

# 双向图分析器的改进\*

沈李斌 陆汝占

(上海交通大学计算机科学与工程系 上海 200030)

**摘要:** 双向图算法在自然语言处理中有着重要的应用。文章澄清了关于双向图算法的一些模糊概念,并给出了一个适用于合一文法的双向图算法分析器。通过完备性证明和算法复杂度比较分析,说明了算法的正确性和可行性。在此基础上,文章对触发类驱动思想进行了扩充,提出了组合触发类驱动的图算法用于处理自然语言中的远距照应关系。

**关键字:** 自然语言处理 双向图算法分析器 触发类驱动

## Improvement of Bidirectional Chart Parser

Shen Libin Lu Ruzhan

(Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200030)

**Abstract:** Bidirectional chart parser is an important algorithm primitive for natural language processing systems. In this paper, the authors first clarify some misleading concepts, then propose an efficient bidirectional chart parsing algorithm to fit Unification Grammar better. The validity of the algorithm is verified by proving its soundness and completeness. Its feasibility is confirmed by comparing with other relative algorithms. Furthermore, as an extension of the original head-driven concept, an algorithm called composite-head driven parser is proposed to tackle long-distance relationship in Chinese.

**Keywords:** Natural Language Processing Bidirectional Chart Parser Head-driven

### 一、引言

自从上下文无关文法(CFG)的概念由 Chomsky 引入以后,有关句法分析的算法在计算机界得到了深入的研究。这些早期的算法中就包括图算法,如 Left-corner 算法<sup>[1]</sup>(自底至上的图算法)、Earley 算法<sup>[2]</sup>(自顶向下的图算法)。因为是用来处理程序语言的,早期的算法通常都是从左到右单向处理的,并且主要把信息放在语法规则上。

从八十年代以来,各类基于合一的文法的出现引发了对自然语言分析算法的新认识。在这些合一文法中,由 Gazdar (1985) 等提出的广义短语结构文法(GPSG)强调了语法分析与语义解释的一体化,并坚持严格的形式化。Bresnan & Kaplan (1982) 的词汇功能文法

---

\* 本文受国家自然科学基金和 863 计划资助

(LFG) 强调把大量语言学知识放入词典中, 以词典来驱动语法分析, M.Kay (1982) 的功能合一文法 (FUG) 则引入了功能描述与合一运算的方法。PATR-II 系统是 Shieber (1984) 提出的, 其语法描述的形式被后来的许多系统使用。由 Pollard & Sag (1987) 提出的触发类驱动文法 (HPSG) 基本上继承了 GPSG 的体系, 但更加面向实际的计算, 通过在中心词中加入子类 (Subcategorization) 信息, 使语法大大地简化了, 而信息大量集中在词典中。

正是这些文法导致了触发类这一概念的出现。传统上, 触发类是指一条产生式的右边包含信息最多的那个成分。附属在触发类上词典信息限制了产生式右边其它成分的属性, 这样一个句法分析器就能利用这些信息来减少大量不必要的搜寻。一个理想的触发类驱动<sup>[3]</sup>

(Head-driven) 的语法分析器应该首先分析产生式右边的触发类, 然后再分析产生式右边的其它成分。早期的语法分析器都是首先分析产生式右边的第一个元素。事实上, 触发类有可能出现在产生式右边任何一个位置上。所以理想的语法分析应该能够从触发类出发, 同时向两边进行分析。这类语法分析器称为双向图算法分析器 (Bidirectional Chart Parser)<sup>[4]</sup>。

关于现有的各种图算法分析器, 有一些模糊的概念需要澄清。

首先就触发类而言, 其概念已经与传统的定义有所不同。它不再是指包含信息最多的那个成分。从语法分析器的效率来看, 触发类的选择应更多地考虑到效率方面的因素。在一条规则中, 我们应该选择触发成功率最高的成分作为一条规则的触发类。

