non-linear algorithms is greatly increased. The solution space makes great difference to the efficiency of matching algorithms, but the characteristics of the problem solution space is still lack of systematic research. Therefore, the problem solution space was presented, and the divisibility of the space was analyzed. Afterwards, a solution space partitioning algorithm, named SPLIT, was proposed, and its time complexity was also illustrated. In our experiments, three pattern matching algorithms were used as baselines running on real DNA data sets with 5109 sets of patterns. Our empirical results verify that SPLIT can effectively reduce the time consumption of non-linear matching algorithms without influencing the matching solution.

Keywords Solution space, Partitioning, Pattern matching, Wildcard

1 引言

目前,带有通配符和长度约束的模式匹配问题(PMWL)[1]受到越来越多的关注。PMWL问题可描述为,给定字母表Σ、文本T和模式P,其中,模式任意两字符间带有通配符ϕ,且ϕ可指代区间约束范围内的字符串$S(S\in\Sigma^*)$,对S长度的限制即为长度约束,其任务是找出P在T中的匹配解。PMWL问题在生物信息学[2]、模式挖掘[3,4]、信息检索与过滤[5]中有广泛的实际应用。例如,TATA框有助于研究者从数以亿计的DNA序列中快速定位内含子的起始位置,它常常出现在CAATCT序列下游30～50bp处,因此可以表示为…CAATCTϕ[30,50]TATA…[6],以提高序列特异性。

PMWL问题定义在增加灵活性的同时也增加了求解的困难。在已有研究工作中,Fischer和Paterson首次将通配符引入到模式匹配问题[7]中,随后文献[1,8-10]研究了带有不同约束的通配符的匹配问题。其中,PMWL问题的任务是找出模式在文本中的最大匹配解集[1]。针对该问题,大多数匹配算法采用贪心策略得到近似最优解。具有代表性的有:Chen等人提出的SAIL算法[1]使用滑动窗口结构和left-most贪心匹配策略得到在线情况下的完备解,但离线情况下不完备;Guo等人提出的BPBM算法用位并行技术处理PMWL问题[11],降低了时间复杂度,但贪心匹配策略与SAIL相同;武等人的SBO启发式算法使用了网树结构[12],采用最小相关度和最右双亲的贪心策略,增加匹配数目,其时间复杂度与文本长度呈非线性关系;侯等人在PMWL问题中对文本构建后缀树,该索引结构构建所需时间与文本长度呈非线性关系[13]。

上述算法本质上是在解的数目和时间效率上折中,匹配策略决定了算法得到解的数目,而合理的数据结构能在不影响匹配解的前提下提高时间效率。对组合优化的问题,可从

到稿日期:2013-10-23 返修日期:2014-02-22 本文受国家自然科学基金:港澳学者合作研究基金项目(61229301),国家自然科学基金(61305062),博士后面上基金项目(2012M511403),安徽省自然科学基金(1308085QF102)资助。
项泰宁(1988－),男,硕士生,主要研究方向为数据挖掘,E-mail:Xiang.taining@163.com;郭 丹(1983－),女,讲师,主要研究方向为人工智能;王海平(1986－),男,博士生,主要研究方向为数据挖掘;胡学钢(1961－),男,教授,博士生导师,主要研究方向为人工智能。

解空间角度进行算法设计和分析[14]。此外，解空间已在点模式[15]、序列模式[16]等匹配问题中得到广泛研究。然而尚未有工作对PMWL问题的解空间进行充分研究。因此，本文以PMWL问题的解空间为研究核心，一方面系统研究了解空间的特征性质；另一方面在不影响匹配解的前提下优化已有算法的数据结构，以提高匹配效率。

以此为基础，本文工作如下：(1)给出了PMWL解空间的定义；(2)提出一种PMWL解空间划分的算法SPLIT，并给出了SPLIT的时间复杂度分析。实验部分以SBO、PAIG和SAIL算法为对照，列举了9组代表性模式和5100组不同特征下的随机模式，文本为真实DNA序列和人工随机序列，在不同模式特征、文本长度、子空间数目下进行了比较。结果表明，SPLIT能对PMWL解空间完备划分，同时可以有效降低非线性算法的时间复杂度。

