Turbo Parsers: Analisadores Sintácticos Estatísticos Baseados em Relaxações Lineares

Joaquim Nabuco

I.7 DOCUMENT AND TEXT PROCESSING
I.2 ARTIFICIAL INTELLIGENCE
I.5 PATTERN RECOGNITION

Conteúdo

Resumo

Este trabalho aborda o problema da análise sintáctica de texto em linguagem natural, usando métodos de inferência estatística. Em particular, considera-se a *sintaxe de dependências*, um formalismo adequado para aplicações como tradução automática, extracção de informação e resposta automática a perguntas. Dada uma frase, pretende-se produzir a análise sintáctica mais provável. Para o efeito, propõe-se uma *nova formulação*, *novos modelos* e um *novo algoritmo*.

Primeiro, introduz-se *uma nova formulação* do problema em termos de um programa linear inteiro com fluxos multi-mercadoria. Com base em *novos modelos*, obtém-se uma função de compatibilidade que se decompõe numa soma de funções elementares cujos domínios se sobrepõem, interpretando-se este problema—o qual é NP-completo—como a identificação da configuração de probabilidade máxima num modelo gráfico com ciclos.

Mostra-se em seguida que a relaxação linear deste problema tem afinidades com uma técnica usada em teoria da comunicação que está na base dos famosos "turbo-códigos", na qual se ignoram os efeitos globais provocados pelos ciclos.

Para resolver o problema linear, propõe-se um *novo algoritmo* de decomposição dual baseado no método dos multiplicadores com direcções alternadas, o qual apresenta melhores propriedades de convergência do que outros anteriormente propostos, sendo particularmente adequado para decomposições com um elevado número de sobreposições.

A abordagem proposta é validada experimentalmente em 14 línguas diferentes, obtendo-se desempenhos acima do estado-da-arte.

Palavras-chave: processamento de linguagem natural, análise sintáctica de dependências, modelos gráficos probabilísticos, optimização linear inteira, decomposição dual.

Capítulo 1

Introdução

Nos últimos anos, o progresso científico e tecnológico na área das tecnologias da linguagem tem sido notório, com efeitos visíveis na qualidade dos motores de pesquisa, dos sistemas de extracção de informação e dos sistemas de tradução automática. Em boa medida, este sucesso pode ser explicado pelo advento da World Wide Web e, em consequência, pela enorme quantidade de dados tornados disponíveis. O tratamento estatístico destes dados, utilizando técnicas de aprendizagem automática [59, 51, 70, 8], fez emergir um novo paradigma que coloca ênfase no processamento em grande escala dos dados [35], demarcando-se da metodologia clássica de representação do conhecimento que foi dominante na área de inteligência artificial até à década de 1980.

No contexto do processamento de linguagem natural, a disponibilidade crescente de *corpora* anotados por especialistas abriu caminho para o uso de métodos baseados em aprendizagem automática estatística. Como consequência, têm-se obtido modelos estatísticos cada vez mais fidedignos, com aplicações na análise sintáctica e semântica e na tradução automática de texto [13, 25, 40, 45]. Nas décadas de 1980 e 1990, os modelos probabilísticos empregues eram assaz rudimentares, assentando em relações de independência estatística conducentes (no caso supervisionado) a uma fácil estimação de parâmetros e inferência. São exemplo desse paradigma os modelos generativos de Markov com variáveis latentes (*hidden Markov models*, [39, 65]). Pese embora a grande eficiência computacional destes modelos, a sua simplicidade não permite capturar a riqueza de fenómenos inerentes às linguagens naturais.

Na última década, os importantes avanços que se registaram em *predição estruturada discriminativa* [48, 75, 3, 77] abriram as portas para modelos mais sofisticados, com menos "assunções de independência", permitindo, por exemplo, aliar a robustez dos métodos estatísticos ao poder de expressão da *lógica de primeira ordem* [66]. Se, por um lado, estes avanços fizeram ressurgir o interesse nos métodos de representação do conhecimento, também vieram colocar novos desafios, porquanto conduzem a problemas de estimação e inferência mais difíceis, tendo frequentemente que empregar-se métodos de inferência aproximada, baseados em amostragem, procura heurística, ou cálculo de variações. É nesta linha de investigação que se enquadra o trabalho aqui apresentado.

Em concreto, o problema que aqui abordamos é o da *análise sintáctica de texto*, adoptando-se como formalismo gramatical a *sintaxe de dependências*. As origens históricas da sintaxe de dependências remontam à gramática sânscrita de Pāṇini no século IV a.C., devendo-se o seu tratamento moderno aos trabalhos seminais de Tesnière [76], Hudson [38], Mel'čuk [58], entre outros. Trata-se de um formalismo que, pelo seu carácter lexical, pela relativa facilidade de anotação, e por não requerer a construção explícita de uma gramática, apresenta importantes vantagens computacionais face a formalismos mais tradicionais, tais como as famosas *gramáticas de estrutura sintagmática* (*phrase-structure grammars*, [14]). Embora as gramáticas sintagmáticas sejam ainda predominantes, nos últimos anos tem-se assistido, na área da linguística computacional, à sua progressiva substituição pela análise sintáctica de dependências, a qual tem sido incorporada em sistemas para extrair relações entre entidades [18], sistemas de tradução automática [19], e sistemas de resposta a perguntas [81, 24], entre outros.

A Figura 1.1 mostra representações sintácticas segundo estes dois formalismos, ilustrando

o conceito de *ambiguidade*, o qual é premente nas linguagens naturais. Uma árvore de dependências é constituída por *arcos* entre palavras, em que cada arco representa uma função gramatical. Por exemplo, na Figura 1.1, o verbo "resolveu" tem um arco dirigido a "Joaquim", que desempenha a função de sujeito, outro dirigido a "problema", o objecto directo, e ainda outro dirigido a "com", que serve de núcleo do modificador oblíquo *com o método estatístico*. O problema da análise sintáctica de texto consiste em, dada uma frase, obter a sua representação sintáctica mais provável, resolvendo possíveis ambiguidades.

1.1 Motivação

A análise sintáctica de texto sofreu uma evolução considerável nas últimas duas décadas. Nos últimos anos, a criação de bancos de árvores sintácticas para várias línguas, de que são exemplos o projecto *Penn Treebank* para a língua inglesa [52] e a *Floresta Sintá(c)tica* para o Português [2], tornou possível abordar o problema através de técnicas de aprendizagem supervisionada. Estes bancos contêm milhares de frases emparelhadas com as respectivas representações sintácticas, anotadas por especialistas. Desta forma, pode enquadrar-se o problema no âmbito da *classificação estatística estruturada* [48, 75, 3, 77], utilizando uma fracção dos dados anotados como conjunto de treino. Em publicações anteriores, debruçámo-nos sobre formas de *treinar* o modelo formulando o problema de aprendizagem como minimização de uma função de custo convexa (o *risco empírico regularizado*) e empregando algoritmos de gradiente estocástico (o estado-da-arte em problemas estruturados de grande escala). Neste trabalho, coloca-se ênfase no problema de *inferência*, no qual se assume dispor de um modelo já treinado, e cujo objectivo é utilizá-lo para *inferir* (ou *descodificar*) a representação sintáctica da frase apresentada.

Conforme será discutido de seguida (Secção 1.2), os últimos anos têm sido pródigos no desenvolvimento de modelos estatísticos de linguagem cada vez mais sofisticados. Existe a percepção, por parte da comunidade científica, de que há uma "longa cauda" de fenómenos linguísticos que só poderá ser capturado introduzindo no modelo características (*features*) mais "globais".[1] Uma limitação actual é que a inclusão deste tipo de características afecta negativamente a simplicidade e eficiência dos algoritmos de inferência. Por esta razão, o progresso nesta área tem sido moldado pela capacidade de estender os algoritmos existentes, ao invés de ser orientado pelas características que podem ter impacto na qualidade do modelo, como seria desejável. Isto é, a tendência actual é a de *subjugar* o modelo ao desenho do algoritmo.

Este trabalho contribui para uma mudança de paradigma, aliviando a necessidade de desenvolver novos algoritmos de cada vez que se pretende adicionar novas características. Como se verá, isto é conseguido introduzindo uma nova formulação do problema e reconhecendo a sua ligação com problemas estudados em modelos gráficos probabilísticos e teoria da comunicação, os quais servem de inspiração para propormos uma nova técnica de inferência aproximada e um novo algoritmo de optimização, conforme descreveremos na Secção 1.3.

1.2 Trabalho Relacionado

Essencialmente, existem três classes de analisadores sintácticos de dependências propostos na literatura: *analisadores baseados em optimização global*, *analisadores baseados em transições* e *analisadores híbridos*. Para além disso, existem dois formalismos de dependências de sintaxe que têm sido considerados: aqueles que se restringem a árvores projectivas (conceito que definiremos rigorosamente na Secção 2.1) e os que permitem árvores não-projectivas.

[1] Por "características globais", neste contexto, deve entender-se características capazes de modelizar dependências de longo alcance, as quais não são possíveis de incorporar em modelos markovianos de baixa ordem.

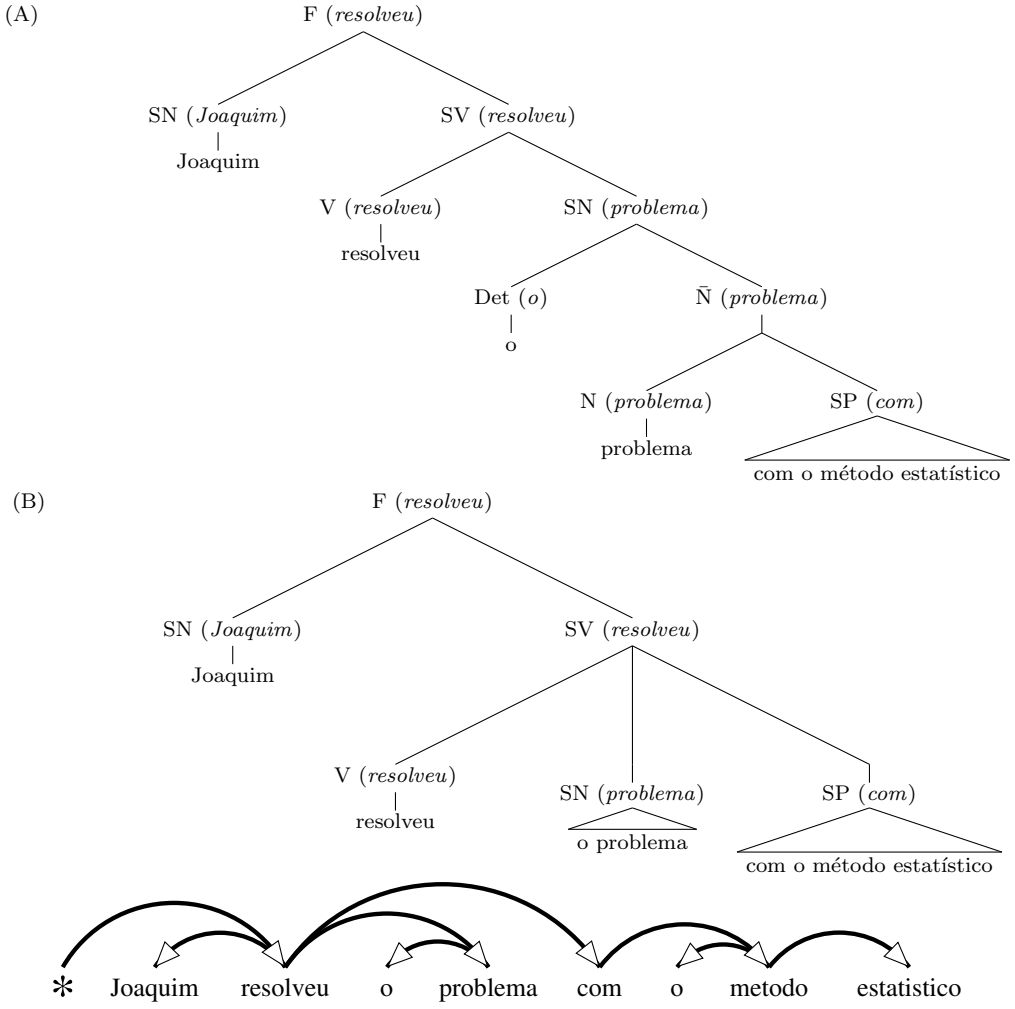

Figura 1.1: Análises sintácticas para a frase ambígua *Joaquim resolveu o problema com o método estatístico*. Em (A) e (B), mostra-se duas árvores sintagmáticas ilustrando duas possíveis leituras para esta frase: na primeira, o método estatístico *tem* um problema, que o Joaquim resolveu; na segunda, Joaquim empregou o método estatístico *como forma de* resolver o problema. As duas análises diferem na posição onde é introduzido o sintagma preposicional (SP) "com o método estatístico". Este tipo de ambiguidade é muito comum e um dos mais difíceis de tratar computacionalmente. Apresentam-se ambas as árvores sintagmáticas com anotações lexicais em cada nó (ou *sintagma*), que representam a palavra que serve de *núcleo* ao sintagma. Por norma, o núcleo de um sintagma verbal (SV) é o verbo, o de um sintagma nominal (SN) é o nome, e assim por diante. Em (B), apresenta-se ainda a correspondente *árvore de dependências*. Esta árvore é obtida da respectiva árvore sintagmática deixando cair toda a informação acerca dos sintagmas e preservando apenas as anotações lexicais. Cada arco entre palavras representa uma função gramatical (ver texto).

O nosso trabalho enquadra-se na classe dos analisadores baseados em optimização global, a qual passamos a descrever. Eisner [23] e McDonald et al. [56] consideraram modelos *factorizados em arcos*; estes modelos assignam a cada possível arco um determinado *preço*, calculado com base nos parâmetros do modelo e em características extraídas da frase. De seguida, procura-se a árvore que maximiza o *lucro*, isto é, a soma dos preços de cada arco. Se nos restringirmos a árvores projectivas, a solução para este problema pode ser calculada em tempo $O(L^3)$ (em que L é o número de palavras da frase), utilizando algoritmos de programação dinâmica [23]. Sem a restrição de projectividade, o problema foi pela primeira vez estudado por McDonald et al. [56], que mostrou equivalência com o problema de encontrar a *arborescência de maior peso* (*maximal weighted arborescence*), o qual pode ser resolvido em tempo $O(L^3)$ através do algoritmo de Chu-Liu-Edmonds [15, 22], ou até mais rapidamente [74, 27].

Na prática, os modelos factorizados em arcos são pouco expressivos e incapazes de capturar a riqueza das linguagens naturais. No caso projectivo, o algoritmo de Eisner [23] pode ser generalizado, mantendo a complexidade original de $O(L^3)$, para certos *modelos de segunda ordem* que, além de preços para arcos, incluem também preços para determinados *pares de arcos*—nomeadamente aqueles que formam um contexto de Markov "horizontal", como será descrito em detalhe na Secção 2.3. Na mesma linha de investigação, Carreras [12] e Koo e Collins [44] consideram um conjunto mais alargado de pares de arcos (formando um contexto de Markov "vertical") e introduzem *modelos de terceira ordem*, aumentando o tempo de execução do algoritmo para $O(L^4)$. Sem a restrição de projectividade, o problema é mais complicado: com efeito, McDonald e Satta [55] mostraram que qualquer tentativa de estender o modelo factorizado em arcos torna o problema NP-completo. Não obstante, têm sido propostos vários algoritmos aproximados para modelos de segunda ordem com bons desempenhos [54, 72, 45]. Estes métodos serão revistos em maior detalhe ao longo deste trabalho.