第二点, 在“双向”这个功能的处理上, 也存在不同的理解。由于, “双向”功能一方面带来了很大的灵活性, 另一方面又需要为此付出一定的代价。各种图算法分析器的性能如下: CKY 算法和 Earley 算法在最坏情况下的时间复杂度是  $O(|G|^2n^3)$ <sup>[2]</sup>, 文献[6]中, 采用了 Star-product 方法的 Earley 算法的时间复杂度是  $O(|G|n^3)$ ; 文献[7]中, 以二线性文法

(Bilinear Grammar) 为操作对象的双向图算法的实际时间复杂度为  $O(\sum |G|n^3)$ 。所以, 出于对算法时间复杂性的考虑, 对于“双向”功能的处理存在一些偏差。下文将详细讨论。

此外, 需要指出的是, 算法的时间复杂度并不是评价算法的主要标准, 双向图算法优于单向图算法体现在算法的整体效率上。同样, 时间复杂度最低的双向图算法也并非是最好的算法。由于图算法是用于处理合一文法的, 所以, 一个好的算法应该要考虑到这些方面的因素。例如, 以二线性文法为操作对象的双向图算法, 会生成大量表示中间状态的非终结符, 从而造成大量不必要的复杂特征集的复制, 同时也不便于概率信息的使用。这是第三个需要澄清的地方。

基于以上分析, 本文将首先给出一个性能良好的、以上下文无关文法为输入的双向图算法。此算法可以用作基于合一的算法的框架。本文还对传统触发类驱动的思想进行了拓展, 给出了组合触发类驱动的图算法。文章还对上述算法进行分析和论证, 并给出示例。

## 二、一个双向图算法

双向分析的最大优点在于其灵活性, 但这同时也造成了一定的问题。本文在消除这些问题同时, 将保持双向算法固有的灵活性。与传统的图算法的原理相同, 双向算法的目标也是搜索一条跨过整个输入语段的完成弧。在双向算法中, 左右可同时扩展的灵活性, 会大量

增加冗余的搜索，产生重复的弧或不必要的弧。例如：规则 R:  $D \rightarrow X_1 X_2 X_3$  (2)，括号内的 2 表示触发类是  $X_2$ 。参见图 1。

如果对双向扩展不加限制，规则 R 从触发到规约共会产生五条弧，其中弧 A、B、C 和弧 A、D、E 分别构成一个扩展的序列。一般的，对于产生式  $D \rightarrow X_1 X_2 \dots X_t$  ( $s$ )， $X_s$  为触发类，Earley 算法共产生  $t$  条弧，而不受限制的双向算法将会产生  $(t-1)$

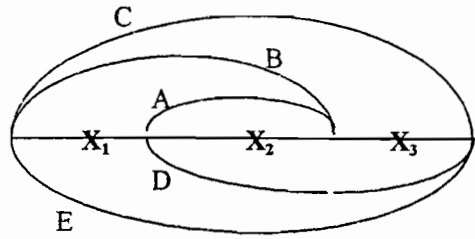


图 1

$C_{t-1}^{s-1} + 1$  条弧。为了克服这样的冗余性，有些触发类驱动的双向算法就对扩展的顺序做了限制<sup>[3,4,8]</sup>，如先向右扩展直到扩展到最右的成分，然后再向左扩展。这样的处理使得一条规则的分析方向完全取决于规则本身，而与输入语段无关。在很多场合下，这一约定是不合适的。例如在随机 CFG 分析器的计算策略中，常常要求被扩展的项所对应得概率值必须大于一定的阈值。事实上，这个概率值是依赖于输入串的。动态选择扩展方向的能力使我们能够选择一个合适的方向，从而做到搜索树的分叉尽量少，试探成功的概率尽量大。本文给出的算法中使用了称为禁止表的技术，从而消除了冗余的搜索弧，同时又保持了动态选择扩展方向的灵活性。