本文第2节给出PMWL问题的解空间描述；第3节提出一种解空间划分算法SPLIT；第4节给出实验结果和分析；最后总结了全文。

2 问题描述

2.1 PMWL问题解空间定义

定义1 带有通配符和长度约束的模式匹配问题(Pattern Matching with Wildcards and Length constraints, PMWL)可以描述为：

$$Q=\{T,P,\Sigma,LC,RC,H,OCC\} \tag{1}$$

其中，$T=\{(T_1T_2,\cdots,T_n)\mid T_i\in\Sigma,1\leqslant i\leqslant n\}$为文本，$P=\{(p_1p_2\cdots p_m)\mid p_i\in\Sigma,1\leqslant i\leqslant m\}$为模式，$\Sigma$为字母表，$LC$为模式相邻字符的位置约束，$RC$为匹配解之间的约束，$H$为解空间，$OCC$为匹配结果。其中$h_i$为$p[i]$在$T$中匹配位置的集合，即$h_i=\{k\mid p_i=t_k,1\leqslant k\leqslant n\}$。本文认为，匹配问题的解空间，本质上是模式各个字符匹配位置集合的笛卡尔积，于是得到：

$$H=\{h_1\times h_2\times\cdots\times h_m\}=\{(\alpha_1,\alpha_2,\cdots,\alpha_m)\mid\alpha_i\in h_i,1\leqslant i\leqslant m\} \tag{2}$$

模式相邻字符的匹配位置存在长度约束。本文用LC(Length Constraint)表示。形式化描述为：

$$LC=\{(\alpha_1,\alpha_2,\cdots,\alpha_m)\mid lc(\alpha_i,\alpha_{i+1}),1\leqslant i<m\} \tag{3}$$

例如，在模式$P=A\text{¢}[1,2]T\text{¢}[2,4]T$中，$1\leqslant lc(\alpha_1,\alpha_2)\leqslant 2,2\leqslant lc(\alpha_2,\alpha_3)\leqslant 4$。

此外，匹配解之间也存在约束，用RC(Relation Constraint)表示。描述为：

$$RC(A_1,A_2,\cdots,A_N)=\{\forall i,j,R_{i,j}(A_i,A_j),1\leqslant i,j\leqslant N,i\neq j\} \tag{4}$$

其中，A是解空间中的一组匹配解，$R_{i,j}$是匹配解A_i和A_j之间的关系，可描述为：

$$R_{i,j}=\{A_i,A_j\mid A_i=(\alpha_1^i,\alpha_2^i,\cdots,\alpha_m^i)\in H,A_j=(\alpha_1^j,\alpha_2^j,\cdots,\alpha_m^j)\in H,r(A_i,A_j)\}$$

例如，在one-off条件中，$r(A_i,A_j)$约束了任意两个解A_i和A_j之间不同解使用相同的匹配位置[1]，如$A_i=\{1,3,5\}$，$A_j=\{2,4,6\}$满足条件，而$A_i=\{1,3,5\}$，$A_j=\{2,5,7\}$不满足条件。

在满足上述约束条件的情况下，问题将输出匹配结果，用OCC表示：

$$OCC=\{A_1,A_2,\cdots,A_N\mid A_k\in H,A_k\in LC,1\leqslant k\leqslant N,RC(A_1,A_1,\cdots,A_N)\} \tag{5}$$

定义2 one-off条件是指同一个匹配位置至多只能用在一个匹配解中。两个匹配解之间的若满足one-off条件，则认为这两个解存在OFF关系：

$$OFF_{i,j}=\{A_i,A_j\mid A_i=(\alpha_1^i,\alpha_2^i,\cdots,\alpha_m^i)\in H,A_j=(\alpha_1^j,\alpha_2^j,\cdots,\alpha_m^j)\in H,\forall s,t,\alpha_s^i\neq\alpha_t^j,1\leqslant s,t\leqslant m)\}$$

则：

$$RC_{OFF}(A_1,\cdots,A_N)=\{\forall i,j,OFF_{i,j}(A_i,A_j),1\leqslant i,j\leqslant N,i\neq j\} \tag{6}$$

例如，匹配解$\{0,3,5\}$和$\{1,3,6\}$不满足one-off条件，因为有共同的匹配位置3。因此，OFF关系可视为RC关系的实例。

由以上定义可知，PMWL问题的解空间是模式各字符在文本中匹配位置的笛卡尔积，且受到LC和RC的条件约束，定义为：

$$OCC_{PMWL}=\{A_1,\cdots,A_N\mid A_k\in H,A_k\in LC,1\leqslant k\leqslant N,RC_{OFF}(A_1,\cdots,A_N)\} \tag{7}$$

2.2 PMWL问题解空间的相关性质

定义3 匹配问题的解空间划分可描述为：如果存在集合$U_H=\{H_i\mid 1\leqslant i\leqslant B,H_i\subseteq H\text{ 且}\bigcup H_i=H\}$，则认为$H_i$是$H$的子空间，$B$是子空间的数目，$U_H$是子空间的集合。

假设输出的一组匹配解是：

$$OCC=\{A_1,\cdots,A_N\mid A_k\in H,A_k\in LC,1\leqslant k\leqslant N,RC(A_1,\cdots,A_N)\}$$