Numa linha de investigação diferente, Nivre et al. [60] e Huang e Sagae [37] propuseram *analisadores sintácticos baseados em transições*—aqui, um analisador é visto como um autómato que percorre a frase da esquerda para a direita, realizando uma sequência de decisões "gananciosas" (*greedy*), o que resulta, no final, numa árvore de dependências. Estes analisadores são mais rápidos do que os baseados em optimização global e têm a vantagem de permitir incorporar facilmente características globais; porém, são geralmente menos precisos.

Finalmente, os *analisadores híbridos* procuram juntar as melhores propriedades das duas classes de métodos supra-mencionadas, utilizando técnicas de combinação de classificadores [61]. Tipicamente, estes analisadores são mais precisos do que cada um dos analisadores envolvidos na combinação.

1.3 Principais Contribuições

As principais contribuições deste trabalho são as seguintes:

- A introdução de novos modelos estatísticos para a análise sintáctica de dependências, os quais são *mais expressivos* que os modelos anteriormente propostos. Para além de características de segunda-ordem, os nossos modelos introduzem novas características não-markovianas horizontais e verticais, bigramas de núcleos e características para arcos não-projectivos.

- Uma nova formulação do problema de inferência em termos de um programa inteiro com um número polinomial de restrições. Esta formulação é baseada em fluxos multi-mercadoria e é vantajosa face a formulações anteriores, em que o número de restrições é exponencial.

- Um método aproximado consistindo na relaxação linear da formulação anterior. Esta relaxação é interpretada como um método de inferência aproximada num grafo de factores

com ciclos, em que se desprezam os efeitos globais causados pelos ciclos. O mesmo princípio é utilizado em outros domínios, estando por exemplo na origem dos famosos turbo-códigos em teoria da comunicação [6, 57]. Por esta razão denominam-se os analisadores sintácticos daqui resultantes como *turbo parsers*.

- Um novo algoritmo de optimização para resolver este problema linear de forma eficiente, tirando partido da estrutura do grafo. Trata-se de um algoritmo de consenso, reminiscente da técnica de decomposição dual, mas que recorre a penalizações quadráticas, conduzindo a consensos mais rápidos do que algoritmos anteriores [45].

A fim de testar a aplicabilidade dos novos modelos e algoritmos propostos, procedeu-se à sua validação experimental em 14 línguas diferentes, obtendo-se resultados acima do estado-da-arte para algumas destas línguas. Os tempos de execução são competitivos com aqueles obtidos com *software* comercial de optimização linear (CPLEX).

1.4 Organização do Documento

Este documento está organizado da forma que a seguir se descreve.

O Capítulo 2 começa por rever os conceitos fundamentais de análise sintáctica de dependências que são necessários para a exposição deste trabalho. De seguida, introduz uma nova formulação do problema em termos de um programa linear inteiro com um número polinomial de restrições, mostrando-se a sua flexibilidade e capacidade para incorporar uma grande variedade de características globais.

O Capítulo 3 introduz os *turbo parsers*. Primeiro, revê-se os conceitos necessários de modelos gráficos probabilísticos e introduz-se um grafo de factores que representa a estrutura do programa linear inteiro acima mencionado. De seguida, sugere-se e analisa-se uma relaxação linear do problema, que se interpreta como um método de inferência aproximada nesse grafo de factores, mostrando-se a ligação a outros métodos utilizados em teoria de correcção de códigos.

O Capítulo 4, para resolver a relaxação linear de forma eficiente, introduz um novo algoritmo de optimização baseado na técnica de decomposição dual e no método das direcções alternadas. De seguida, valida-se experimentalmente os modelos e algoritmos propostos, comparando-os com outros analisadores sintácticos e com optimizadores lineares genéricos.

Finalmente, o Capítulo 5 apresenta as conclusões do trabalho e traça direcções de investigação futura.

1.5 Publicações Prévias

Partes do trabalho que aqui se apresenta foram desenvolvidas nos últimos três anos, tendo tido elevada receptividade por parte da comunidade científica. Foram publicados seis artigos em conferências internacionais: um artigo publicado nas actas da conferência *Annual Meeting of the Association for Computational Linguistics* (ACL), dois artigos na conferência *International Conference on Machine Learning* (ICML), um artigo nas actas da *International Workshop in Optimization for Machine Learning*, colocada com a conferência *Neural Information Processing Systems* (NIPS), e dois artigos na conferência *Empirical Methods on Natural Language Processing* (EMNLP). Uma das publicações acima referidas foi galardoada com o prémio de melhor artigo da conferência desse ano.

Para respeitar o anonimato imposto pelo regulamento, não incluimos estas publicações nas referências deste trabalho.

Capítulo 2

Formulação Linear Inteira para Análise de Dependências

Neste capítulo, apresentamos uma nova formulação inteira para o problema da *análise sintáctica de dependências*. O objectivo é separar o desenho do modelo das considerações de natureza algorítmica, trilhando o caminho para empregar um vasto conjunto de características e restrições globais, conducente a modelos sintácticos mais eficazes do que outros considerados na literatura.

Em certa medida, as formulações inteiras constituem uma tendência actual no processamento de linguagem natural, tendo conduzido a resultados notáveis em várias aplicações, tais como classificação de tarefas semânticas [64], alinhamento de palavras para tradução automática [47] e sumarização automática de texto [16], entre outras. Estas formulações permitem incorporar com facilidade características ou restrições globais, as quais são difíceis de lidar com algoritmos tradicionais baseados em programação dinâmica. Apesar da programação linear inteira pertencer à classe de problemas NP-completos [71], existe um leque alargado de *software* de optimização que torna esta uma solução prática para certos problemas de dimensão limitada.

No contexto específico da análise de dependências, Riedel e Clarke [67] propuseram uma formulação inteira; porém, essa formulação possui um número *exponencial* de restrições, limitando a sua aplicabilidade. A obtenção de formulações *eficientes* permanece um problema em aberto, o qual agora endereçamos. A nossa formulação possui as seguintes vantagens comparativas:

- O número de variáveis e restrições depende *polinomialmente* do comprimento da frase, ao invés de requerer um número exponencial de restrições.
- Pode obter-se um procedimento de inferência eficiente considerando uma relaxação linear do problema (o que possibilita ainda o treino rápido do modelo de forma discriminativa).
- É possível aprender automaticamente restrições "suaves" através dos dados de treino. Para além das características anteriormente consideradas na literatura, consideramos novas interacções de segunda ordem, algumas não-markovianas, e favorecemos árvores aproximadamente projectivas, quando este fenómeno é observado nos dados (Figura 2.1).

É de referir que a formulação apresentada é extremamente flexível, podendo incluir-se com relativa facilidade outras características ou restrições globais, utilizando conhecimento linguístico. Diferimos para o Capítulo 4 a análise experimental dos analisadores sintácticos daqui resultantes.

2.1 Sintaxe de Dependências

Considera-se uma frase $x := (x_1, \ldots, x_L)$ constituída por L palavras, às quais se acrescenta uma palavra especial x_0 (representada pelo símbolo $*$ na Figura 1.1). Uma árvore de dependências é constituída por *arcos* entre palavras; cada arco representa uma relação gramatical. Se (h, m) for um arco partindo de h e com destino em m, a palavra h diz-se *núcleo* e a palavra m *modificador*. Diz-se que m *modifica* h, representando-se esta relação como $h \rightarrow m$. Em geral, adjectivos modificam nomes e estes modificam verbos. A sequência de palavras entre h e m (incluindo

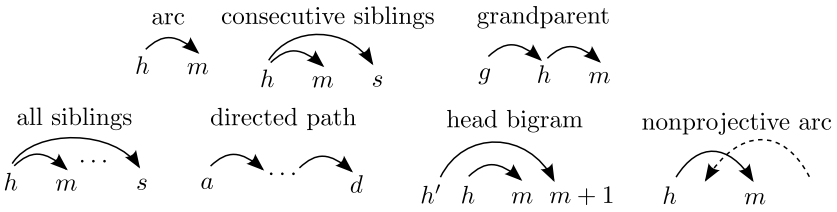

Figura 2.1: Partes utilizadas pelo analisador sintáctico desenvolvido neste trabalho. Os *arcos* constituem as partes básicas: uma árvore de dependências pode ser "lida" através da lista dos arcos que possui. As partes para *irmãos consecutivos* (*consecutive siblings*) e *avós* (*grandparents*) introduzem markovização horizontal e vertical [54, 12]. A assunção de Markov horizontal é quebrada através de partes para *todos os irmãos* (*all siblings*) e a vertical através de partes de *caminho direccionado* (*directed path*). Inspirados por analisadores baseados em transições, consideramos ainda partes de *bigrama de núcleo* (*head bigrams*), que observam os núcleos atribuídos a palavras consecutivas. Finalmente, incluímos partes que indicam se um arco é *não-projectivo*.

as próprias) é chamada a *extensão* do arco (h, m). Diz-se que uma palavra a *antecede* outra palavra d se existe um caminho direccionado de a para d, dizendo-se então que d *descende* de a e escrevendo-se $a \rightarrow^* d$. Por convenção, toda a palavra descende de si própria. Um arco (h, m) diz-se *projectivo* se qualquer palavra na sua extensão (ou seja, entre h e m) descende do núcleo h. Diz-se que uma árvore de dependências é *projectiva* se todos os seus arcos forem projectivos. A Figura 2.2 mostra uma árvore não-projectiva.[1] Estas definições implicam os seguintes factos:

1. Uma árvore é projectiva se e só se todos os arcos estiverem *aninhados*; *i.e.*, para quaisquer par de arcos (h, m) e (h', m') ou as respectivas extensões não se intersectam ou, intersectando-se, uma está contida na outra, não podendo ter-se, por exemplo, $h < h' < m < m'$. Graficamente, *os arcos nunca se cruzam se desenhados no mesmo lado.*

2. Quando restritas a árvores de dependência projectivas, as gramáticas de dependência têm o mesmo poder generativo das de estrutura sintagmática [28].

2.2 O Polítopo das Arborescências

Seja $\mathcal{Y}(x)$ o conjunto das *potenciais* árvores de dependências para a frase x. Vamos em seguida caracterizar este conjunto geometricamente. Considere-se o grafo direccionado completo $\mathscr{D} = (\mathcal{N}, \mathcal{A})$, com nós $\mathcal{N} = \{0, \ldots, L\}$ e arcos $\mathcal{A} = \mathcal{N}^2$. Diz-se que $\mathcal{B} \subseteq \mathcal{A}$ é uma *r-arborescência* de $\mathscr{D}$ (em inglês, *arborescence* ou *directed spanning tree*) se $(\mathcal{V}, \mathcal{B})$ é uma árvore direccionada enraizada no nó r; a Figura 2.2 ilustra este conceito. Por definição, o conjunto $\mathcal{Y}(x)$ é isomorfo ao conjunto das 0-arborescências de $\mathscr{D}$. Podemos portanto formular o problema da análise de dependências como um problema de procura no conjunto das arborescências de $\mathscr{D}$, conjunto este que pode ser caracterizado poliedralmente, como veremos de seguida.

Dada uma árvore de dependências $y \in \mathcal{Y}(x)$, seja $\boldsymbol{z} := (z_a)_{a \in \mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|}$ o seu vector de incidência, *i.e.*, z_a é 1 se o arco a pertence à árvore y, e 0 caso contrário. Tomando todos os vectores de incidência válidos obtemos um conjunto de pontos em $\{0, 1\}^{|\mathcal{A}|}$. O fecho convexo deste conjunto[2] é o *polítopo das arborescências*, o qual denotamos por $\mathcal{Z}_{\text{tree}}$. Cada vértice deste

[1] A sintaxe não-projectiva adequa-se sobretudo a linguagens de "ordem livre," isto é, para as quais é comum poder permutar-se a ordem dos sintagmas de uma frase; exemplos são o Russo, o Coreano, o Japonês, o Alemão e o Finlandês. Porém, conforme atesta a Figura 2.2, análises não-projectivas são também convenientes para linguagens predominantemente projectivas (como o Inglês e o Português).

[2] O *fecho convexo* de um conjunto $\mathcal{S} := \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K\}$ é o menor conjunto convexo que contém $\mathcal{S}$, consistindo no polítopo $\operatorname{conv} \mathcal{S} := \{\sum_{i=1}^K \lambda_i \boldsymbol{z}_i \mid \sum_{i=1}^K \lambda_i = 1, \lambda_i \geq 0\}$.

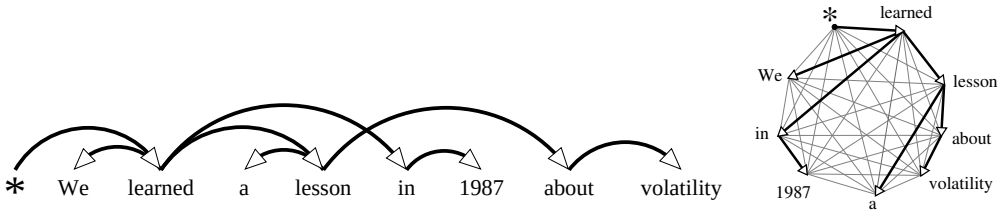

Figura 2.2: Uma árvore de dependências não-projectiva; o arco *lesson* → *about* é não-projectivo. À direita, a mesma árvore vista como uma arborescência do grafo direccionado completo.

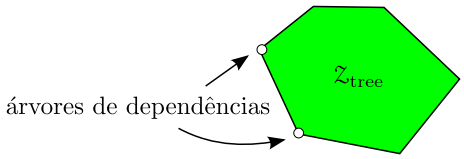

Figura 2.3: Representação esquemática do polítopo das arborescências $\mathcal{Z}_{\text{tree}}$. Cada vértice deste polítopo corresponde a uma árvore de dependências no conjunto $\mathcal{Y}(x)$.

polítopo corresponde de forma biunívoca a uma árvore de dependências em $\mathcal{Y}(x)$ (Figura 2.3). O teorema de Minkowski-Weyl [68] garante que $\mathcal{Z}_{\text{tree}}$ pode ser representado em termos de um conjunto de desigualdades lineares para algum $P \in \mathbb{N}$, matrix $\mathbf{A} \in \mathbb{R}^{P \times |\mathcal{A}|}$ e vector $\boldsymbol{b}$:

$$\mathcal{Z}_{\text{tree}} = \{\boldsymbol{z} \in \mathbb{R}^{|\mathcal{A}|} \mid \mathbf{A}\boldsymbol{z} \leq \boldsymbol{b}\}. \tag{2.1}$$

Porém, não é óbvio como obter uma representação *concisa*—isto é, na qual P cresça polinomialmente com o número de palavras L. Descrevemos de seguida duas possíveis representações:

- uma *representação baseada em ciclos*, em que P cresce exponencialmente com L; trata-se da representação implicitamente utilizada por Riedel e Clarke [67];
- uma *representação baseada em fluxos multi-mercadoria* (*multi-commodity flows*), requerendo $O(L^3)$ variáveis e restrições, a qual demonstraremos ser exacta.

Representação Baseada em Ciclos. Para um conjunto $\mathcal{B} \subseteq \mathcal{A}$ ser uma 0-arborescência de $\mathscr{D}$ é necessário e suficiente que cubra todos os nós e que seja uma árvore enraizada em 0, ou seja, um grafo sem ciclos. Formalmente, tem que satisfazer as seguintes condições:

C1. Cada nó em $\mathcal{N} \setminus \{0\}$ tem exactamente um arco em $\mathcal{B}$ que lhe é dirigido; e não há nenhum arco em $\mathcal{B}$ dirigido para 0,

C2. $\mathcal{B}$ não contém ciclos.