以下就算法所涉及到的概念进行一下说明：

首先我们以下列形式分割输入的语段，其中 0—5 称为语段中的坐标：

0 他 1 感动 2 得 3 哭 4 了 5

规则  $Hr: Dr \rightarrow X_{r_1} X_{r_2} \dots X_{r_{\pi}}(s)$  是由规则左部  $Dr$ ，规则右部和触发类  $X_{r_s}$  构成。 $\pi$  是规则右部的长度。规则右部也采用类似语段分割的方法，用坐标将其分割成一个个的成分。

弧描述一条正在被分析的规则，或是一条已经被规约的规则。其中前者称为活动弧，后者称为完成弧。图用来存放所有的活动弧和完成弧。

活动弧 C 表示成  $(Hr, i, j, l, m, Child)$ 。 $Hr$  是弧 C 所使用的规则； $i, j$  分别表示规则中已被分析的成分的首尾坐标； $l, m$  表示规则中已被分析的成分在输入串中所对应语段的首尾坐标； $Child$  记录弧 C 的子弧。下文中  $i, j$  用来表示规则中的坐标， $l, m$  用来表示语段中的坐标。

完成弧表示成  $(Hr, \_, \_, l, m, Child)$ ，参数含义同上。

扩展表用来存放所有待扩展的弧。其中既包括完成弧，也包括活动弧；有异于 Earley 算法中 Agenda 的概念。扩展表中每次弹出一条弧用来执行扩展操作，弹出的顺序取决于所选择的竞争策略。竞争策略可以使广度优先，可以是深度优先，也可以概率统计的手段等。

禁止表，如前所述，用来限制一条弧向两边扩展。一条弧只能选择向一个方向扩展，生成一条新的弧。禁止表 I 用一个  $(n+1) \times (n+1) \times 2$  的数组表示。其中  $0 \leq l, m \leq n$ ， $d \in \{\text{left, right}\}$   $I(l, m, d) \subseteq P \times \pi_G \times \pi_G$ 。其中 P 是上下文无关文法 G 的规则集， $\pi_G$  表示 G 中规则的最长长度。例如  $(Hr, i, j) \in I(l, m, \text{left})$  表示形如  $(Hr, i, j, l, m, Child)$  的弧不能向左扩展以生成新的弧。

算法 1:

设输入  $W = w_1 w_2 \dots w_n$ ,  $|W| = n$ .

Main () // 主程序

begin

for every  $w_j$  ( $j = 1 \dots n$ ) // 初始化

for  $w_j$  每个可选的词类 T

生成完全弧  $C = ( (T \rightarrow \text{terminal}), 0, 1, j-1, j, w_j )$ ,

并执行子程序 AddArc (C);

for every 扩展表中的弧 C ( $Hr, i, j, l, m, \text{Child}$ ), ( $Hr:Dr \rightarrow X_{r,1} X_{r,2} \dots X_{r,m}$

(s)), 做下列操作直到满足结束条件

if C 是活动弧

if ( $Hr, i, j$ )  $\notin I(l, m, \text{left})$

for every 满足  $Dp = X_{r,i-1}$  完成弧 E ( $Hp, \_, \_, k, l, \text{Child}$ )

合并 C 和 E, 即执行子程序 AddArc (F), 其中  $F =$

( $Hr, i-1, j, k, m, E + C.\text{Child}$ );

$I(l, m, \text{right}) = I(l, m, \text{right}) \cup \{ (Hr, i, j) \};$

if ( $Hr, i, j$ )  $\notin I(l, m, \text{right})$

for every 满足  $Dp = X_{r,j+1}$  完成弧 E ( $Hp, \_, \_, m, k, \text{Child}$ )

合并 C 和 E, 即执行子程序 AddArc (F), 其中  $F =$

( $Hr, i, j+1, l, k, C.\text{Child} + E$ );