其中，解$A_i=(\alpha_1^i,\alpha_2^i,\cdots,\alpha_m^i)$是字符出现位置的序列，因此可对解集合$A_1,A_2,\cdots,A_N$使用字典排序，得到解与解之间的关系。

定义4 任意给定解A_i和A_j，若满足$\alpha_1^i\leqslant\alpha_1^j\leqslant\alpha_m^i\leqslant\alpha_m^j$，则$A_i$和$A_j$存在嵌套关系ER(Embed Relation)。由此N组解之间的嵌套关系为：

$$ER(A_1,A_1,\cdots,A_N):\forall i\text{ s.t. }W_i\cap W_{i+1}\neq\varnothing,1\leqslant i<N \tag{8}$$

其中，$W_i=\{[\alpha_1^i,\alpha_m^i]\mid A_i=(\alpha_1^i,\alpha_2^i,\cdots,\alpha_k^i,\cdots,\alpha_m^i)\in H\}$。

传统匹配问题的解空间较容易划分，满足$U_i=W_i$，子空间的数目等于匹配解的数目。传统匹配问题的滑动窗口即为划分解空间的数据结构。然而，在PMWL问题中匹配解之间的嵌套关系影响了子空间的划分。例如SAIL算法[1]的数据结构为滑动窗口(见图1)，窗口的区间由匹配策略决定，在离线情况下可能丢失完备解。

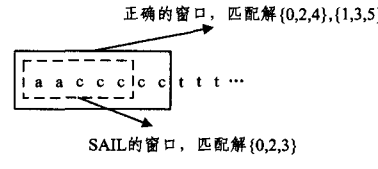

图1 SAIL算法的滑动窗口

定义5 解空间中的分割点可描述为$x\notin\bigcup U_i,1\leqslant i<N$。在分割点位置划分解空间不影响原有匹配解的结构，即保持划分前后匹配数不变。特别地，文本的起始位置$T[1]$和终止位置$T[n]$也可被视为分割点。例如，匹配解$\{0,3,5\}$和

{2,4,6}存在嵌套关系，匹配解{0,3,5}和{7,9,11}不存在嵌套关系，可选取位置6作为分割点。

定义6 PMWL解空间划分是指：给定输入T、P、Σ、LC、RC，输出$U_{cut}=\{x_i \mid 1\leqslant i\leqslant B, x_i$ 是分割点$\}$构成分割点的集合，其中B是分割点的数目。考虑到文本两端的位置可作为分割点，有$B>1$。

由此，文本T可被划分成

$$U_{subT}=\{subT_i \mid subT_i=T_{x_i}\cdots T_{x_{i+1}}, x_i\in U_{cut}, 1\leqslant i\leqslant B\} \tag{9}$$

其中，子空间数目为$B-1$。

2.3 PMWL问题的匹配完备性

由上文知，PMWL问题的解为OCC，模长为$|OCC|$，因为该集合是有限集，一定有最大值MAX。对于任意满足PMWL问题约束的解，当解的个数等于MAX时，称其为完备解。

定义7 问题的完备解集定义如下：

$$U_{complete}=\{OCC \parallel OCC|=\mathrm{MAX}\} \tag{10}$$

问题的不完备解集定义如下：

$$U_{incomplete}=\{OCC \parallel OCC|<\mathrm{MAX}\} \tag{11}$$

定义$\varepsilon=|OCC|/\mathrm{MAX}$为近似比，显然$\varepsilon\leqslant 1$。

定义8 在PMWL问题中，如果存在匹配算法A，在任意给定输入的情况下，所得到的输出OCC满足$OCC\in U_{complete}$，则认为该匹配算法完备和PMWL问题可解。特别地，当算法A是多项式时间复杂度时，则认为PMWL问题多项式可解。

针对不带one-off条件的PMWL问题，文献[17]一方面分析了该问题的匹配数呈指数级增长，认为不存在多项式算法能求得PMWL完备匹配位置；另一方面给出了仅求解该问题完备匹配数的算法。然而，带one-off条件的PMWL问题是否在多项式时间内可解仍是开放性问题[1]。

性质1 PMWL问题中，子空间匹配数满足加和性。

设解空间H，$U_H=\{H_i \mid 1\leqslant i\leqslant B, H_i\subseteq H$ 且 $\bigcup H_i=H\}$为完备划分下子空间的集合，则有$|H|=\sum_{i=1}^{B}|H_i|$。由定义3和定义6可知，子空间没有交集，不同子空间的匹配解没有位置相关性，因此子空间满足加和性。

定理1 在PMWL问题中，若解空间不完备，则一定存在一个子空间不完备。

证明：假设所有子空间完备，由性质1知子空间具有加和性，因此总体解空间完备，与题设矛盾，因此一定存在某个子空间不完备。证毕。

3 PMWL问题的解空间划分算法

PMWL问题中影响解空间划分的关键因素是相邻匹配解之间的嵌套关系ER。本节将提出一种面向PMWL问题的解空间划分算法SPLIT。首先将结合实例给出SPLIT的算法描述，并给出相关分析。