Para cada nó $n \in \mathcal{N}$, seja $\delta^-(n) := \{(i,j) \in \mathcal{A} \mid j = n\}$ o conjunto dos arcos que lhe são dirigidos e $\delta^+(n) := \{(i,j) \in \mathcal{A} \mid i = n\}$ o conjunto dos que nele têm origem. A primeira condição é facilmente expressa através de restrições lineares:

$$\textstyle\sum_{a \in \delta^-(j)} z_a = 1, \text{ para cada } j \in \mathcal{N} \setminus \{0\}; \quad \sum_{a \in \delta^-(0)} z_a = 0. \tag{2.2}$$

A segunda condição é mais trabalhosa, sendo necessário enumerar cada possível ciclo e excluir as configurações que o incluam. Seja $\mathscr{C} \subseteq 2^{\mathcal{A}}$ o conjunto de todos os possíveis ciclos, cada ciclo $\mathcal{C} \in \mathscr{C}$ sendo representado pelos arcos que contém. As restrições seguintes impõem a condição:

$$\textstyle\sum_{a \in \mathcal{C}} z_a \leq |\mathcal{C}| - 1, \quad \text{para quaisquer } \mathcal{C} \in \mathscr{C}. \tag{2.3}$$

Infelizmente, o conjunto $\mathscr{C}$ cresce exponencialmente com L, resultando num número exponencial de restrições na forma (2.3). Para obviar este problema, Riedel e Clarke [67] empregaram um método de planos de corte que adiciona restrições de forma incremental à medida que as respectivas violações são detectadas. Porém, o algoritmo resultante é demasiado lento.

Representação Baseada em Fluxos Multi-Mercadoria. Vamos ver de seguida que é possível representar $\mathcal{Z}_{\text{tree}}$ com um número polinomial de restrições. A chave é substituir a condição C2 acima, que proíbe ciclos, por uma condição alternativa:

C2'. $\mathcal{B}$ é conexo. Por outras palavras, existe um caminho direccionado da raiz 0 para cada um dos restantes nós em $\mathcal{N} \setminus \{0\}$ consistindo apenas de arcos em $\mathcal{B}$.

Resulta da definição de arborescência que as condições C1-C2 e C1-C2' são equivalentes, ambas definindo o conjunto $\mathcal{Y}(x)$. Porém, o último conjunto de condições é mais conveniente. Uma forma de assegurar a conectividade de grafos é através de *restrições de fluxo*, impondo que, para cada $n \in \mathcal{N} \setminus \{0\}$, exista um conjunto de arcos instalados em $\mathcal{B}$ que assegure o fluxo de mercadorias de 0 para n. Esta é a intuição subjacente à representação do polítopo de arborescências através de *fluxos multi-mercadoria*. Neste modelo, associa-se a cada nó $k \neq 0$ uma "mercadoria": uma unidade da mercadoria k é produzida na raíz 0 e deve ser transportada para o nó k; a variável binária ϕ_{ij}^k designa o fluxo da mercadoria k no arco (i, j). Fazendo uso destas variáveis adicionais, obtemos as seguintes restrições (para além daquelas expressas em (2.2)):

- A raiz envia uma unidade de mercadoria para cada nó:

$$\textstyle\sum_{a \in \delta^-(0)} \phi_a^k - \sum_{a \in \delta^+(0)} \phi_a^k = -1, \qquad \text{para cada } k \in \mathcal{N} \setminus \{0\}. \tag{2.4}$$

- Cada nó consome a sua própria mercadoria e nenhuma outra:

$$\textstyle\sum_{a \in \delta^-(j)} \phi_a^k - \sum_{a \in \delta^+(j)} \phi_a^k = \begin{cases} 1 & \text{se } j = k, \\ 0 & \text{caso contrário,} \end{cases} \qquad \text{para cada } j, k \in \mathcal{N} \setminus \{0\}. \tag{2.5}$$

- O fluxo é nulo para arcos que não estão instalados na árvore:

$$\phi_a^k \leq z_a, \qquad \text{para cada } a \in \mathcal{A} \text{ and } k \in \mathcal{N}. \tag{2.6}$$

Para além disso, todas as variáveis pertencem ao intervalo unitário. Consideremos o polítopo definido por (2.2) e (2.4–2.6), projectado no espaço das variáveis $\boldsymbol{z}$:

$$\mathcal{Z}_{\text{mc}} \;:=\; \left\{ \boldsymbol{z} \in [0,1]^{|A|} \mid \exists \boldsymbol{\phi} \geq \mathbf{0} : \; (\boldsymbol{z}, \boldsymbol{\phi}) \text{ satisfaz a Eq. 2.2 e as Eqs. 2.4–2.6} \right\}. \tag{2.7}$$

A proposição seguinte assegura que esta representação é exacta, ou seja, que as restrições de fluxo providenciam uma representação "levantada" (*lifted*) do polítopo de arborescências $\mathcal{Z}_{\text{tree}}$.

Proposição 2.1 *A representação com fluxos multi-mercadoria é exacta,* i.e.*,* $\mathcal{Z}_{\text{mc}} = \mathcal{Z}_{\text{tree}}$.

Demonstração: Seja $\mathcal{S} \in \mathcal{N}$ uma colecção arbitária de nós, e designe-se por $\delta^+(\mathcal{S}) := \{(i, j) \in \mathcal{A} \mid i \in \mathcal{S}, j \notin \mathcal{S}\}$ o conjunto dos arcos com origem num nó de $\mathcal{S}$. Estes arcos formam um *corte* do grafo, associado com a partição $\{\mathcal{S}, \mathcal{N} \setminus \mathcal{S}\}$. Impôr conectividade é o mesmo que requerer que cada possível corte tenha pelo menos um arco instalado, o que pode ser expresso pelas restrições:

$$\textstyle\sum_{a \in \delta^+(\mathcal{S})} z_a \geq 1, \quad \text{para cada subconjunto próprio } \mathcal{S} \subset \mathcal{N} \text{ com } 0 \in \mathcal{S}. \tag{2.8}$$

Magnanti e Wolsey [50, Proposição 5.2] mostram que as restrições (2.8), em conjunto com a Eq. 2.2 e a restrição de não-negatividade $\boldsymbol{z} \geq \mathbf{0}$, constituem uma representação exacta do polítopo $\mathcal{Z}_{\text{tree}}$. Mostramos de seguida que a nossa representação com fluxos multi-mercadoria é equivalente a esse conjunto de restrições, aplicando o teorema "min-cut max-flow" [71]. Para

cada nó $k \in \mathcal{N} \setminus \{0\}$, seja $\mathscr{S}_k := \{\mathcal{S} \subset \mathcal{N} \mid 0 \in \mathcal{S} \wedge k \notin \mathcal{S}\}$. Note-se que as restrições (2.8) podem ser reescritas como L restrições não lineares, uma para cada $k \in \mathcal{N} \setminus \{0\}$:

$$\min_{\mathcal{S} \in \mathscr{S}_k} \sum_{a \in \delta^+(\mathcal{S})} z_a \geq 1. \tag{2.9}$$

Esta desigualdade é na realidade uma igualdade, uma vez que para o conjunto $\mathcal{S} = \mathcal{N} \setminus \{k\}$, tem-se $\delta^+(\mathcal{S}) = \delta^-(k)$ e, da restrição de pai único (Eq. 2.2), tem-se $\sum_{a \in \delta^-(k)} z_a = 1$. Agora, fixemos $\boldsymbol{z}$; podemos interpretar a minimização na Eq. 2.9 como um problema de corte mínimo (*min-cut*) com fonte 0 e destino k, em que a capacidade de cada arco a é dada por z_a. A restrição impõe que o valor do corte mínimo é 1; pelo teorema "min-cut max-flow", este valor é igual ao da solução do problema equivalente de fluxo máximo (*max-flow*); as restrições de capacidade e da conservação do fluxo são justamente as restrições expressas nas Eqs. 2.4–2.6. ■

2.3 Formulação Linear Inteira

Agora que estabelecemos uma representação poliedral para o polítopo das arborescências, temos o caminho aberto para formular a análise de dependências como um programa linear inteiro.

Modelo Factorizado em Arcos. Como vimos na Secção 1.2, para este modelo básico já existem algoritmos de inferência eficientes, com tempos assimptóticos de $O(L^3)$ [15, 22]. No entanto, a formulação que vamos descrever servirá de introdução para o que faremos de seguida, quando adicionarmos partes globais. Para cada arco a, tem-se um vector de características $\boldsymbol{f}_a(x)$, sendo o preço desse arco, s_a, dado pelo produto interno deste vector com o vector de parâmetros $\boldsymbol{w}$, *i.e.*, $s_a := \boldsymbol{w} \cdot \boldsymbol{f}_a(x)$. Definindo $\boldsymbol{s} := (s_a)_{a \in \mathcal{A}}$, obtemos o programa linear

$$\text{maximize} \quad \boldsymbol{s} \cdot \boldsymbol{z} \quad \text{w.r.t.} \quad \boldsymbol{z} \in \mathcal{Z}_{\text{mc}}; \tag{2.10}$$

o qual em termos de desigualdades lineares (e adicionando as variáveis de fluxo) toma a forma

$$\text{maximize} \quad \boldsymbol{s} \cdot \boldsymbol{z} \quad \text{w.r.t.} \quad \boldsymbol{z} \in [0,1]^{\mathcal{A}},\ \boldsymbol{\phi} \geq \boldsymbol{0} \quad \text{s.t.} \quad \mathbf{A} \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\phi} \end{bmatrix} \leq \boldsymbol{b}, \tag{2.11}$$

onde $\mathbf{A}$ é uma matriz de restrições, com $O(L^3)$ elementos não nulos, e $\boldsymbol{b}$ é o vector de restrições, encapsulando as restrições na Eq. 2.2 e nas Eqs. 2.4–2.6. Nenhuma restrição inteira é necessária, visto que, em virtude da Proposição 2.1, a solução deste programa é garantidamente inteira.

Características de Segunda Ordem: Pares de Arcos. McDonald et al. [54] e Carreras [12] mostram claros benefícios de se incorporar no modelo características de segunda ordem (no caso, irmãos e avós, cf. Figura 2.1). Infelizmente, como vimos na Secção 1.2, a inclusão destas características torna o problema NP-completo [55], obrigando ao emprego de métodos aproximados. É aqui que a nossa formulação linear se reveste de utilidade: se a estendermos para acomodar características globais e de seguida relaxarmos as restrições inteiras, obteremos um problema aproximado, conciso, que pode ser resolvido em tempo polinomial.

Seja $\mathcal{P} \subseteq \mathcal{A}^2$ um conjunto de pares de arcos $\mathcal{P}$. Em particular, consideramos os seguintes pares, todos eles ilustrados na Figura 2.1: *arcos irmãos arbitrários*, da forma (h, m) e (h, s); *arcos avós*, da forma (h, m) e (g, h); e *arcos de bigrama de núcleos*, da forma (h, m) e $(h', m+1)$. Cada um destes grupos contém $O(L^3)$ arcos, portanto $|\mathcal{P}| = O(L^3)$. Para cada par $(a, b) \in \mathcal{P}$, assumimos dispôr de um preço s_{ab} calculado a partir das características da frase e dos parâmetros do modelo; este preço contribui para o objectivo se os arcos a and b forem ambos incluídos. Adoptando uma estratégia de linearização para optimização pseudo-booleana [9], adicionamos novas variáveis $\boldsymbol{z}_{\mathcal{P}} := (z_{ab})_{(a,b) \in \mathcal{P}}$, bem como as seguintes restrições para cada $(a, b) \in \mathcal{P}$:

$$z_{ab} \leq z_a, \quad z_{ab} \leq z_b, \quad z_{ab} \geq z_a + z_b - 1. \tag{2.12}$$

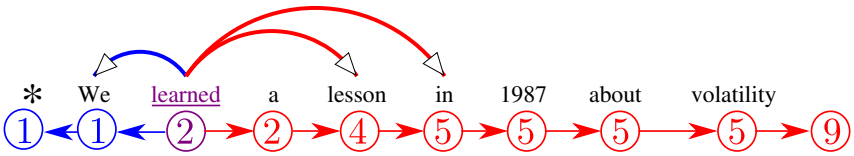

Figura 2.4: Autómatos de núcleo para a palavra *learned* na frase da Figura 2.2, vistos como modelos sequenciais. Representa-se a azul o autómato esquerdo e a vermelho o autómato direito. Os estados de cada modelo (numerais nos círculos) indicam a posição do último modificador.

Compondo tudo isto com o problema de optimização na Eq. 2.10, obtemos

$$\begin{array}{lll} \text{maximize} & \sum_{a\in\mathcal{A}} s_a z_a + \sum_{(a,b)\in\mathcal{P}} s_{ab} z_{ab} & \\ \text{w.r.t.} & \boldsymbol{z} \in \mathcal{Z}_{\text{mc}},\ \boldsymbol{z}_{\mathcal{P}} \in [0,1]^{|\mathcal{P}|} & \\ \text{s.t.} & z_{ab} \leq z_a,\ z_{ab} \leq z_b,\ z_{ab} \geq z_a + z_b - 1, \quad \text{para cada } (a,b) \in \mathcal{P}. & \end{array} \tag{2.13}$$

Trata-se ainda de um programa linear com $O(L^3)$ variáveis e restrições. Porém, apesar de $\mathcal{Z}_{\text{mc}}$ coincidir com o polítopo das arborescências $\mathcal{Z}_{\text{tree}}$, a inclusão das variáveis e restrições adicionais fazem com que o programa linear possa ter soluções fraccionais. Para garantir a solução exacta do problema, teríamos que adicionar a restrição de integralidade $\boldsymbol{z} \in \mathbb{Z}^{|\mathcal{A}|}$. Este facto não é surpreendente, dado que o problema original é NP-completo [55]. Não obstante, a relaxação linear do problema (ignorando a restrição de integralidade) conduzirá a um analisador sintáctico aproximado muito eficaz, como se verá adiante.