$I(l, m, \text{left}) = I(l, m, \text{left}) \cup \{ (Hr, i, j) \};$

else // C 是完成弧

for every 图中的活动弧 E

if E 形如 ( $Hq, u, v, m, k, \text{Child}$ ) and  $X_{q,u-1} = Dr$

and ( $Hq, u, v$ )  $\notin I(m, k, \text{left})$

合并 C 和 E, 即执行子程序 AddArc (F), 其中  $F =$

( $Hq, u-1, v, l, k, C + E.\text{Child}$ );

$I(m, k, \text{right}) = I(m, k, \text{right}) \cup \{ (Hq, u, v) \};$

If E 形如 ( $Hq, u, v, k, l, \text{Child}$ ) and  $X_{q,v+1} = Dr$

and ( $Hq, u, v$ )  $\notin I(k, l, \text{right})$

合并 C 和 E, 即执行子程序 AddArc (F), 其中  $F =$

( $Hq, u, v+1, k, m, E.\text{Child} + C$ );

$I(k, l, \text{left}) = I(k, l, \text{left}) \cup \{ (Hq, u, v) \};$

end;

AddArc (弧 F ( $Hr, i, j, l, m, \text{Child}$ )) // 添加弧

begin

将 F 加入图;

将 F 加入扩展表;

if 弧 F 是完全弧

for every 有 F 触发的规则  $H_t: D_r \rightarrow X_{t,1} \dots X_{t,m}, \pi_t(s), X_{t,s}=F$   
 执行子程序  $AddArc(G)$ , 其中  $G=$   
 $(H_t, s-1, s, l, m, F)$ ;

end;

例 1: 设输入语段是“0 他 1 感动 2 得 3 哭 4 了 5”, 规则有五条:

- $H_1 \quad S \rightarrow NP \ VP \ Comp \ (3)$
- $H_2 \quad VP \rightarrow V \ (1)$
- $H_3 \quad VP \rightarrow V \ Aux \ \bar{\tau} \ (2)$
- $H_4 \quad Comp \rightarrow Aux \ 得 \ VP \ (1)$
- $H_5 \quad NP \rightarrow Pron \ (1)$