3.1 SPLIT算法描述

在匹配解均匀分布的假设前提下，PMWL解空间大小与文本长度呈线性关系。然而，为得到更高的解的质量，匹配算法往往需多次扫描文本，使其时间复杂度与文本长度呈非线性关系，如SBO[9]。因此，本算法的设计动机是通过均匀划分文本，一方面可以将解空间均匀划分，另一方面可以降低非线性匹配算法的时间复杂度。此外，需保证划分的完备性，从而能快速找出离初始分割位置最近的分割点，且不影响原有匹配解的结构。

3.1.1 SPLIT算法流程

算法1 解空间划分算法SPLIT

输入：文本T，模式P，字母表Σ，初始分割位置initPost

输出：分割点cutPost

流程：

```
1. cutPost←initPost
2. flag←false
3. begin←initPost
4. for i←initPost to initPost+(maxLen−1) do
5.     if P[m−1]=T[i] then
6.         end ←i
7.         if Forward(end,begin) then
8.             SearchPEnd (begin,end,Q)
9.             cutPost←min(begin,cutPost)
10.            flag← TRUE
11.            break
12. if not flag then
13.     return initPost
14. if flag then
15.     if Q. empty()
16.         return begin
17.     while not Q. empty() do
18.         end ←Q. front()
19.         Q. pop()
20.         if Forward(end,begin) then
21.             SearchPEnd (begin,end,Q)
22.             cutPost←min(begin,cutPost)
23.  return cutPost
```

该算法描述如图2所示。

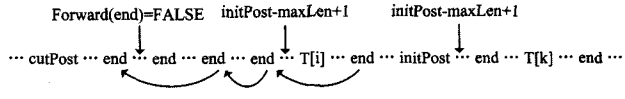

图2 SPLIT算法描述

算法2 局部空间匹配解搜索算法Forward[1]

输入：尾字符位置end

输出：TRUE或FALSE

流程：

1. 通过end和模式最大长度确定滑动窗口大小；
2. 利用通配符的局部约束标记窗口中潜在的匹配解，若没有任何匹配解满足条件，返回FALSE；
3. 将窗口起始位置赋给begin，返回TRUE。

算法3 记录滑动窗口中模式尾字符匹配位置算法SearchPEnd

输入：区间上下限begin，end，字符队列Q

输出：在[begin，end)区间上，将匹配P[m−1]的位置入队。

流程：

```
1. for i←begin to end −1 do
2.     if P[m−1]=T[i] and T[i]从未进入队列 Q then
3.     Q. push(T[i])
```

3.1.2 SPLIT 算法实例

给定文本 T=atcattcagtactgcagaccag，模式 P=a$\not\subset$[0,3]c，全局长度约束 $minLen$=2，$maxLen$=5。

已知初始分割位置 $initPost$=14，P[0]=a，P[m−1]=c。

(1)由全局约束，算法将在[14,18]上搜索尾字符“c”。其中，T[14]=c。调用 *Forward*(14)，返回 TRUE，并计算得到 $begin$=10。将“14”放入队列 Q，并用 *usedEndPost* 记录“14”已入队，同时更新 $cutPost$=min(10,16)=10。

(2)取队列 Q 首元素位置“14”，则 end=14。又由 $begin$=10，算法调用 *SearchPEnd*(10,14,Q)，即在[10,13]范围内查找未入队的尾字符 c。发现位置“11”满足条件，进入 Q。队首元素“14”出队。

(3)由于 Q 不为空，算法进入主函数步骤 17 的循环。取队首元素位置“11”，调用 *Forward*(11)，返回 TRUE，通过计算得到 $begin$=7。将“11”放入队列 Q，并用 *usedEndPost* 记录“11”已入队，同时更新 $cutPost$= min(7,11)=7。

(4)取队首元素位置“11”，则 end=11。又由 $begin$=7，算法调用 *SearchPEnd*(7,11,Q)，在[7,11]范围内查找未入队的尾字符 c。未找到满足条件的位置。队首元素 11 出队。

(5) 队列为空，循环停止。算法最终输出 $cutPost$=7。因此，SPLIT 可在位置“6”与位置“7”之间分割文本串。

3.2 SPLIT 算法分析

3.2.1 SPLIT 时间复杂度分析

根据文献[1]，Forward 的时间复杂度是 O(mg)，其中 m 是 P 的模长，l 是 P 的最大长度，g 是 P 中通配符跨度。若 Forward 返回 True，则算法调用 SearchPEnd，其作用是在[$begin$,end]上搜索 P 的尾字符，其中 $begin$ 和 end 分别是 P 首尾字符的匹配位置。最大长度是全局约束的最大值，记为 l。因此复杂度是 O(mg+l)。设 k 为搜索区间上嵌套的匹配解数目，则循环步骤 17 的复杂度即为 O(k * (mg+l))。算法 SPLIT 的总时间复杂度是 O(k * B * (m * g+l))，其中，B 是子空间数目。SPLIT 采用的是滑动窗口结构，因此其空间复杂度即为窗口大小 O(l)。