Características de Segunda Ordem: Autómatos de Núcleo. A técnica acima permite incorporar características para arcos irmãos *arbitrários* mas não necessariamente *consecutivos*. Isto contrasta com modelos anteriormente propostos [23, 45], os quais fazem uso de *autómatos de núcleo* para indicar este tipo de relações, detectando quando m é o *filho mais próximo* de h (no lado esquerdo ou no direito), ou quando (h, m) e (h, s) são *irmãos consecutivos* no mesmo lado.[3] Seja h a posição do núcleo e consideremos os seus potenciais modificadores no lado direito. Define-se as seguintes variáveis indicadoras de irmãos consecutivos, para cada h, m, s com $0 \leq h < m < s \leq L$:

$$z^{\text{cs}}_{hms} \quad := \quad \begin{cases} 1 & \text{se } (h,m) \text{ e } (h,s) \text{ são irmãos consecutivos,} \\ 0 & \text{caso contrário.} \end{cases} \tag{2.14}$$

Por conveniência, definimos também estas variáveis para o caso em que $m = h$ (em cujo caso se indica que s é o primeiro modificador de h) e para $s = L + 1$ (indicando-se que m é o último modificador de h). À semelhança dos casos anteriores, cada uma destas variáveis é associada a um preço s^{cs}_{hms}, calculado com base em características da frase e nos parâmetros do modelo. Dado h, empilham-se estas variáveis nos vectores $\boldsymbol{z}^{\text{cs}}_{h,\rightarrow}$ e $\boldsymbol{s}^{\text{cs}}_{h,\rightarrow}$, e analogamente para o lado esquerdo, originando $\boldsymbol{z}^{\text{cs}}_{h,\leftarrow}$ e $\boldsymbol{s}^{\text{cs}}_{h,\leftarrow}$. Um *autómato de núcleo* é um procedimento que, dado h, encontra a sequência de modificadores que maximiza o lucro (Figura 2.4). Para o autómato direito, trata-se de encontrar a sequência de estados mais provável $(m_h, \ldots, m_{L+1})$ num modelo sequencial cujos possíveis estados para a variável aleatória M_j são $\{h, \ldots, j\}$; o evento $M_j = a$ significa que "o último modificador, até à posição j, é a." Por conveniência, introduzem-se a variáveis binárias ω_{haj} que indicam a veracidade deste evento. Entre as palavras j e $j+1$, apenas

[3] Estas características não são nem mais nem menos expressivas do que as de irmãos arbitrários: enquanto as últimas permitem correlacionar modificadores que estão longe, eventualmente em lados opostos do núcleo, sendo por isso particularmente úteis em linguagens de ordem livre, as primeiras olham apenas para modificadores consecutivos no mesmo lado do núcleo, mas modelam o evento de nenhum outro modificador existir entre eles—isto é geralmente referido como *markovização horizontal*.

duas transições são possíveis: ou $M_{j+1} = m_j$ (o que sucede se $j+1$ não for um modificador), ou $M_{j+1} = j+1$ (o que sucede no caso oposto). Utilizando a maquinaria dos modelos gráficos (Capítulo 3) é possível representar o problema de optimização inerente ao autómato de núcleo $(h, \rightarrow)$ como a maximização de $\sum_{j=h+1}^{L} s_{hj} z_{hj} + \sum_{j=h}^{L} \sum_{k=j+1}^{L} s^{\text{cs}}_{hjk} z^{\text{cs}}_{hjk}$ sujeito às restrições:

$$\omega_{hii} = z_{hi}; \quad \sum_{a=0}^{i} \omega_{hai} = 1; \quad \sum_{a=0}^{i} z^{\text{cs}}_{ha(i+1)} = z_{h(i+1)}; \quad \forall i \in \{1, \ldots, L\}, \tag{2.15}$$

$$\omega_{ha(i+1)} + z^{\text{cs}}_{ha(i+1)} = \omega_{hai}; \quad \omega_{hai} \geq 0; \quad z^{\text{cs}}_{ha(i+1)} \geq 0; \quad \forall i \in \{1, \ldots, L\}, a \in \{1, \ldots, i\}. \tag{2.16}$$

Seja $\boldsymbol{z}_h := (z_{hm})_{m=h+1}^{L}$ e $\boldsymbol{\omega}_h := (\omega_{ham})_{h \leq a \leq m \leq L}$. Define-se o seguinte polítopo:

$$\mathcal{Z}_{\text{head},h,\rightarrow} := \left\{ (\boldsymbol{z}_h, \boldsymbol{z}^{\text{cs}}_{h,\rightarrow}) \;\middle|\; \exists \boldsymbol{\omega} \;:\; (\boldsymbol{z}_h, \boldsymbol{z}^{\text{cs}}_{h,\rightarrow}, \boldsymbol{\omega}_h) \text{ satisfaz Eqs. 2.15–2.16} \right\}. \tag{2.17}$$

Este polítopo é isomorfo ao polítopo marginal do modelo gráfico sequencial descrito acima. Os seus vértices são inteiros e correspondem às configurações válidas das variáveis $(\boldsymbol{z}_h, \boldsymbol{z}^{\text{cs}}_{h,\rightarrow})$. Define-se o *polítopo dos autómatos de núcleo* como o produto euclidiano de todos estes polítopos:

$$\mathcal{Z}_{\text{head}} := \textstyle\prod_{h=0}^{L} \mathcal{Z}_{\text{head},h,\leftarrow} \times \prod_{h=0}^{L} \mathcal{Z}_{\text{head},h,\rightarrow}. \tag{2.18}$$

Resumindo, para incorporar características de irmãos consecutivos na nossa formulação linear inteira basta adicionar as $O(L^3)$ variáveis e restrições nas Eqs. 2.15–2.16.

Características de Caminho Direccionado. Uma das mais-valias do modelo de fluxo multi-mercadorias introduzido na Secção 2.2 é a sua capacidade de representar relações de descendência: se um arco (i, j) transportar a k-ésima mercadoria (*i.e.*, se $\phi^k_{ij} = 1$), então é porque existe na árvore um caminho direccionado de j para k. Podemos então definir variáveis $\boldsymbol{z}^{\text{path}} := (z^{\text{path}}_{jk})_{j,k}$ indicando a existência destes caminhos. Precisamos apenas de adicionar as seguintes restrições:

$$z^{\text{path}}_{jk} = \textstyle\sum_{a \in \delta^-(j)} \phi^k_a, \quad j, k \in \mathcal{N} \setminus \{0\}, \quad z^{\text{path}}_{0k} = 1, \quad k \in \mathcal{N} \setminus \{0\}. \tag{2.19}$$

Com estas variáveis, podemos incorporar características que se activam quando duas palavras têm uma relação de descendência, originando um preço no objectivo que premeia ou penaliza as árvores para as quais essa relação se verifica. Tal como as características de irmãos arbitrários podem ser vistas como uma forma de quebrar a propriedade de Markov horizontal, estas características quebram a propriedade de Markov vertical das características de avós (Figura 2.1).

Características de Arco Não-Projectivo. Muitas linguagens têm a propriedade sintáctica de "preferir" árvores projectivas, evitando arcos não-projectivos (cf. [11]). Para dotar o modelo da capacidade de promover esta preferência quando essa propriedade é observada nos dados de treino, incorporamos características que se activam em *arcos não-projectivos*. Para esse efeito, introduzem-se indicadores $\boldsymbol{z}^{\text{np}} := (z^{\text{np}}_a)_{a \in \mathcal{A}}$, em que

$$z^{\text{np}}_a := \begin{cases} 1 & \text{se } a \in y \text{ e } a \text{ é não-projectivo,} \\ 0 & \text{caso contrário.} \end{cases} \tag{2.20}$$

Este evento pode ser capturado através das variáveis indicadoras de caminho direccionado. Por definição de arcos projectivos (Secção 2.1), temos que $z^{\text{np}}_a = 1$ se e só se o arco a está instalado na árvore ($z_a = 1$) e existe algum nó k na extensão de $a = (i, j)$ tal que $z^{\text{path}}_{ik} = 0$. Assumindo sem perda de generalidade que $i < j$, isto pode ser expresso através das seguintes $O(L^3)$ restrições para cada $(i, j) \in \mathcal{A}$:

$$z^{\text{np}}_{ij} \leq z_{ij}; \quad z^{\text{np}}_{ij} \leq -\sum_{k=i+1}^{j-1} z^{\text{path}}_{ik} + |j - i| - 1; \quad z^{\text{np}}_{ij} \geq z_{ij} - z^{\text{path}}_{ik}; \quad \forall k \in \{i, \ldots, j\}. \tag{2.21}$$

Capítulo 3

Relaxações Lineares e Turbo Parsers

Este capítulo introduz uma relaxação linear da formulação apresentada no Capítulo 2. Esta relaxação é interpretada à luz dos *modelos gráficos probabilísticos*, estabelecendo-se a sua equivalência com um método de inferência aproximada para grafos com ciclos.

Os modelos gráficos [62, 49, 80, 41] permitem uma representação compacta de distribuições de probabilidade, sendo amplamente utilizados em áreas como visão computacional, processamento de linguagem natural, bioinformática e teoria da comunicação. Neste trabalho, focamo-nos numa classe de modelos gráficos não direccionados designados por *grafos de factores* [46], os quais se assemelham a redes de Markov mas são mais informativos, pois representam explicitamente os factores de uma distribuição. Em particular, consideramos grafos de factores *com restrições*, os quais têm sido aplicados em códigos correctores de erros [29, 6] e, mais recentemente, no processamento de linguagem natural [72, 53].

Os conceitos introduzidos neste capítulo têm aplicações que não se limitam ao problema de análise sintáctica que temos vindo a tratar, sendo relevantes para qualquer problema de *classificação estruturada* [48, 75, 3, 77]: dado um conjunto de objectos observáveis $\mathcal{X}$ e um conjunto de classes $\mathcal{Y}$, que se assume ser de grande cardinalidade e possuir *estrutura*, quer-se estimar (*aprendizagem*) e avaliar (*inferência*) uma função $h : \mathcal{X} \to \mathcal{Y}$, através da qual podem classificar-se novas observações. Neste trabalho, debruçamo-nos sobre o problema de *inferência*, em que a "estrutura" de $\mathcal{Y}$ é capturada pelo modelo gráfico.

3.1 Grafos de Factores Probabilísticos com Restrições Lógicas

Um *grafo de factores* [46] é um grafo bipartido $\mathcal{G} := (\mathcal{V} \cup \mathcal{F}, \mathcal{E})$, em que $\mathcal{V}$ e $\mathcal{F}$ são conjuntos disjuntos de nós, chamados respectivamente *variáveis* e *factores*, e $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{F}$ é um conjunto de arestas ligando variáveis a factores. Para cada factor $\alpha \in \mathcal{F}$, define-se o conjunto de vizinhos $\mathcal{N}(\alpha) := \{i \in \mathcal{V} \mid (i, \alpha) \in \mathcal{E}\}$, cuja cardinalidade é o *grau* do factor, $\deg(\alpha) := |\mathcal{N}(\alpha)|$. Assumimos que cada variável $i \in \mathcal{V}$ tem um factor unário $\alpha \in \mathcal{F}$, *i.e.*, para o qual $\mathcal{N}(\alpha) = \{i\}$.

Seja $\boldsymbol{Y} := (Y_i)_{i \in \mathcal{V}}$ um grupo de variáveis aleatórias, cada uma tomando valores no conjunto finito $\mathcal{Y}_i$. Diz-se que a função de probabilidade $P(\boldsymbol{Y})$ factoriza de acordo com $\mathcal{G}$ se puder escrever-se como

$$P(\boldsymbol{Y} = \boldsymbol{y}) \;\propto\; \textstyle\prod_{\alpha \in \mathcal{F}} \psi_\alpha(\boldsymbol{y}_\alpha), \tag{3.1}$$

onde cada $\psi_\alpha : \mathcal{Y}_\alpha \to \mathbb{R}_+$ é uma *função de potencial* (não-negativa) e $\boldsymbol{Y}_\alpha$ é a variável aleatória que toma valores no produto euclidiano $\mathcal{Y}_\alpha := \prod_{i \in \mathcal{N}(\alpha)} \mathcal{Y}_i$. A Figura 3.1 mostra um exemplo.

Um factor β diz-se *restritivo* se existir pelo menos uma configuração $\boldsymbol{y}_\beta \in \mathcal{Y}_\beta$ para a qual a função de potencial se anula, $\psi_\beta(\boldsymbol{y}_\beta) = 0$. Sem perda de generalidade, assume-se que todo o factor restritivo tem uma função de potencial *binária* (ou seja, tomando valores em $\{0, 1\}$). Assim sendo, designa-se por $\mathcal{S}_\beta := \{\boldsymbol{y}_\beta \mid \psi_\beta(\boldsymbol{y}_\beta) = 1\}$ o *conjunto de aceitação* do factor; qualquer configuração fora deste conjunto terá probabilidade zero. Um factor α que não seja restritivo diz-se *suave*. Nesse caso, a função de potential ψ_α é estritamente positiva e pode escrever-se como $\psi_\alpha(\boldsymbol{y}_\alpha) := \exp(\theta_\alpha(\boldsymbol{y}_\alpha))$, sendo cada $\theta_\alpha(\boldsymbol{y}_\alpha)$ um *log-potencial*. Temos portanto uma partição $\mathcal{F} = \mathcal{F}_{\text{hard}} \cup \mathcal{F}_{\text{soft}}$, onde $\mathcal{F}_{\text{soft}}$ são os factores suaves e $\mathcal{F}_{\text{hard}}$ os restritivos. Define-se ainda o

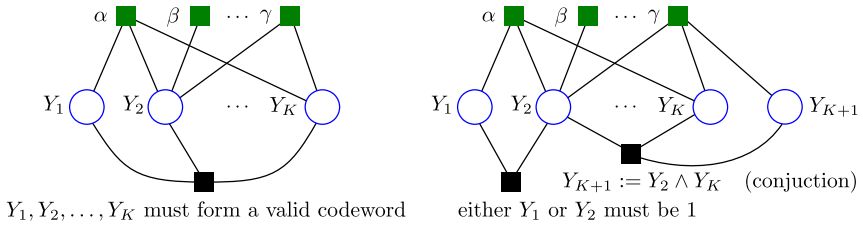

Figura 3.1: Grafos de factores. Representam-se variáveis como círculos e factores como quadrados, dos quais os verdes são suaves e os pretos restritivos. Por simplicidade, omitem-se os factores unários. No grafo da esquerda, um factor restritivo global tem como conjunto de aceitação uma lista de configurações válidas. No da direita, inclui-se uma restrição lógica e adiciona-se uma variável binária Y_{K+1}, que é a conjunção de duas existentes, para aumentar o poder expressivo.

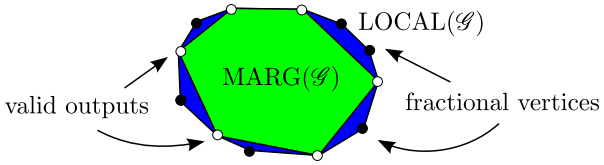

Figura 3.2: Polítopo marginal (a verde) e a sua aproximação exterior, o polítopo local (a azul). Cada elemento do polítopo marginal corresponde a uma distribuição $P(\boldsymbol{Y})$, e cada vértice corresponde a uma configuração válida $\boldsymbol{y} \in \mathcal{Y}$, tendo coordenadas em $\{0, 1\}$. O polítopo local pode apresentar, para além destes vértices, outros com coordenadas fraccionais, em $[0, 1]$.

vector de parâmetros do grafo, $\boldsymbol{\theta} := (\theta_\alpha(.))_{\alpha \in \mathcal{F}_{\text{soft}}}$, e o *conjunto das configurações possíveis*, $\mathcal{Y} := \{\boldsymbol{y} \mid \boldsymbol{y}_\beta \in \mathcal{S}_\beta, \forall \beta \in \mathcal{F}_{\text{hard}}\}$. Pode então reescrever-se (3.1) como:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}) \;=\; Z(\boldsymbol{\theta})^{-1} \exp\left(\textstyle\sum_{\alpha \in \mathcal{F}_{\text{soft}}} \theta_\alpha(\boldsymbol{y}_\alpha)\right), \quad \text{para } \boldsymbol{y} \in \mathcal{Y}, \tag{3.2}$$

designando-se $Z(\boldsymbol{\theta}) := \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\sum_{\alpha \in \mathcal{F}_{\text{soft}}} \theta_\alpha(\boldsymbol{y}_\alpha)\right)$ por *funcão de partição*.

Dado um grafo $\mathcal{G}$ parameterizado, existem três típicos problemas de inferência:

1. Inferência *maximum a posteriori* (MAP)—obter a configuração de máxima probabilidade:

$$\widehat{\boldsymbol{y}} := \arg\max_{\boldsymbol{y}} P_{\boldsymbol{\theta}}(\boldsymbol{y}) \;=\; \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \textstyle\sum_{\alpha \in \mathcal{F}} \theta_\alpha(\boldsymbol{y}_\alpha). \tag{3.3}$$

2. Calcular a função de partição $Z(\boldsymbol{\theta})$.

3. Inferência marginal—calcular as distribuições marginais *a posteriori*:

$$\mu_i(y_i) := P_{\boldsymbol{\theta}}(Y_i = y_i) \quad \text{e} \quad \mu_\alpha(\boldsymbol{y}_\alpha) := P_{\boldsymbol{\theta}}(\boldsymbol{Y}_\alpha = \boldsymbol{y}_\alpha). \tag{3.4}$$

Se o grafo não possuir ciclos (*i.e.*, se for uma árvore ou floresta) os três problemas podem ser resolvidos de forma eficiente com programação dinâmica [5, 78]; se existirem ciclos, pode empregar-se o algoritmo da árvore de junções [49], cuja complexidade é linear na "largura de árvore" do grafo (em inglês, *treewidth*); infelizmente, em muitos casos de interesse (incluindo o que aqui abordamos) este parâmetro tem crescimento exponencial com o tamanho do grafo, tornando o algoritmo impraticável. Uma alternativa é o algoritmo de *propagação de crenças* (*belief propagation*, [62]), que consiste em passar mensagens entre variáveis e factores até atingir um ponto fixo; na presença de ciclos, o algoritmo é aproximado e não tem garantias de convergência.