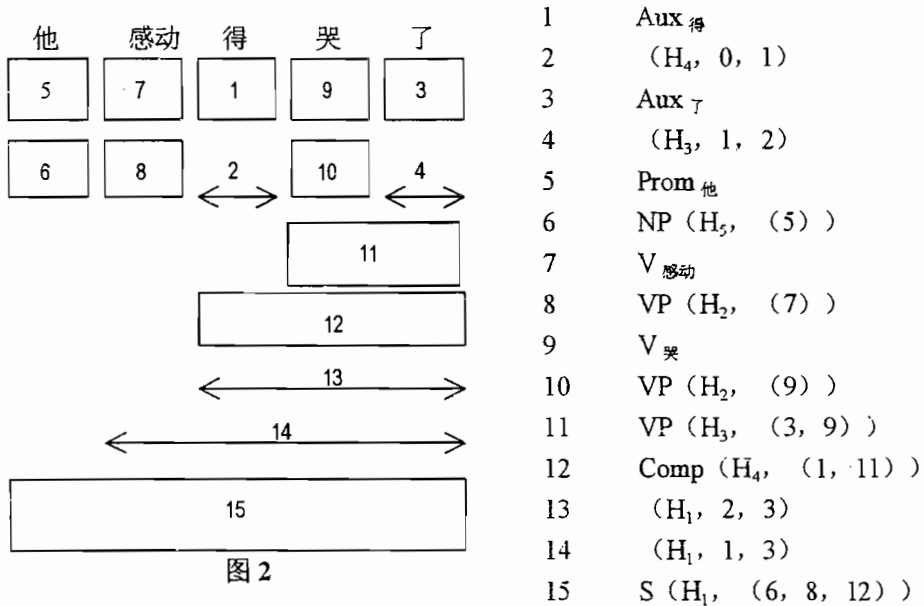


图 2

图 2 显示了计算的中间状态和结果。其中左边表示的是图，右边表示的是扩展表。在图中活动弧用箭头表示，完成弧用方格表示，在扩展表中，活动弧用  $(H, i, j)$  表示，完成弧用  $Dr(Hr, Child)$  表示。序号代表弧被写入图和扩展表的先后次序。出于叙述上的方便，例子中仅列出了五条规则。本例只是用来说明算法的，所以几乎没有用什么策略，只是在五个词写入图时交换了一下次序。以‘得’‘了’‘他’‘感动’‘哭’的次序将这五个词先后写入图和扩展表。在竞争策略的选择上，可以采用各种方法。除了使用概率统计信息之外，可以规定对应语段越长的弧优先级越高，也可以规定只有在清空当前的扩展表之后才能读取下一个输入词（用于某些改进的算法），或是采用某种唤醒的策略。竞争策略的选择上反映了双向分析算法的灵活性和高效率。

对于算法 1，我们不难证明以下命题的正确性。

命题 1: 每一条活动弧至多被扩展一次。

命题 2:  $I(l, m, left) \cap I(l, m, right) = \Phi$ 。

命题 3 (算法的完备性): 任何一个语段，若它能被某条规则规约，那么，在算法 1 中它也能被规约。

命题 3 的证明中用到了命题 2。

在算法时间复杂性计算上，如果我们采用文献[7]中原则来处理同形异构的弧（仅仅 Child 不同的弧），即将所有同形异构的弧合并成一条弧，那么此时算法的实际复杂度为  $O(\pi_G^2 |G|n^3)$ 。这里的实际复杂性是指算法中规则进行移进规约操作的时间复杂度。事实上，我们所关心的就是实际时间复杂性，因为它表示的是合一操作次数的上限。复杂度的证明如下。

证明：对于任何一次左扩展，不妨设是从语段  $(l, m)$  到语段  $(k, m)$ ， $(k < m)$ ，其中语段  $(l, m)$  所对应的活动弧是  $C(Hr, i, j, l, m, \_)$ 。那么根据上述的假设条件，对于给定的  $i, j, k, l, m, Hr$  来说，这样的扩展在算法中只会出现一次。所以，此时的实际复杂度  $O(\pi_G^2 |G|n^3)$ 。■

考虑到算法所带来的方便，从整体效率上来看，这一结果是可以接受的。实际的语法中， $\pi_G$  通常也比较小。由于，算法 1 是为合一文法设计的，所以不能简单地将同形异构的弧合并。事实上，我们可以采用 packing 技术<sup>[5]</sup>，并调整了扩展表的竞争策略，使算法 1 的实际时间复杂度也降到了  $O(\pi_G^2 |G|n^3)$ 。但是，正如我们前面所提到的那样，算法的时间复杂度并不是评价算法的主要标准。此外，算法 1 中还有很多显而易见的可改进之处。比如可以使用优化的数据结构对图和扩展表进行组织，也可以在对弧进行扩展前先比较一下左右扩展的优劣。凡此种种，考虑到算法的可读性和篇幅原因，这里就不展开讨论了。

### 三、组合触发类驱动的图算法

在自然语言的处理中，有大量的远距照应的现象存在。例如汉语中的离合词现象，又如汉语的补语结构中，趋向动词的割裂现象也形成了一种远距照应。在触发类驱动的图算法中，我们通常将照应成分中的一个作为触发类。然而，这种处理方式会造成大量冗余操作，搜索很多无效分支。例如有两条规则