3.2.2 SPLIT 对匹配算法时间性能的影响

定理 2 设 A 是 PMWL 算法，其时间复杂度 O(A)与文本长度|T|关系为 O(A)∝|T|$^{\alpha}$。若 B 为 SPLIT 子空间数目，m 是 P 的模长，l 是 P 的最大长度，g 是 P 中通配符跨度，k 为搜索区间上嵌套的匹配解数目。若 SPLIT 对子空间的划分是均匀的，且匹配解在文本中为均匀分布，则算法 A 的时间复杂度可以降至 O(k * B * (m * g+l)+$O(A)/B^{\alpha-1}$)。

证明：考虑到 SPLIT 对子空间划分是均匀的，且匹配解在文本中均匀分布，因此 SPLIT 可以将文本 T 均匀划分，且 B 为子空间数目，则每个子文本长为$\frac{|T|}{B}$。由于 O(A)∝|T|$^{\alpha}$，则对每个子文本，算法 A 的匹配时间为$\frac{O(A)}{B^{\alpha}}$，从而匹配所有子文本的时间为$\frac{O(A)}{B^{\alpha-1}}$。考虑到 SPLIT 预处理时间为 O($k$ * b * (m * g+l))，算法 A 总时间复杂度为 O(k * B * (m * g+l)+$\frac{O(A)}{B^{\alpha-1}}$)。

由定理 2 可知，当 α > 1 时匹配算法为非线性算法，此时 SPLIT 算法可以有效降低其时间复杂度。当 α=1 时匹配算法为线性算法，此时 SPLIT 可以与并行计算结合，将其消耗降低至 O(k * B * (m * g+l)+$\frac{O(A)}{B^{\alpha}}$+O(C))，其中 O(C)为通讯代价。由于 SPLIT 划分后的子空间彼此独立，因此 SPLIT 算法适用于并行计算，这将作为本文下一步研究工作。

4 实验与分析

本节实验验证 SPLIT 的划分完备性和有效性。实验文本包括来自 NCBI[18] 上的 DNA 序列数据集 AB038490_1、AX829174 和随机文本。模式包括具有代表性的 9 组模式和在不同模式特征下随机生成的 5100 组模式。考虑到 SPLIT 用于预处理文本，本文把 SPLIT 与匹配算法 SAIL[1]、PAIG[17] 和 SBO[12] 相结合，根据是否使用 SPLIT 划分解空间，将实验分成两组进行对照，最终输出匹配数和匹配时间。实验运行的环境为：奔腾双核 E5200、主频 2.49GHz、内存 2.0G、WINDOWS XP SP2 操作系统。

4.1 解空间划分完备性和有效性验证

为了验证 SPLIT 算法的完备性和有效性。实验列举 10 个代表性的模式，采用真实 DNA 数据，(1)对比 SAIL、PAIG 和 SBO 算法在使用 SPLIT 划分前后解空间的匹配数；(2)比较结合 SBO 算法与 SPLIT 前后时间消耗的变化。实验中的参数如表 1 所列，实验结果分别见表 2 和表 3。

表 1 解空间划分完备性实验参数列表

文本		AB038490_1,DNA 序列，|T|=131892
模式	P1	a0,3c
	P2	a0,3c0,3c
	P3	a0,0c0,0a
	P4	a1,5c1,5t1,5c
	P5	a1,5c2,10t0,6c
	P6	g0,3t0,3t0,3t0,3t0,3t0,3t0,3t0,3t
	P7	g0,3t0,3c0,3t0,3g0,3t0,3a0,3t
	P8	g0,2t0,2c0,2t0,2g0,2t0,2a0,2t0,2g0,2t0,2a0,2t
	P9	g0,50t
实验组 SPLIT_A		1. 调用 SPLIT 算法划分解空间 2. 调用算法 A 分别匹配划分后的子空间 3. 将结果汇总
对照组 A		直接调用匹配算法 A

表 2 解空间划分前后不同算法匹配解数目比较

B=10，单位：匹配数	P1	P2	P3	P4	P5	P6	P7	P8	P9
SAIL	14474	6371	2306	6393	7854	2143	1857	152	24916
SPLIT_SAIL	14474	6371	2306	6393	7854	2143	1857	152	24916
SBO	14474	6448	2306	6579	8332	2220	1892	153	24916
SPLIT_SBO	14474	6448	2306	6579	8332	2220	1892	153	24916
PAIG	26923	21236	2448	50851	124956	447363	39316	2531	644701
SPLIT_PAIG	26923	21236	2448	50851	124956	447363	39316	2531	644701