Polítopo Marginal. Na Eq. 3.2, parameterizámos uma família de distribuições através dos log-potenciais $\boldsymbol{\theta}$. Existe porém uma parameterização alternativa, em termos das *marginais* $\boldsymbol{\mu}$ definidas na Eq. 3.4. Cada distribuição naquela família tem uma interpretação geométrica como um ponto no *polítopo marginal* [80], podendo os problemas de inferência acima mencionados ser formulados como problemas de optimização neste polítopo.

Seja $\mathcal{R} := \{(\alpha, \boldsymbol{y}_\alpha) \mid \alpha \in \mathcal{F}_{\text{soft}}, \boldsymbol{y}_\alpha \in \mathcal{Y}_\alpha\}$ o conjunto das *partes* de $\mathcal{G}$. Diz-se que uma parte $(\alpha, \boldsymbol{y}_\alpha)$ está activa em $\bar{\boldsymbol{y}} \in \mathcal{Y}$ se $\boldsymbol{y}_\alpha = \bar{\boldsymbol{y}}_\alpha$; cada configuração global $\bar{\boldsymbol{y}} \in \mathcal{Y}$ pode ser representada como um vector binário $\boldsymbol{\chi}(\bar{\boldsymbol{y}}) \in \mathbb{R}^{|\mathcal{R}|}$ indicando as partes activas, ou seja, $[\boldsymbol{\chi}(\bar{\boldsymbol{y}})]_{(\alpha,\boldsymbol{y}_\alpha)} = 1$ se $\boldsymbol{y}_\alpha = \bar{\boldsymbol{y}}_\alpha$ e 0 caso contrário. O *polítopo marginal* de $\mathcal{G}$ [80], designado por $\text{MARG}(\mathcal{G})$, é o fecho convexo de todos os possíveis vectores binários, varrendo todas as configurações válidas:

$$\text{MARG}(\mathcal{G}) := \text{conv}\{\boldsymbol{\chi}(\boldsymbol{y}) \mid \boldsymbol{y} \in \mathcal{Y}\}. \tag{3.5}$$

A importância do polítopo marginal advém de dois factos: (i) cada vértice de $\text{MARG}(\mathcal{G})$ corresponde a uma potencial configuração em $\mathcal{Y}$; (ii) cada ponto em $\text{MARG}(\mathcal{G})$ corresponde a um vector de probabilidades marginais que é realizável. Isto é ilustrado na Figura 3.2.

Inferência MAP. Atente-se no problema de inferência MAP (Eq. 3.3). Recorrendo-se às variáveis marginais, e dada a correspondência biunívoca entre os vértices de $\text{MARG}(\mathcal{G})$ e os elementos de $\mathcal{Y}$, pode reescrever-se esse problema combinatório como um *programa linear*:

$$\boxed{\begin{array}{lll} \textbf{MAP:} & \text{maximize} & \boldsymbol{\theta} \cdot \boldsymbol{\mu} \\ & \text{w.r.t.} & \boldsymbol{\mu} \in \text{MARG}(\mathcal{G}). \end{array}} \tag{3.6}$$

A equivalência entre as duas formulações é devida ao facto de que qualquer programa linear cujo poliedro de restrições é não-vazio e limitado—o que é o caso de $\text{MARG}(\mathcal{G})$—tem uma solução num vértice. Infelizmente, para um grafo geral $\mathcal{G}$, esta reformulação não torna o problema mais fácil. Se é verdade que o teorema de Minkowsky-Weyl [68] garante uma representação de $\text{MARG}(\mathcal{G})$ em termos de um número finito de desigualdades lineares, a conjectura $P \neq NP$ implica que, em geral, este número cresce superpolinomialmente com o tamanho do grafo $\mathcal{G}$. Ainda assim, o programa linear na Eq. 3.6 tem grande utilidade teórica, pois abre portas para que se desenvolvam algoritmos de inferência aproximada considerando relaxações de $\text{MARG}(\mathcal{G})$.

3.2 Inferência Aproximada e Relaxação Linear

Polítopo Local. Recorde-se que os elementos do polítopo marginal são os vectores de marginais $\boldsymbol{\mu}$ que são realizáveis por alguma distribuição $P(\boldsymbol{Y})$. Na presença de ciclos, torna-se difícil especificar as restrições de consistência global que tornam um vector realizável. Há, contudo, certas restrições de natureza local que são simples de identificar:

1. **Não-negatividade.** Todas as marginais $\mu_i(y_i)$ e $\mu_\alpha(\boldsymbol{y}_\alpha)$ têm que ser não-negativas.

2. **Normalização.** Deve ter-se $\sum_{y_i \in \mathcal{Y}_i} \mu_i(y_i) = 1$ para cada $i \in \mathcal{V}$, e $\sum_{\boldsymbol{y}_\alpha \in \mathcal{Y}_\alpha} \mu_\alpha(\boldsymbol{y}_\alpha) = 1$ para cada $\alpha \in \mathcal{F}_{\text{soft}}$, assegurando que *localmente* todas as marginais estão normalizadas.

3. **Marginalização.** Cada marginal $\boldsymbol{\mu}_i(y_i)$ tem que ser consistente com as marginais dos factores $\alpha \in \mathcal{F}_{\text{soft}}$ em que participa, *i.e.*, deve ter-se $\mu_i(y_i) = \sum_{\boldsymbol{y}_\alpha \sim y_i} \mu_\alpha(\boldsymbol{y}_\alpha)$.[1]

4. **Anulação em configurações proibidas.** As marginais nos factores restritivos $\beta \in \mathcal{F}_{\text{hard}}$ devem anular-se fora do conjunto de aceitação, *i.e.*, deve ter-se $\mu_\beta(\boldsymbol{y}_\beta) = 0$ se $\boldsymbol{y}_\beta \notin \mathcal{S}_\beta$.

[1] A notação "$\boldsymbol{y}_\alpha \sim y_i$" significa que a soma é sobre todos os $\boldsymbol{y}_\alpha$ consistentes com y_i.

Juntando todas as peças (e eliminando restrições redundantes), formamos o *polítopo local*:

$$\mathrm{LOCAL}(\mathscr{G}) = \left\{ \boldsymbol{\mu} \geq \mathbf{0} \;\middle|\; \begin{array}{ll} \sum_{y_i \in \mathcal{Y}_i} \mu_i(y_i) = 1, & \forall i \in \mathcal{V} \\ \mu_i(y_i) = \sum_{\boldsymbol{y}_\alpha \sim y_i} \mu_\alpha(\boldsymbol{y}_\alpha), & \forall \alpha \in \mathcal{F}_{\mathrm{soft}},\ i \in \mathcal{N}(\alpha),\ y_i \in \mathcal{Y}_i \\ \mu_\beta(\boldsymbol{y}_\beta) = 0, & \forall \beta \in \mathcal{F}_{\mathrm{hard}},\ \boldsymbol{y}_\beta \notin \mathcal{S}_\beta \end{array} \right\}. \quad (3.7)$$

Note-se que o número de restrições que define $\mathrm{LOCAL}(\mathscr{G})$ é apenas *linear* (em $|\mathcal{R}|$).[2] Como qualquer vector realizável de marginais tem que satisfazer as restrições locais (3.7), temos

$$\mathrm{MARG}(\mathscr{G}) \subseteq \mathrm{LOCAL}(\mathscr{G}). \quad (3.8)$$

Por outras palavras, $\mathrm{LOCAL}(\mathscr{G})$ é uma *aproximação exterior* de $\mathrm{MARG}(\mathscr{G})$. Esta aproximação é exacta se o grafo não contiver ciclos. No caso geral, temos o seguinte resultado mais fraco:

$$\mathrm{MARG}(\mathscr{G}) = \mathrm{conv}(\mathrm{LOCAL}(\mathscr{G}) \cap \mathbb{Z}^{|\mathcal{R}|}). \quad (3.9)$$

Inferência LP-MAP. A razão pela qual o polítopo $\mathrm{LOCAL}(\mathscr{G})$ é importante é que se relaciona com métodos de inferência aproximada, que podem ser vistos como optimização sobre aquele polítopo [84, 79]. No caso da inferência MAP (Eq. 3.6), a relaxação do conjunto de restrições sugerido pela Eq. 3.8 conduz ao seguinte problema linear:

$$\boxed{\begin{array}{lll} \textbf{LP-MAP:} & \text{maximize} & \boldsymbol{\theta} \cdot \boldsymbol{\mu} \\ & \text{w.r.t.} & \boldsymbol{\mu} \in \mathrm{LOCAL}(\mathscr{G}). \end{array}} \quad (3.10)$$

Sendo uma relaxação, a solução deste problema é um majorante do valor óptimo do problema MAP. Para resolver o problema LP-MAP, é desejável obter algoritmos eficientes e capazes de tirar partido da estrutura do grafo; na literatura, têm sido propostos vários baseados em passagem de mensagens e em decomposições [79, 42, 31, 43]; no Capítulo 4 proporemos um algoritmo alternativo. Análises empíricas recentes mostram uma clara vantagem destes algoritmos especializados face a *software* genérico de optimização linear, em vários problemas padrão [83].

3.3 Factores de Restrição Lógica e Seus Polítopos Marginais

Introduzimos de seguida um conjunto de factores restritivos sobre variáveis binárias. Apesar da sua simplicidade, estes factores constituem peças elementares através das quais pode impôr-se qualquer restrição lógica de primeira ordem. Visto que cada um destes factores realiza uma função lógica, representamo-los graficamente como *portas lógicas* (Figura 3.3).

Restrições de Unicidade: Factor XOR. O factor XOR envolve $K \geq 1$ variáveis binárias e é definido através da seguinte função de potencial:

$$\psi_{\mathrm{XOR}}(y_1, \ldots, y_K) := \begin{cases} 1 & \text{se } \exists! k \in \{1, \ldots, K\} \text{ t.q. } y_k = 1 \\ 0 & \text{caso contrário,} \end{cases} \quad (3.11)$$

em que o símbolo $\exists!$ significa "existe um e um só." O conjunto de aceitação deste factor, $\mathcal{S}_{\mathrm{XOR}}$, é constituído pelas configurações em que *exactamente uma* das variáveis de entrada toma o valor 1. Trata-se de uma restrição de *unicidade*, a qual pode ser escrita em lógica de primeira ordem

[2] As restrições 1–3, para um factor suave $\alpha \in \mathcal{F}_{\mathrm{soft}}$, determinam precisamente o polítopo marginal do subgrafo $\mathscr{G}_\alpha$ constituído apenas pelo factor α e as suas variáveis vizinhas; para um factor restritivo β, pode ver-se o polítopo marginal do subgrafo $\mathscr{G}_\beta$ como o fecho convexo do seu conjunto de aceitação (restrição 4). Assim, o polítopo local $\mathrm{LOCAL}(\mathscr{G})$ é apenas a intersecção de polítopos marginais mais pequenos "levantados" para o espaço ambiente.

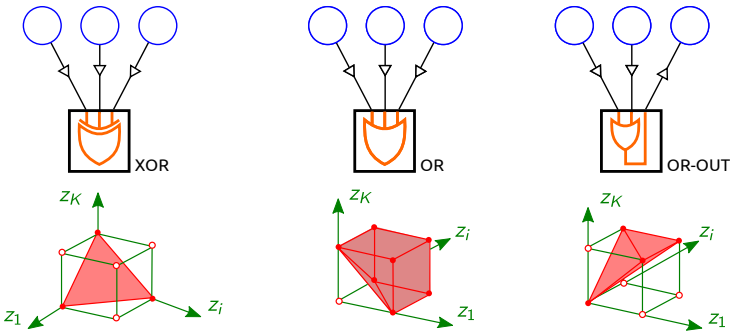

Figura 3.3: Factores lógicos restritivos considerados neste trabalho e os seus polítopos marginais.

como "$\exists! y : Q(y)$". O nome XOR advém do facto de (3.11) generalizar a função OU-exclusivo para $K \geq 2$. O polítopo marginal associado ao factor XOR é, por definição, o fecho convexo do seu conjunto de aceitação, o que resulta no *simplex de probabilidades* (Figura 3.3):

$$\boxed{\mathcal{Z}_{\text{XOR}} \;:=\; \text{conv}\, \mathcal{S}_{\text{XOR}} \;=\; \left\{ \boldsymbol{z} \in [0,1]^K \;\middle|\; \textstyle\sum_{k=1}^K z_k = 1 \right\}}. \tag{3.12}$$

Restrições de Existência: Factor OR. O factor OR representa a disjunção de $K \geq 1$ variáveis binárias. É definido através da seguinte função de potencial:

$$\psi_{\text{OR}}(y_1, \ldots, y_K) := \begin{cases} 1 & \text{se } \exists k \in \{1, \ldots, K\} \text{ s.t. } y_k = 1 \\ 0 & \text{caso contrário,} \end{cases} \tag{3.13}$$

onde o símbolo $\exists$ tem o significado usual "existe pelo menos um." A diferença em relação ao factor XOR é que o factor OR impõe uma restrição de *existência* mas não de unicidade; a expressão em lógica de primeira ordem é "$\exists y : Q(y)$". O seu polítopo marginal é (Figura 3.3):

$$\boxed{\mathcal{Z}_{\text{OR}} \;:=\; \text{conv}\, \mathcal{S}_{\text{OR}} \;=\; \left\{ \boldsymbol{z} \in [0,1]^K \;\middle|\; \textstyle\sum_{k=1}^K z_k \geq 1 \right\}}. \tag{3.14}$$

Negações. Os factores acima podem ser generalizados para acomodar variáveis *negadas*. A capacidade de lidar com negações dota estes factores de um grau adicional de flexibilidade. Considere-se por exemplo o seguinte factor que expressa uma *relação de implicação*:

$$\psi_{\text{IMPLY}}(y_1, \ldots, y_K, y_{K+1}) \;:=\; \begin{cases} 1 & \text{if } \left(\bigwedge_{k=1}^K y_k\right) \Rightarrow y_{K+1} \\ 0 & \text{caso contrário.} \end{cases} \tag{3.15}$$

Pelas leis de De Morgan, tem-se que $\psi_{\text{IMPLY}}(y_1, \ldots, y_K, y_{K+1}) = \psi_{\text{OR}}(\neg y_1, \ldots, \neg y_K, y_{K+1})$, pelo que este factor pode ser construído a partir de um OR negando as primeiras K variáveis. Se α for um factor restritivo com polítopo marginal $\mathcal{Z}_\alpha$, e β for o factor que se obtém de α negando a k-ésima variável, então o polítopo $\mathcal{Z}_\beta$ obtém-se de $\mathcal{Z}_\alpha$ aplicando uma transformação de simetria:

$$\mathcal{Z}_\beta \;=\; \left\{ \boldsymbol{z} \in [0,1]^K \;\middle|\; (z_1, \ldots, z_{k-1}, 1 - z_k, z_{k+1}, \ldots, z_K) \in \mathcal{Z}_\alpha \right\}. \tag{3.16}$$

Por indução, pode generalizar-se este resultado para um número arbitrário de variáveis negadas.