$$H_1: D_1 \rightarrow X_{11}X_{12}X_{13} \quad (1)$$

$$H_2: D_2 \rightarrow X_{21}X_{22}X_{23} \quad (1)$$

其中  $X_{11} = X_{21}$ ， $X_{11}$  与  $X_{13}$  是远距照应共现的。如果有一个以  $X_{11}$  类开头的语段，采用上述算法的分析器有可能同时按两种规则进行规约，直到分析到最后的输入才能将其中的一种情况排除。即使分析器采用了某种策略，以深度优先的方式进行规约，分析器也很难从两条规则中选择出正确的那条。在这种情况下，如果我们将  $X_{11}$  与  $X_{13}$  的组合看成是规则  $H_1$  的触发器的话，搜索将会有效得多。在上例中，如果输入语段不是以  $X_{13}$  结尾的，那么  $H_1$  就不会被触发，分析器只按规则  $H_2$  进行分析。如果输入语段是以  $X_{13}$  结尾的，那么输入语段是以  $H_1$  规约的可能性当然就非常大了，于是就可以尝试先按照规则  $H_1$  进行分析。

基于这种思想，本文提出组合触发类驱动（Composite-head Driven）的图算法。现在，一条规则是由若干的成分的组合触发。考虑到实用性和效率的问题，组合至多可有两个成分

合成。在这种情况下，一条规则由这两个成分中的任意一个触发进入准活动态，当它与另一个触发成分结合时才进入活动态，并可以从四个方向继续扩展。组合触发类驱动的图算法是双向图算法的一个扩充，它所处理的规则包含两类，一类是由单个成分触发的，另一类是由两个成分的组合所触发的。在实际语法规则中，主要是前一类的规则，而后一类规则是一种有益的补充。

扩展算法中规则表示与算法 1 中规则表示基本一致，只是触发类是两个成分的组合，而非一个成分。在扩展算法中，弧分为活动弧、完全弧和准活动弧。一条规则被某一成分触发之后，并不是马上生成一条活动弧，而是生成一条准活动弧。当准活动弧被另一个触发成分触发后，才成为一条活动弧。组合触发类驱动的图算法，作为双向图算法的扩展，在算法思想上与算法 1 很相似。限于篇幅，算法的伪码从略。如果使用 packing 技术，此算法的实际时间复杂度为  $O(\pi_G^4 |G|n^3)$ 。

## 四、结语

以文中给出的双向图算法为框架，我们开发了一个汉语句法分析器<sup>[9]</sup>。在句法分析器中，我们非常方便地使用了汉语短语结构文法、词汇语法语义特征。实验证明，上述算法充分体现了图算法的灵活性，同时又能很好地控制合一次数，降低句法分析器的时间复杂度。

作为一个原型系统，算法的效率上还有很大的改进余地。我们认为，句法分析算法的效率，在很大程度上是取决于具体的句法规则和复杂特征集。一个好的算法为句法分析器提供了一个可行的框架。而整个系统的完善则需要语言知识的不断积累、改进以及语料库的逐步扩充。这也是我们下一步的计划。

考虑到合一文法在句法分析器中的广泛使用，如何选择合适的语法分析器以体现合一文法的优势非常重要。希望我们的工作能起到抛砖引玉的作用。

## 参考文献

- [1] Aho, A. V. and J. D. Ullman. The Theory of Parsing, Translation and Compiling. Prentice-Hall, 1972.
- [2] Earley, J. "An efficient context-free parsing algorithm", Commun. of the ACM 13, 2, 1970.
- [3] M. Kay. Head-driven parsing, in Proceedings Workshop on Parsing Technologies, Pittsburgh, PA 1989.
- [4] S. Stell and A. D. Roeck, Bidirectional parsing, in Hallam and C. Mellish, eds., Advances in AI, Proceedings 1987 AISB Conference, (Wiley, London, 1987)
- [5] Allen, James. Natural Language Understanding, 2nd Ed., The Benjamin/Cummings Publishing Com., 1995.
- [6] S. L. Graham, etc., An improved context free recognizer, ACM Trans. Program. Lang. Syst. 2 (3), 1980.
- [7] G. Satta, O. Stock, Bidirectional context-free grammar parsing for natural language processing, Artificial Intelligence, Sep 1994.
- [8] 吴立德, 大规模中文文本处理, 复旦大学出版社, 1997.
- [9] 沈李斌等, 一个汉语句法分析器的设计与实现, JSCL'99.