由表 2 可以看出：

(1)实验组和对照组的匹配数完全一致。本文同时比较了下节实验中所使用的 5100 组模式，结果也完全一致。因此，SPLIT 对解空间的划分并未影响 PMWL 问题中解的结构。

(2)对上述模式，PAIG 的匹配数均大于 SAIL 和 SBO。随着模式模长和跨度的增加，PAIG 的匹配数急剧增加，与文献[17]的结论一致。此外，模式 $P3$ 不含通配符，却同样受到

one-off 条件的影响。实验验证了 one-off 条件对 PMWL 解空间大小的约束。

(3)SBO 算法在模式 $P2-P8$ 的匹配数好于 SAIL，这与文献[12]分析的 SBO 能得到更高质量的匹配解的结果吻合。

表 3　解空间划分前后 SBO 算法时间性能比较

B=10，单位：秒	P1	P2	P3	P4	P5	P6	P7	P8	P9
SBO	327.5	235.97	26.47	470.11	863.36	568.63	192.81	7.16	25277
SPLIT	0.16	0.19	0.16	0.17	0.19	0.17	0.17	0.19	1.81
SBO (After SPLIT)	0.64	0.47	0.08	0.81	1.42	1.02	0.34	0.06	25277

由表 3 可以看出：

(1)经过 SPLIT 算法对解空间的划分后，非线性匹配算法 SBO 可以显著提高时间性能，平均大约提高 50 倍，这超出了定理 2 的理论预期。本文认为，SPLIT 可以将 SBO 的空间复杂度从 $O(n)$ 降至 $O(n/B)$，空间消耗的下降有助于额外降低时间消耗。

(2)对模式 $P1-P8$，SPLIT 可在 0.2s 内将解空间划分成 10 个子空间。因此 SPLIT 算法可以迅速定位分割点。

(3)对模式 $P9$，SPLIT 无法划分其解空间，$P9$ 模式在文本中即为 2。本文认为，对此情况可以考虑在完备性与时间效率上折中，即用已有的能得到更高质量的解的非线性匹配算法，如 SBO。针对上述极端嵌套情况，选择匹配数损失较小的位置进行划分，以保证时间效率。此外，由文献[12]可知，在字母表较大时，如在蛋白质序列和中文文本序列中，SPLIT 更能体现优势。

接下来，本文以 PAIG 为匹配算法，对 9 组模式调用 SPLIT_PAIG 进行了实验。SPLIT 将文本 T 分割成 100 个子文本，即 $B=100$，得到了 8 组子空间的分布图，其中模式 $P9$ 的解空间无法划分。

由图 3－图 10 可知：SPLIT 可以较为均匀地划分解空间，这与定理 2 的假设前提吻合。例如，对模式 $P2$，子文本的平均匹配数为 21236/100=212，如图 4 显示，在 100 个子文本中有 98 个文本的匹配数不超过 400。又如在模式 $P5$ 中，子文本平均匹配数为 124956/100=1250，在图 7 中，仅有 4 个子文本的匹配数超过 2000。