Factores Lógicos com Saída. Os factores acima referidos tomam um grupo de variáveis $y_1, \ldots, y_K$ e aplicam uma restrição. Outra possibilidade (ilustrada na Figura 3.1, à direita) é definir uma nova variável (chamemos-lhe y_{K+1}) à qual se atribui o valor de uma função lógica

envolvendo as primeiras. Este processo permite, por exemplo, introduzir "restrições suaves" cuja violação é permitida mas que tem um custo associado. Um exemplo é o factor XOR-com-saída:

$$\psi_{\text{XOR}-\text{out}}(y_1,\ldots,y_K,y_{K+1}) := \begin{cases} 1 & \text{se } y_{K+1}=1 \wedge \exists! k \in \{1,\ldots,K\} : y_k = 1 \\ 1 & \text{se } y_{K+1}=0 \wedge \forall k \in \{1,\ldots,K\} : y_k = 0 \\ 0 & \text{caso contrário,} \end{cases} \tag{3.17}$$

Este factor pode ser expresso como um factor XOR normal *em que a última variável está negada*: $\psi_{\text{XOR}-\text{out}}(y_1,\ldots,y_K,y_{K+1}) = \psi_{\text{XOR}}(y_1,\ldots,y_K,\neg y_{K+1})$. Invocando a Eq. 3.16, temos:

$$\boxed{\mathcal{Z}_{\text{XOR}-\text{out}} \;=\; \left\{ \boldsymbol{z} \in [0,1]^{K+1} \;\middle|\; \textstyle\sum_{k=1}^{K} z_k = z_{K+1} \right\}}. \tag{3.18}$$

Outro importante factor lógico com saída é o factor OR-com-saída:

$$\psi_{\text{OR}-\text{out}}(y_1,\ldots,y_K,y_{K+1}) := \begin{cases} 1 & \text{se } y_{K+1} = \bigvee_{k\in\{1,\ldots,K\}} y_k \\ 0 & \text{caso contrário.} \end{cases} \tag{3.19}$$

Este factor obriga a que a variável de saída (y_{K+1}) indique a existência de uma variável activa na entrada (entre $y_1,\ldots,y_K$), escrevendo-se como uma expressão em lógica de primeira ordem do tipo "$R(x) := \exists z : Q(x,z)$". Corresponde-lhe o seguinte polítopo marginal (Figura 3.3):

$$\boxed{\mathcal{Z}_{\text{OR}-\text{out}} \;:=\; \operatorname{conv} \mathcal{S}_{\text{OR}-\text{out}} \;=\; \left\{ \boldsymbol{z} \in [0,1]^{K} \;\middle|\; \textstyle\sum_{k=1}^{K} z_k \geq z_{K+1},\; z_k \leq z_{K+1}, \forall k \right\}}. \tag{3.20}$$

3.4 Representação Gráfica do Modelo de Análise Sintáctica

Podemos agora recorrer à maquinaria dos modelos gráficos probabilísticos para interpretar o analisador sintáctico introduzido no Capítulo 2. Apesar da natureza declarativa da formulação ali apresentada, vamos ver que existe um modelo gráfico com ciclos que lhe é subjacente; e que o problema de inferência LP-MAP neste grafo é *precisamente* a relaxação linear daquela formulação. Assim, o analisador sintáctico resultante é um "*turbo parser*", pois assenta num método de inferência aproximada que despreza os efeitos globais causados pelos ciclos.

Descrição das Variáveis e Factores do Grafo. Smith e Eisner [72] introduziram o grafo de factores reproduzido na Figura 3.4. Designamos esse modelo gráfico por *grafo de factores baseado em árvore* e denotamo-lo por $\mathcal{G}_{\text{tree}}$. As variáveis de $\mathcal{G}_{\text{tree}}$ são todas binárias, cada qual correspondendo a um potencial arco (h,m); existe assim um número quadrático de variáveis (com relação ao comprimento L da frase).[3] Existe um factor restritivo TREE, ligado a todas estas variáveis, impondo que a sua configuração conjunta defina uma árvore de dependências válida. Existem ainda $O(L)$ factores HEAD representando autómatos de núcleo, e $O(L^3)$ factores suaves ligando certos pares de variáveis (arcos irmãos e arcos avós).[4]

A nossa formulação do Capítulo 2 está associada a um modelo gráfico alternativo $\mathcal{G}_{\text{flow}}$, o qual designamos por *grafo de factores baseado em fluxos* (Figura 3.4); vamos agora descrever este modelo gráfico em detalhe. Comecemos por enunciar as suas variáveis (todas binárias):

- *variáveis para arcos*, $\boldsymbol{z} := (z_a)_{a\in\mathcal{A}}$, representando potenciais arcos na árvore. Trata-se das mesmas variáveis que figuram no grafo $\mathcal{G}_{\text{tree}}$ (Figura 3.4, em cima);

[3] Note-se que aludimos aqui a dois grafos de natureza diferente que não devem ser confundidos: um é o *grafo de factores*, cujas variáveis representam arcos $h \to m$; outro é a *árvore de dependências*, cujos nós são palavras.

[4] Smith e Eisner [72] utilizam o algoritmo de propagação de crenças com ciclos como método de inferência marginal aproximada; em publicações anteriores, mostrámos como o analisador resultante é também um *turbo parser* e escrevemos explicitamente o problema variacional nele tratado; omitimos detalhes por razões de espaço.

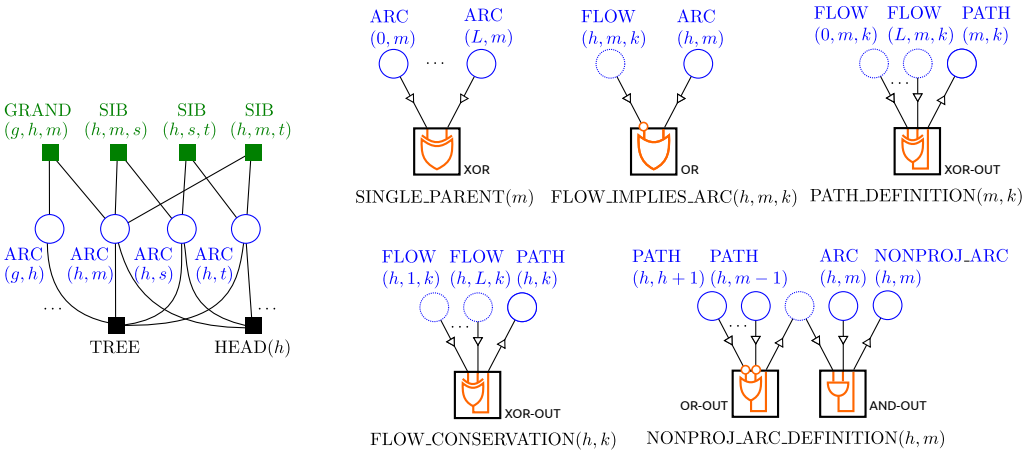

Figura 3.4: À esquerda: excerto do grafo de factores $\mathscr{G}_{\text{tree}}$ proposto por Smith e Eisner [72]. À direita: os vários factores restritivos que compõem o grafo de factores $\mathscr{G}_{\text{flow}}$ correspondendo à formulação do Capítulo 2 (omitimos os factores associados aos autómatos de núcleo).

- *variáveis para irmãos consecutivos*, $\boldsymbol{z}^{\text{cs}} := (z^{\text{cs}}_{hms})_{h,m,s}$, cada qual representando um par de irmãos consecutivos (h,m) e (h,s);
- *variáveis para caminhos direccionados*, $\boldsymbol{z}^{\text{path}} := (z^{\text{path}}_{ij})_{i,j=0}^{L}$, em que cada z^{path}_{ij} representa o evento da palavra i ser um antecedente da palavra j na árvore de dependências;
- *variáveis para arcos não-projectivos*, $\boldsymbol{z}^{\text{np}} := (z^{\text{np}}_{a})_{a\in\mathcal{A}}$, em que cada z^{np}_{a} indica que a é um arco não-projectivo na árvore de dependências.
- *variáveis de fluxo*, $\boldsymbol{\phi} := (\phi^{k}_{a})_{a\in\mathcal{A},k=1,\ldots,L}$, representando o valor dos fluxos multi-mercadoria;
- *variáveis dos autómatos de núcleo*, $\boldsymbol{\omega} := (\omega_{ham})_{h,a,m}$, representando estados do autómato (ver Secção 2.3).

Enunciemos agora os factores de $\mathscr{G}_{\text{flow}}$. Os factores suaves representam interacções entre pares de arcos; são os mesmos dos do grafo $\mathscr{G}_{\text{tree}}$ para *irmãos arbitrários* e *avós* (GRAND e SIB na Figura 3.4), acrescentando-se factores para *bigramas de núcleo*. Através de uma tabela de contingências, é fácil verificar que o polítopo marginal associado a cada um destes $O(L^3)$ factores suaves é dado pelo mesmo conjunto de restrições que escrevêramos na Eq. 2.12.

Para impôr as restrições do programa linear inteiro, utilizam-se factores restritivos que substituem os factores TREE e HEAD de $\mathscr{G}_{\text{tree}}$. Alguns são factores unários que atribuem valores designados a algumas variáveis: $z^{\text{path}}_{0k} = z^{\text{path}}_{kk} = 1, \forall k$ (*cada palavra descende da raiz e de si própria*) e $\phi^{h}_{(h,m)} = 0, \forall h,m$ (*arcos com origem numa palavra não transportam mercadoria para essa palavra*). O conjunto seguinte de factores equivale ao factor TREE:

- $O(L)$ factores XOR, cada um ligando os arcos $\{(h,m)\}_{h=0}^{L}$ para m fixo, assegurando que *cada palavra tem um e um só pai*; da Eq. 3.12 temos que o polítopo marginal é dado por:
$$\textstyle\sum_{h=0}^{L} z_{(h,m)} = 1, \quad m \in \{1,\ldots,L\}; \tag{3.21}$$
estas restrições são aquelas que haviam sido apresentadas na Eq. 2.2 (a restrição que impõe que a raiz não tem pai é trivialmente imposta removendo todos os arcos da forma $(h,0)$).
- $O(L^3)$ factores IMPLY, cada um impondo que *se um arco tem fluxo, então tem que estar instalado*. Estes factores equivalem a OR com a primeira variável negada, logo geram as seguintes restrições (cf. Eq. 3.12) que equivalem às que havíamos escrito na Eq. 2.6:
$$\phi^{k}_{a} \leq z_{a}, \quad a \in \mathcal{A}, k \in \{1,\ldots,L\}. \tag{3.22}$$

- $O(L^2)$ factores XOR-com-saída impondo que *cada variável de caminho* z^{path}_{mk} *está activa se há um arco (forçosamente único) em* $\{(h,m)\}_{h=0,\ldots,L}$ *transportando a* k*-ésima mercadoria.* Da Eq. 3.18, obtemos justamente as restrições da Eq. 2.19 que definem z^{path}_{mk}:

$$z^{\text{path}}_{mk} = \sum_{h=0}^{n} \phi^{k}_{(h,m)}, \quad m,k \in \{1,\ldots,L\}. \tag{3.23}$$

- $O(L^2)$ factores XOR-com-saída impondo que *cada palavra só consome a sua própria mercadoria*; *i.e.*, se $h \neq k$ e $k \neq 0$, então há um caminho de h para k se e só se há um e um só arco em $\{(h,m)\}_{m=1,\ldots,L}$ transportando a k-ésima mercadoria. Obtém-se as restrições:

$$z^{\text{path}}_{hk} = \sum_{m=1}^{n} \phi^{k}_{(h,m)}, \; h,k \in \{0,\ldots,L\}, \; k \notin \{0,h\}. \tag{3.24}$$

  É fácil de ver que estas correspondem às restrições de fluxo multi-mercadoria na Eq. 2.5 para o caso $j \neq k$, uma vez feita a substituição na Eq. 3.23.

Intersectando todos os polítopos marginais definidos nas Eqs. 3.21–3.24, obtém-se o polítopo das arborescências $\mathcal{Z}_{\text{mc}} = \mathcal{Z}_{\text{tree}}$ que introduzimos na Secção 2.2. Este polítopo é, por definição, o polítopo marginal do factor TREE. Do mesmo modo, através de factores XOR, é simples emular cada uma das restrições do núcleo de autómato que havíamos escrito nas Eqs. 2.15–2.16. Os respectivos polítopos marginais, quando intersectados, coincidem com o polítopo dos autómatos de núcleo $\mathcal{Z}_{\text{head}}$ (Secção 2.3), que por sua vez é o polítopo marginal dos factores HEAD.

O grafo $\mathcal{G}_{\text{flow}}$ inclui ainda um conjunto de factores para detectar arcos não-projectivos:

- Para cada arco $a = (h,m)$, assumindo sem perda de generalidade que $h < m$, há um factor OR-com-saída ligado a todas as variáveis de caminho negadas $\{\neg z^{\text{path}}_{hk}\}_{k=h+1}^{m-1}$, cuja saída é uma variável adicional (chamemos-lhe γ_a) indicando que *existe uma palavra* k *na extensão de* a *que não descende de* h. Da Eq. 3.20, temos que o polítopo marginal é dado por:

$$\gamma_a \leq -\sum_{k=h+1}^{m-1} z^{\text{path}}_{hk} + |m-h| - 1; \quad \gamma_a \geq 1 - z^{\text{path}}_{hk}, \quad \forall k \in \{h+1,\ldots,m-1\}. \tag{3.25}$$

- Para cada arco $a = (h,m)$, um factor AND-com-saída ligado a z_a e γ_a, cuja saída é z^{np}_a, assinalando que *a árvore contém o arco* a*, o qual é não-projectivo.* Tem-se da Eq. 3.20:

$$z^{\text{np}}_a \leq z_a; \quad z^{\text{np}}_a \leq \gamma_a; \quad z^{\text{np}}_a \geq z_a + \gamma_a - 1. \tag{3.26}$$

As restrições nas Eqs. 3.25–3.26 são equivalentes àquelas anteriormente apresentadas na Eq. 2.21.

Turbo Parsers. Façamos o ponto da situação:

- construímos um grafo de factores baseado em fluxos multi-mercadoria, $\mathcal{G}_{\text{flow}}$ (Figura 3.4);
- por construção, o problema de inferência MAP nesse grafo é equivalente ao programa linear inteiro que introduzimos no Capítulo 2;
- no problema de inferência aproximada LP-MAP, o polítopo marginal de $\mathcal{G}_{\text{flow}}$ é aproximado pelo polítopo LOCAL($\mathcal{G}_{\text{flow}}$), o qual é definido pelas restrições nas Eqs. 2.12 e 3.21–3.26.

Como vimos, as restrições nas Eqs. 2.12 e 3.21–3.26 são as mesmas da relaxação linear do programa linear inteiro. Como tal, essa relaxação equivale à aproximação LP-MAP, resultando de desprezar os efeitos globais causados pelos ciclos de $\mathcal{G}_{\text{flow}}$—trata-se do mesmo tipo de aproximação que subjaz diversos algoritmos de inferência aproximada, populares em áreas como a visão computacional, a bioinformática e a teoria dos códigos correctores de erros. O nosso método aproximado relaciona-se também com os de outros analisadores sintácticos [72, 45], os quais utilizam um subconjunto das características globais que aqui consideramos e baseiam-se no grafo de factores $\mathcal{G}_{\text{tree}}$ em vez de $\mathcal{G}_{\text{flow}}$. Pese embora as motivações diferentes destes analisadores sintácticos, todos eles podem ser vistos como *turbo-parsers*.

Capítulo 4

Algoritmo de Decomposição Dual com Direcções Alternadas

Neste capítulo, introduzimos um novo algoritmo de inferência LP-MAP, o qual designamos por AD^3 (*Alternating Directions Dual Decomposition*). Este algoritmo baseia-se no "método dos multiplicadores com direcções alternadas", um método de optimização com origem na década de 1970 [33, 26], mas que recentemente despertou o interesse da comunidade científica [1, 34, 10].