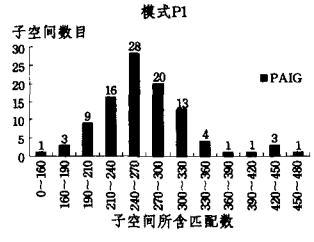

图 3　模式 $P1$ 的子空间分布图

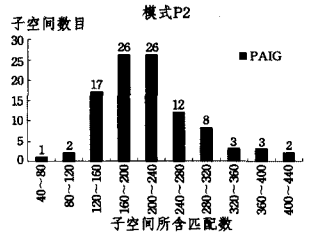

图 4　模式 $P2$ 的子空间分布图

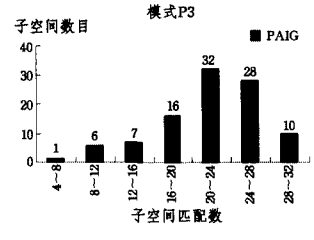

图 5　模式 $P3$ 的子空间分布图

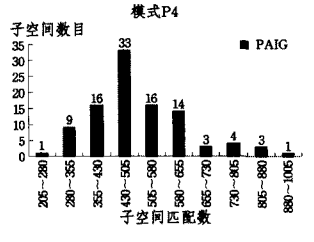

图 6　模式 $P4$ 的子空间分布图

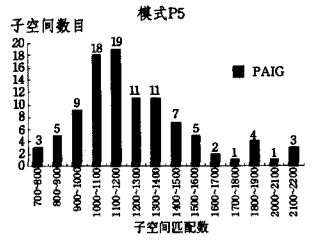

图 7　模式 $P5$ 的子空间分布图

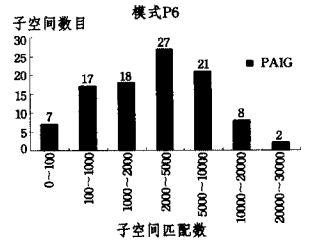

图 8　模式 $P6$ 的子空间分布图

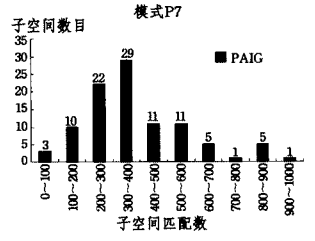

图 9　模式 $P7$ 的子空间分布图

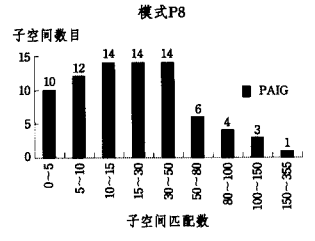

图 10　模式 $P8$ 的子空间分布图

4.2　模式特征对 SPLIT 划分效果的影响

本节实验以模式特征为变化参数，进行 5100(7 * 6 * 50+10 * 6 * 50)组模式的匹配实验，以验证在不同通配符跨度 g、模式模长 m、文本长度 n 和子空间数目 B 的情况下，SPLIT 与非线性匹配算法时间效率的关系。

首先，验证在随机文本中模式模长 m 和子空间数目 B 对 SPLIT 时间效率的影响。实验中参数如表 4 所列。

表 4　实验参数列表 1

文本	随机文本，字母表为{a,c,g,t}，n=50000
模式	g=3，m=3,6,9
B	2,5,10,20,50,100

实验结果如图 11 所示。

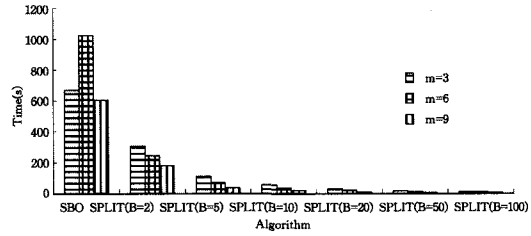

图 11　模式模长对 SPLIT 算法时间效率的影响

接下来，验证在 DNA 数据集中通配符跨度 g 和子空间数目 B 对 SPLIT 时间效率的影响。参数如表 5 所列。

表 5　实验参数列表 2

文本	AX829174，DNA 序列，n=10011
模式	m=5，g=3,6,9
B	2,5,10,20,50,100

实验结果如图 12 所示。

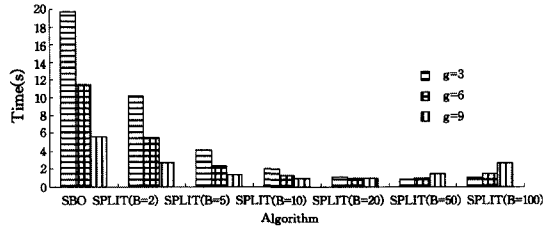

图 12　通配符跨度对 SPLIT 算法时间效率的影响

(下转第 310 页)

energy minimization[C]// 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Providence, RI, 2011: 1265-1272

[12] Andriyenko A, Roth S, Schindler K. An analytical formulation of global occlusion reasoning for multi-target tracking[C]// 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). Barcelona, 2011: 1839-1846

[13] Roshan Zamir A, Dehghan A, Shah M. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs[C]// Computer Vision-ECCV. 2012: 343-356

[14] Suard F, Rakotomamonjy A, Bensrhair A, et al. Pedestrian Detection Using Infrared images and Histograms of Oriented Gradients[C]// Intelligent Vehicles Symposium, 2006. Tokyo: IEEE, 2006: 206-212

[15] Feremans C, Labbe M, Laporte G. Generalized network design problems[J]. European Journal of Operational Research, 2003, 148(1): 1-13

[16] Benfold B, Reid I. Stable multi-target tracking in real-time surveillance video[C]// 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Providence, RI, 2011: 3457-3464

[17] Kasturi R, Goldgof D, Soundararajan P, et al. Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, 31(2): 319-336

(上接第 273 页)

本实验的目的在于比较 SBO 和 SPLIT_SBO 的时间消耗，其中每一组数据都是 50 个随机模式匹配时间的累加，以消除随机误差。可得如下结论：

(1)由图 11 和图 12 可知，在不同模式模长 m 和通配符跨度 g 的情况下，相比于 SBO，SPLIT_SBO 能够有效降低时间消耗，且随着 B 的增加，SPLIT_SBO 时间优势逐渐加大。当 B 增加到一定阈值时，图 12 中 SPLIT_SBO 的时间消耗会上升，因为此时 B 的大小已超过完备划分下子空间的数目。SPLIT 只能消耗更多的搜索分割点的时间，却无法进一步划分子空间。

(2)比较图 11 和图 12，在较长的文本中调用 SPLIT，解空间较大，可提高 B 的阈值。本文认为文本长度、模式特征与 B 阈值大小满足某种关系，这也将作为下一步研究工作。

综上所述，实验结果表明：(1)SPLIT 对解空间的划分是完备的，即在不影响原有匹配解的结构下能快速定位分割点；(2)SPLIT 可以有效降低非线性匹配算法的时间复杂度，且时间消耗与通配符跨度、模式模长、子空间数目和文本长度有关。

结束语 本文以 PMWL 问题为研究对象，着重研究了问题的解空间，有如下贡献：(1)描述了 PMWL 问题的解空间，分析了解空间的可分性；(2)提出一种面向 PMWL 解空间的划分算法 SPLIT。该算法在不影响匹配解结构的前提下能显著降低非线性匹配算法的时间复杂度。对此本文在生物 DNA 序列数据上进行了实验验证。下一步工作将从模式特征角度和并行计算角度对 PMWL 问题的复杂性进行研究。

参 考 文 献

[1] Chen Gong, Wu Xin-dong, Zhu Xing-quan, et al. Efficient string matching with wildcards and length constraints[J]. Knowledge and Information Systems, 2006, 10(4): 399-419

[2] Cole J R, Chai B, Marsh T L, et al. The ribosomal database project (RDP-11): Sequences and tools for high-throughput rRNA analysis[J]. Nucleic Acids Research, 2005, 33(1): 294-296

[3] Xie Fei, Wu Xin-dong, Hu Xue-gang, et al. Sequential Pattern Mining with Wildcards[C]// 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI). Arras, France, 2010: 241-247

[4] He Yu, Wu Xin-dong, Zhu Xing-quan, et al. Mining Frequent Patterns with Wildcards from Biological Sequences[C]// IEEE International Conference on Information Reuse and Integration. Las Vegas, IL, 2007: 329-334

[5] Califf M E, Mooney R J. Bottom-up relational learning of pattern matching rules for information extraction[J]. Journal of Machine Learning Research, 2003, 4(6): 177-210

[6] Manber U, Baeza-Yates R. An algorithm for string matching with a sequence of don't cares[J]. Information Processing Letters, 1991, 37(3): 133-136

[7] Fischer M J, Paterson M S. String matching and other products [R]. Massachusetts Institute of Technology Cambridge Project MAC, 1974: 113-125

[8] Zhang Ming-hua, Kao B, Cheung DW, et al. Mining periodic patterns with gap requirement from sequences[C]// Proc. of ACM SIGMOD. Baltimore Maryland, 2005: 623-633

[9] Bille P, Gørtz I L, Vildhøj H, et al. String matching with variable length gaps[C]// Proc. of 17th SPIRE. 2010: 385-394

[10] Muthukrishnan S, Krishna P. Non-stadard stringology: algorithms and complexity [C]// Proc. of the twenty-sixth annual ACM symposium on Theory of computing. New York, NY, USA, 1994

[11] Guo Dan, Hong Xiao-li, Hu Xue-gang, et al. A Bit-Parallel Algorithm for Sequential Pattern Matching with Wildcards[J]. Cybernetics and Systems, 2011, 42(6): 382-401

[12] 武优西，吴信东，江贺，等. 一种求解 MPMGOOC 问题的启发式算法[J]. 计算机学报，2011，34(8)：1452-1462

[13] 侯宝剑，谢飞，胡学钢，等. 基于后缀树的带有通配符的模式匹配研究[J]. 计算机科学，2012，39(12)：177-180

[14] 张雁，焦方正，卢昕玮，等. 采用染色划分改进的 RLS 算法及性能分析[J]. 软件学报，2011，22(10)：2305-2316

[15] Zhang Li-hua, Xu Wen-li, Chang Cheng. Genetic algorithm for affine point pattern matching[J]. Pattern Recognition Letters, 2003, 24: 9-19

[16] Sagot M F, Viari A. A Double Combinatorial Approach to Discovering Patterns in Biological Sequence[C]// Proc. of the 7th Symposium on Combinatorial Pattern Matching. Springer, 1996: 186-208

[17] Min Fan, Wu Xin-dong, Lu Zhen-yu. Pattern matching with independent wildcard gaps[C]// Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC-2009). Chengdu, China, 2009: 194-199

[18] National center for biotechnology information website[OL]. http://www.ncbi.nlm.nih.gov/