Tal como outros algoritmos de decomposição dual, de que é exemplo o *método do subgradiente projectado* [43, 69], o algoritmo AD^3 é um *algoritmo de consenso iterativo*, o qual alterna entre dois tipos de operações: uma operação de *transmissão* ("*broadcast*"), onde se distribuem subproblemas simples por uma assembleia de nós *locais* (neste caso, os factores do grafo); e uma operação de *agregação* ("*gather*"), onde se processam conjuntamente as soluções locais e se constrói uma solução *global*, ajustando-se multiplicadores no sentido de promover um *consenso*.

Ao contrário do método do subgradiente, o algoritmo AD^3, na sua operação de transmissão, *informa* os nós locais acerca da solução global obtida na iteração anterior, permitindo que estes *regularizem* os seus subproblemas em torno dessa solução. Como consequência, atingem-se consensos muito mais rapidamente, sem sacrificar a modularidade e garantias de convergência dos algoritmos de decomposição dual. Algumas propriedades adicionais do algoritmo AD^3 são:

- é adequado para *paralelização massiva* (*i.e.*, com muitos componentes sobrepostos);
- é *convergente*, mesmo quando cada subproblema local é resolvido de forma aproximada;
- produz *certificados de optimalidade* para a estimativa MAP (quando a relaxação é exacta) e mantém *resíduos primais e duais*, permitindo monitorizar o processo de optimização e parar quando se atinja um nível de precisão desejado.

Mediante validação experimental em 14 línguas, mostramos como o algoritmo AD^3 resolve o problema LP-MAP de forma eficiente, permitindo incorporar nos *turbo parsers* as características globais referidas nos capítulos anteriores. Para além dos ganhos obtidos por via do maior poder expressivo, obtêm-se para os nossos *turbo parsers* tempos de execução competitivos com outros analisadores sintácticos. As velocidades de convergência são muito superiores às do algoritmo do subgradiente e competitivas com o estado-da-arte do *software* de optimização linear.

4.1 Decomposição em Partes

O algoritmo AD^3 é um algoritmo de inferência aproximada (LP-MAP, Secção 3.2) para problemas de classificação estruturada em grafos de factores. Para tornar mais clara a sua exposição, assumimos que todas as variáveis do grafo são binárias e que este descreve uma *decomposição em partes* de um modelo estatístico estruturado (à semelhança do grafo de factores da Secção 3.4). Dada uma observação $x \in \mathcal{X}$ (*e.g.*, uma *frase*), este modelo induz uma função de lucro $f : \mathcal{Y} \to \mathcal{R}$, procurando obter-se a classe $\widehat{y} \in \mathcal{Y}$ (*e.g.*, uma *análise sintáctica*) que maximiza o lucro:

$$\widehat{y} = \arg\max_{y \in \mathcal{Y}} f(y). \tag{4.1}$$

Uma causa típica de intratabilidade advém da explosão combinatória inerente na composição de dois ou mais modelos tratáveis [4]. Recentemente, Rush et al. [69] propôs um método baseado em *decomposição dual* para abordar problemas cuja função de lucro se decompõe como $f(y) = f_1(z_1) + f_2(z_2)$, onde z_1 e z_2 são duas "vistas" sobrepostas de y, reescrevendo-se a Eq. 4.1 como:

$$\text{maximize} \quad f_1(z_1) + f_2(z_2) \quad \text{w.r.t.} \quad z_1 \in \mathcal{Y}_1, z_2 \in \mathcal{Y}_2 \quad \text{s.t.} \quad z_1 \sim z_2. \tag{4.2}$$

A notação informal $z_1 \sim z_2$ significa que z_1 e z_2 "concordam nas suas partes sobrepostas". Estas noções podem ser formalizadas e generalizadas para um número *arbitrário* de modelos. É de particular interesse o cenário em que este número é muito grande e cada componente é muito simples; podemos pensar em cada componente como um *factor* num grafo de factores, e na sobreposição entre componentes como partilha de variáveis por diferentes factores.

Partes básicas. Considera-se um conjunto de *partes básicas* $\mathcal{R}$, sendo cada $y \in \mathcal{Y}$ identificado como um subconjunto de $\mathcal{R}$. Em sintaxe de dependências, podemos definir $\mathcal{R}$ como o conjunto dos potenciais arcos (Figura 2.1). Por conveniência, representa-se y como um vector binário, $y := (y(r))_{r\in\mathcal{R}}$, em que se define $y(r) = 1$ se a parte r pertence a y e 0 caso contrário.

Decomposição. Generalizamos a decomposição na Eq. 4.2 considerando conjuntos $\mathcal{Y}_1, \ldots, \mathcal{Y}_S$ com $S \geq 2$. Associa-se cada $\mathcal{Y}_s$ ao seu próprio conjunto de partes $\mathcal{R}_s$, representando-se os elementos de $\mathcal{Y}_s$ como vectores binários $z_s = (z_s(r))_{r\in\mathcal{R}_s}$. Denotamos por $\bar{\mathcal{R}}_s = \mathcal{R}_s \cap \mathcal{R}$ as partes básicas de $\mathcal{R}_s$. Assumimos ainda o seguinte:

- Conjuntamente, $\mathcal{R}_1, \ldots, \mathcal{R}_S$ cobrem $\mathcal{R}$, *i.e.*, $\mathcal{R} \subseteq \bigcup_{s=1}^{S} \mathcal{R}_s$; e apenas partes básicas se podem sobrepôr, *i.e.*, $\mathcal{R}_s \cap \mathcal{R}_t \subseteq \mathcal{R}, \forall s, t \in \{1, \ldots, S\}$;
- Cada $z_s \in \mathcal{Y}_s$ é completamente definido pelos seus componentes indexados pelos elementos de $\bar{\mathcal{R}}_s$, a partir dos quais se pode "adivinhar" aqueles em $\mathcal{R}_s \setminus \bar{\mathcal{R}}_s$.

Consistência global. Para representar o objecto $y \in \mathcal{Y}$ através da "colagem" dos vários componentes $(z_1, \ldots, z_S)$, temos que garantir que estes sejam "globalmente consistentes". Diz-se que $z_s \in \mathcal{Y}_s$ e $z_t \in \mathcal{Y}_t$ são *consistentes* (escrevendo-se $z_s \sim z_t$) se concordam nas suas partes sobrepostas, *i.e.*, se $z_s(r) = z_t(r), \forall r \in \mathcal{R}_s \cap \mathcal{R}_t$. Uma configuração $(z_1, \ldots, z_S)$ é *globalmente consistente* se todos os pares de componentes forem consistentes. Nesse caso, existe um *vector-testemunha* $(u(r))_{r\in\mathcal{R}}$ tal que $z_s(r) = u(r), \forall s, r \in \bar{\mathcal{R}}_s$. Assumindo a decomposição $f(z) = \sum_{s=1}^{S} f_s(z_s)$, o problema MAP (que generaliza a Eq. 4.2 para $S \geq 2$) pode então escrever-se:

$$\begin{array}{lll} \textbf{MAP:} & \text{maximize} & \sum_{s=1}^{S} f_s(z_s) \\ & \text{w.r.t.} & (u(r))_{r\in\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|}, \quad z_s \in \mathcal{Y}_s, \quad \forall s, \\ & \text{s.t.} & z_s(r) = u(r), \quad \forall s, r \in \bar{\mathcal{R}}_s. \end{array} \tag{4.3}$$

Designamos as restrições de igualdade na última linha por "restrições de consistência". São estas restrições que complicam o problema, porquanto sem as quais este seria separável em S subproblemas. O método da *decomposição dual* [43, 69] constrói uma aproximação (equivalente à inferência LP-MAP) livrando-se destas restrições através de multiplicadores de Lagrange.

4.2 Decomposição Dual: Método do Subgradiente Projectado

Para cada $s \in \{1, \ldots, S\}$, seja $\mathcal{Z}_s := \operatorname{conv} \mathcal{Y}_s$. Tem-se a seguinte relaxação do problema MAP (Eq. 4.3), a qual corresponde à aproximação LP-MAP induzida pelo polítopo local:

$$\begin{array}{lll} \textbf{LP-MAP (P):} & \text{maximize} & \sum_{s=1}^{S} f_s(z_s) \\ & \text{w.r.t.} & (u(r))_{r\in\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|}, \quad z_s \in \mathcal{Z}_s, \quad \forall s, \\ & \text{s.t.} & z_s(r) = u(r), \quad \forall s, r \in \bar{\mathcal{R}}_s. \end{array} \tag{4.4}$$

Introduzindo um multiplicador de Lagrange $\lambda_s(r)$ para cada restrição de consistência na Eq. 4.4, obtém-se a função lagrangiana e o problema dual

$$\mathcal{L}(z,u,\lambda) = \sum_{s=1}^{S}\left(f_s(z_s) + \sum_{r\in\bar{\mathcal{R}}_s}\lambda_s(r)z_s(r)\right) - \sum_{r\in\mathcal{R}}\left(\sum_{s:r\in\bar{\mathcal{R}}_s}\lambda_s(r)\right)u(r); \tag{4.5}$$

$$\boxed{\begin{array}{lll}\textbf{LP-MAP (D):} & \text{minimize} & \sum_{s=1}^{S} g_s(\lambda_s) \\ & \text{w.r.t.} & \lambda = (\lambda_1,\ldots,\lambda_S) \\ & \text{s.t.} & \sum_{s:r\in\bar{\mathcal{R}}_s}\lambda_s(r) = 0,\ \forall r\in\mathcal{R},\end{array}} \tag{4.6}$$

em que cada $g_s(\lambda_s)$ consiste no valor óptimo do seguinte subproblema:

$$\text{maximize}\quad f_s(z_s) + \sum_{r\in\bar{\mathcal{R}}_s}\lambda_s(r)z_s(r)\quad\text{w.r.t.}\quad z_s\in\mathcal{Z}_s. \tag{4.7}$$

Uma vantagem da formulação dual (4.6) é que os componentes $1,\ldots,S$ estão desacoplados, tornando a optimização mais fácil se cada subproblema (4.7) puder ser resolvido de forma eficiente. Nos nossos modelos, as funções de lucro são lineares, *i.e.*, da forma $f_s(z_s) = \sum_{r\in\mathcal{R}_s}\theta_s(r)z_s(r)$ para algum vector $\theta_s = (\theta_s(r))_{r\in\mathcal{R}_s}$. Em consequência, tem-se dualidade forte e além disso a Eq. 4.7 torna-se um programa linear, o qual tem uma solução num vértice de $\mathcal{Z}_s$ (que por sua vez é um elemento de $\mathcal{Y}_s$). Dependendo da estrutura do problema, a Eq. 4.7 pode ser resolvida por força bruta, programação dinâmica, ou algoritmos combinatórios especializados [69, 45].

Aplicando o método do subgradiente projectado [43, 69] ao problema da Eq. 4.6 conduz a um algoritmo extremamente simples, o qual em cada iteração t resolve os subproblemas da Eq. 4.7 para $s = 1,\ldots,S$ (*transmissão*), e depois reune estas soluções (designemo-las por z_s^{t+1}) para calcular um "voto médio" para cada parte básica (*agregação*),

$$u^{t+1}(r) = \tfrac{1}{\delta(r)}\sum_{s:r\in\bar{\mathcal{R}}_s} z_s^{t+1}(r), \tag{4.8}$$

onde $\delta(r) = |\{s : r\in\mathcal{R}_s\}|$ é o número de componentes que contém a parte r. Segue-se uma actualização dos multiplicadores de Lagrange,

$$\lambda_s^{t+1}(r) = \lambda_s^t(r) - \eta_t(z_s^{t+1}(r) - u^{t+1}(r)), \tag{4.9}$$

onde η_t é um passo iterativo. Intuitivamente, o algoritmo promove consenso através do ajustamento dos multiplicadores de Lagrange, tomando em consideração desvios do voto médio (Eq. 4.9). O método do subgradiente é garantidamente convergente para a solução do problema LP-MAP (Eq. 4.6), para uma escolha adequada da sequência de passos iterativos [7]; fornece ainda um certificado de optimalidade sempre que a relaxação for exacta (*i.e.*, MAP=LP-MAP). Caso contrário, não é possível obter certificados e a convergência pode ser muito lenta quando o número de componentes S é elevado (como mostraremos na secção experimental). Na próxima secção, introduzimos o algoritmo AD^3, que não apresenta estas desvantagens.

4.3 Decomposição Dual: Método das Direcções Alternadas

Debruçando-nos novamente sobre o problema LP-MAP (P) na Eq. 4.4, observamos que qualquer configuração viável das variáveis primais tem que satisfazer as restrições de consistência. Esta observação sugere que a penalização das violações destas restrições pode acelerar o consenso. Adicionando um termo penalizador à Eq. 4.5, obtemos a *função lagrangiana aumentada* [36, 63]:

$$\mathcal{L}_\rho(z,u,\lambda) := \mathcal{L}(z,u,\lambda) - \tfrac{\rho}{2}\sum_{s=1}^{S}\sum_{r\in\bar{\mathcal{R}}_s}(z_s(r) - u(r))^2, \tag{4.10}$$

na qual o parâmetro $\rho \geq 0$ controla a intensidade da penalização. Os métodos baseados na lagrangiana aumentada são bem conhecidos em optimização (ver, por exemplo, Bertsekas et al.

Algoritmo 1 Decomposição Dual com Direcções Alternadas (AD^3)

entrada: funções de lucro $\{f_s(.)\}_{s=1}^S$, parâmetros ρ, η, limiares ϵ_P and ϵ_D.
inicializar $t \leftarrow 1$, $u^1(r) \leftarrow 0.5$ e $\lambda_s^1(r) \leftarrow 0, \quad \forall s, \forall r \in \bar{\mathcal{R}}_s$
repeat
  for each $s = 1, \ldots, S$ **do**
    actualizar z_s, obtendo-se z_s^{t+1} (Eq. 4.12)
  end for
  actualizar u, obtendo-se u^{t+1} (Eq. 4.8) e λ, obtendo-se λ^{t+1} (Eq. 4.9)
  $t \leftarrow t + 1$
until $r_P^{t+1} < \epsilon_P$ e $r_D^{t+1} < \epsilon_D$ (Eq. 4.13)
saída: soluções do problema primal relaxado e do problema dual u, z, λ

[7], §4.2). Estes métodos alternam actualizações das variáveis λ enquanto procuram maximizar $\mathcal{L}_\rho$ com respeito a z e u. No nosso caso, esta maximização conjunta é difícil, uma vez que o termo de penalização acopla os dois grupos de variáveis. O *método dos multiplicadores com direcções alternadas*, inventado por Gabay e Mercier [26] e Glowinski e Marroco [33], obvia esta dificuldade executando aquelas maximizações alternadamente,

$$z^{t+1} = \arg\max_z \mathcal{L}_\rho(z, u^t, \lambda^t), \quad u^{t+1} = \arg\max_u \mathcal{L}_\rho(z^{t+1}, u, \lambda^t), \tag{4.11}$$

seguindo-se a actualização dos multiplicadores de Lagrange através da Eq. 4.9. Ora, a maximização com respeito a u apresenta forma fechada, que consiste justamente na operação de cálculo da média executada pelo método do subgradiente projectado (Eq. 4.8). Resta-nos o problema de maximização em ordem a z. Tal como no método do subgradiente, esta maximização é separável em S subproblemas independentes, mas que agora tomam a forma:

$$\text{maximize} \quad f_s(z_s) + \textstyle\sum_{r \in \bar{\mathcal{R}}_s} \lambda_s(r) z_s(r) - \frac{\rho}{2} \sum_{r \in \bar{\mathcal{R}}_s} (z_s(r) - u^t(r))^2 \quad \text{w.r.t.} \quad z_s \in \mathcal{Z}_s(x). \tag{4.12}$$

Comparando as Eqs. 4.7 e 4.12, observamos que a única diferença é a presença na última de um termo quadrático que serve de *regularizador* em torno da solução global na iteração anterior, $u^t(r)$. Este termo quebra a linearidade do objectivo, pelo que a Eq. 4.12 não admite, em geral, uma solução num vértice (constrastando com o método do subgradiente projectado). Na Secção 4.4, descrevemos formas exactas e eficientes de resolver este problema quando cada componente consiste num dos factores restritivos lógicos introduzidos na Secção 3.3.

Resíduos primais e duais. Recorde-se que o método do subgradiente fornece certificados de optimalidade quando a relaxação é exacta (MAP = LP-MAP) e uma solução de MAP é encontrada no decurso do algoritmo. Porém, não existem bons critérios de terminação quando a relaxação não é exacta, sendo típico parar-se o algoritmo quando se atinge um número pré-estabelecido de iterações. Em constraste, o método das direcções alternadas permite monitorizar os seguintes resíduos primal e dual [10]:

$$r_P^t = \frac{\sum_{s=1}^S \sum_{r \in \bar{\mathcal{R}}_s} (z_s^t(r) - u^t(r))^2}{\sum_{r \in \mathcal{R}} \delta(r)}; \quad r_D^t = \frac{\sum_{r \in \mathcal{R}} \delta(r)(u^t(r) - u^{t-1}(r))^2}{\sum_{r \in \mathcal{R}} \delta(r)}. \tag{4.13}$$

O nosso critério de paragem é então que estes dois resíduos estejam abaixo de um limiar. Isto permite fornecer certificados não só para a solução exacta do problema MAP (quando a relaxação é exacta), mas também terminar quando se obtém uma solução suficientemente precisa de LP-MAP. O procedimento completo é exibido como o Algoritmo 1. A convergência deste algoritmo é garantida empregando um passo fixo, $\eta_t = \tau\rho$, com $\tau \in [1, 1.618]$ [32, Thm. 4.2]. Nas nossas experiências, definimos $\tau = 1.5$, e adaptamos ρ como descrito em [10, p.20].

4.4 Resolução dos Subproblemas Quadráticos

Vamos ver como os subproblemas (4.12) associados ao algoritmo AD^3 podem ser resolvidos de forma eficiente para vários casos importantes. Assumindo funções de lucro lineares, $f_s(z_s) := \sum_{r\in\mathcal{R}_s} \theta_s(r) z_s(r)$, estes subproblemas podem ser reescritos, a menos de uma constante, como:

$$\max_{z_s\in\mathcal{Z}_s} \sum_{r\in\mathcal{R}_s\setminus\bar{\mathcal{R}}_s} \theta_s(r) z_s(r) - \frac{\rho}{2} \sum_{r\in\bar{\mathcal{R}}_s} (z_s(r) - a_s(r))^2. \tag{4.14}$$

onde $a_s(r) := u^t(r) + \rho^{-1}(\theta_s(r) + \lambda_s^t(r))$. Visto que $\mathcal{Z}_s$ é um polítopo, (4.14) é um programa quadrático. Em todos os casos que mostramos de seguida, este problema admite uma solução em forma fechada que pode ser calculada em tempo $O(|\mathcal{R}_s|)$, desprezando termos logarítmicos.

Factor Suave Com Duas Variáveis. Neste caso, temos $\mathcal{R}_{\text{PAIR}} = \{r_1, r_2, r_{12}\}$, onde r_1 e r_2 são partes básicas e r_{12} é a sua conjunção. Este factor é usado, por exemplo, nas partes para irmãos, avós e bigramas de núcleo (Figura 2.1). O problema (4.14) pode ser escrito como

$$\begin{array}{ll} \max & \theta_{12} z_{12} - \frac{\rho}{2}[(z_1 - a_1)^2 + (z_2 - a_2)^2] \\ \text{w.r.t.} & (z_1, z_2, z_{12}) \in [0,1]^3 \\ \text{s.t.} & z_{12} \leq z_1,\ z_{12} \leq z_2,\ z_{12} \geq z_1 + z_2 - 1 \end{array} \tag{4.15}$$

e tem solução em forma fechada (omitimos detalhes por razões de espaço).

Factores Restritivos: XOR, OR e OR-com-saída. No caso dos factores restritivos, o subproblema (4.14) fica equivalente ao de projectar no polítopo marginal do factor; estes polítopos estão representados na Figura 3.3 para factores XOR, OR e OR-com-saída. Por razões de espaço, omitimos detalhes e descrevemos apenas a estratégia utilizada.

O caso do factor XOR reduz-se ao de projectar no simplex de probabilidades; essa projecção pode ser calculada em tempo $O(L \log L)$ utilizando uma operação de ordenação (*sort*) seguida de limiarização suave (*soft-thresholding*) [20].

No caso do factor OR, é necessário projectar num hipercubo ao qual se subtraiu um vértice; para esse efeito, empregamos um simples procedimento de *sifting* que pode reverter para projecção no simplex, tendo a mesma complexidade do caso anterior, isto é, $O(L \log L)$.

O caso do factor OR-com-saída é menos óbvio, correspondendo-lhe o subproblema:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} \sum_{i=1}^{K+1} (z_i - a_i)^2 \\ \text{w.r.t.} & (z_1, \ldots, z_{K+1}) \in [0,1]^{K+1} \\ \text{s.t.} & z_{K+1} \geq \sum_{i=1}^{K} z_i;\ z_{K+1} \leq z_i,\ \forall i = 1, \ldots, K. \end{array} \tag{4.16}$$

Porém, este caso pode também ser resolvido de forma exacta em tempo $O(L \log L)$, combinando um método de *sifting* com uma iteração do algoritmo de Boyle-Dykstra.

Finalmente, se um factor restritivo contiver variáveis negadas (por exemplo, a i-ésima), é possível reutilizar estes procedimentos através de uma mudança de variáveis $z_i \mapsto 1 - z_i$, revertendo-se para os problemas de optimização já examinados.

Factores Maiores. A única desvantagem do algoritmo AD^3 face ao método do subgradiente é que não é óbvio como resolver o subproblema (4.12) de forma eficiente para grandes factores combinatórios, tais como o factores TREE e HEAD que vimos na Secção 3.3.[1] O algoritmo AD^3 adequa-se melhor a decomposições envolvendo subproblemas "simples", ainda que em grande quantidade. Porém, Eckstein e Bertsekas [21] mostraram que o método das direcções alternadas apresenta ainda garantias de convergência em certos casos em que o subproblema é resolvido de forma inexacta; diferimos para trabalho futuro a aplicação do algoritmo AD^3 neste cenário.

[1] Por esse motivo, é necessário empregar o grafo baseado em fluxos, descrito na Secção 3.4.

4.5 Experiências

Nesta secção, avaliamos experimentalmente o desempenho dos *turbo parsers* com os modelos e algoritmos descritos ao longo deste trabalho. Utilizámos 14 conjuntos de dados das "*shared tasks*" das conferências CoNLL-2006 e CoNLL-2008 [11, 73], cada qual contendo texto jornalístico numa língua diferente, anotado manualmente com árvores de dependências, e dividido em partições de treino e de teste. Utilizámos ainda um conjunto de dados com árvores projectivas derivado do Penn Treebank (PTB [52]) aplicando as regras de conversão de Yamada e Matsumoto [82].[2]

Para treinar os modelos, executámos 10 iterações do algoritmo *cost-augmented MIRA* [17]; este algoritmo requer vários passos de inferência, para os quais utilizámos a aproximação LP-MAP e o algoritmo AD^3. Para assegurar árvores de dependência válidas durante a fase de teste, arredondamos soluções fraccionais convertendo-as em preços e executando o algoritmo de Chu-Liu-Edmonds (porém, estas são integrais em $> 95\%$ dos casos). As partes utilizadas no nosso modelo completo são as representadas na Figura 2.1. Testámos ainda um modelo de segunda ordem apenas com arcos, irmãos consecutivos e avós que reproduz o modelo de Koo et al. [45].

Qualidade dos analisadores sintácticos. A Tabela 4.1 mostra os resultados obtidos para cada conjunto de dados. Como comparação, incluímos os melhores resultados publicados para cada língua, entre analisadores baseados em transições [60, 37], optimização global [54, 44, 45] e híbridos [61]. O nosso modelo completo superou os melhores resultados conhecidos para 7 línguas. As últimas duas linhas mostram uma melhoria consistente ao usar-se o modelo completo face ao modelo de segunda ordem (com as excepções do Chinês e do Árabe). Conduzimos ainda um estudo simples de ablação de características treinando vários modelos no PTB (§22) com diferentes grupos de características. Como esperado, o desempenho melhora à medida que utilizamos modelos com maior poder expressivo, progredindo de 91.02% (modelo factorizado em arcos) para 92.41% (modelo completo).

Rapidez de convergência e optimalidade. A Figura 4.1 compara o desempenho do algoritmo AD^3 com o algoritmo de subgradiente projectado no PTB (§22).[3] Para o modelo de segunda ordem, o método do subgradiente tem um factor TREE e vários para as partes de irmãos arbitrários, o que resulta numa média de 310.5 e um máximo de 4310 factores. Estes números são ainda comportáveis, e observamos que se atinge bom desempenho relativamente depressa. O método AD^3 tem muito mais componentes porque utiliza o grafo de factores baseado em fluxos (média 1870.8, máximo 65446), no entanto atinge optimalidade mais cedo, como pode ser observado na figura da direita. Para o modelo completo, o método do subgradiente torna-se extremamente lento e o seu desempenho degrada-se severamente (após 1000 iterações o UAS é 2% pior do que o obtido com o algoritmo AD^3). A razão é o elevado número de componentes (média 3327.4, máximo 113207). Em contraste, o método AD^3 mantém um desempenho robusto, com uma fracção larga de certificados de optimalidade nas primeiras iterações.

Tempo de execução e estratégias de *caching*. Não obstante a sua adequação a problemas com muitos componentes sobrepostos, o nosso analisador sintáctico é ainda 1.6 vezes mais lento do que o de Koo et al. [45] (0.34 contra 0.21 segundos por frase no PTB §23). A nossa implementação, contudo, não está totalmente optimizada, podendo obter-se ganhos consideráveis colocando em *cache* alguns subproblemas, como a Figura 4.2 sugere. Após um número reduzido

[2] Treinámos nas secções §02–21, usámos a secção §22 para validação, e testámos na secção §23. Utilizámos a ferramenta SVMTool [30] para obter etiquetas morfológicas nas secções §22–23.

[3] No método do subgradiente, definiu-se o passo $\eta_t = \eta_0/(1+N_{\text{incr}}(t))$, como em Koo et al. [45], em que $N_{\text{incr}}(t)$ é o número de vezes que o valor do programa dual piora até à t-ésima iteração, e η_0 é escolhido para maximizar o decréscimo no dual após 20 iterações. Estas iterações preliminares não são consideradas na Figura 4.1.

	Ara.	Bul.	Chi.	Cze.	Eng.	Dan.	Dut.	Ger.	Jap.	Por.	Slo.	Spa.	Swe.	Tur.	PTB §23
G+CS	81.1	93.0	91.1	88.8	92.5	91.7	84.8	91.3	93.6	92.1	86.1	86.0	89.9	76.2	92.2
Full	81.1	**93.5**	90.6	**89.5**	**92.7**	**91.9**	85.5	**91.9**	**93.7**	92.3	**87.0**	86.7	90.2	76.6	92.5

Tabela 4.1: Desempenho dos vários analisadores sintácticos. Mostra-se a percentagem de arcos correctamente classificada (*unlabeled attachment scores* ou UAS), excluindo pontuação. "Full" é o nosso modelo completo e "G+CS" é a nossa reprodução do modelo de Koo et al. [45]. Pontuações a negrito denotam os melhores resultados conhecidos na literatura.

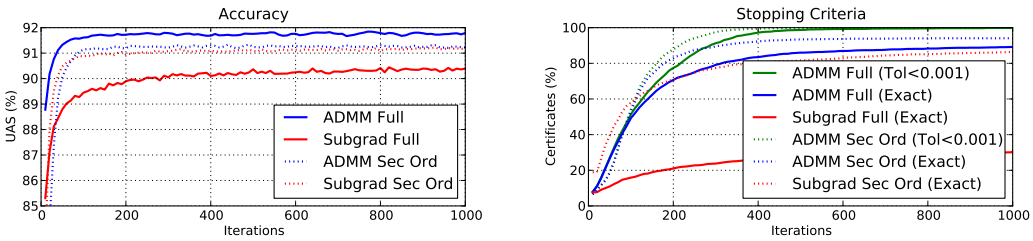

Figura 4.1: UAS com pontuação (esquerda) e fracção de certificados de optimalidade (direita) ao longo de iterações dos algoritmos de subgradiente e AD^3, no PTB §22. "Full" é o modelo completo; "Sec Ord" é um modelo de segunda ordem com avós e irmãos arbitrários, para o qual o método do subgradiente usa uma decomposição mais favorável com o factor TREE. Na figura do lado direito, mostramos também, para o AD^3, a fracção de frases que convergiram para uma solução aproximadamente óptima do problema LP-MAP (resíduos primais e duais $< 10^{-3}$).

de iterações, muitas variáveis $u(r)$ vêem um consenso ser atingido (*i.e.*, $u^t(r) = z_s^{t+1}(r), \forall s$) e entram em estado inactivo: permanecem inalteradas após a actualização na Eq. 4.8, pelo que ficam também inalterados os multiplicadores $\lambda_s^{t+1}(r)$ (Eq. 4.9). Se na t-ésima iteração todas estas variáveis estiverem inactivas, tem-se $z_s^{t+1}(r) = z_s^t(r)$, pelo que o subproblema não precisa de ser resolvido.

A mesma figura compara os tempos de execução da nossa implementação do algoritmo AD^3 com aqueles obtidos com *software* estado-da-arte de optimização linear, CPLEX, na sua configuração de melhor desempenho (dual simplex). Observa-se que o AD^3 é mais rápido em alguns regimes e mais lento noutros. Para frases curtas (< 15 palavras), o algoritmo AD^3 é quase sempre mais rápido. Para frases mais longas, o CPLEX é muito eficiente graças às boas heurísticas que utiliza para os passos de *pivot* no algoritmo simplex; todavia, por vezes fica bloqueado em problemas de maior dimensão. Note-se ainda que a nossa implementação do AD^3 não está optimizada nem foi paralelizada. O AD^3 está mais apto a paralelização do que o algoritmo simplex, pois cada iteração é composta de muitos componentes independentes; isto sugere potenciais ganhos consideráveis em ambientes *multi-core*.

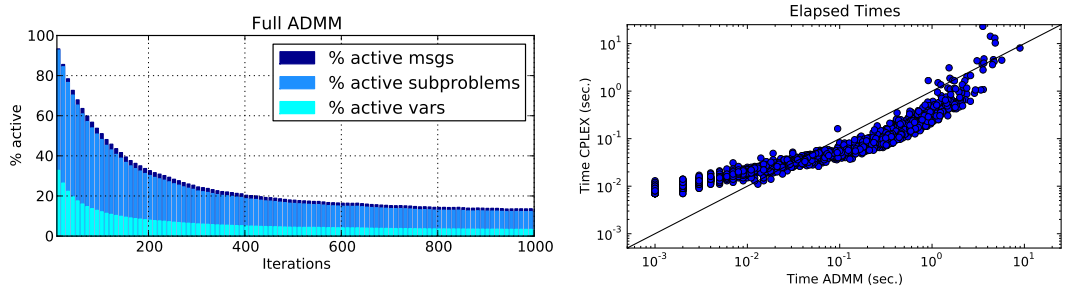

Figura 4.2: Esquerda: fracção de variáveis, subproblemas e mensagens inactivas durante as iterações do algoritmo AD^3 (modelo completo). Direita: tempos de execução do algoritmo AD^3 e do *software* CPLEX no PTB §22 (cada ponto é uma frase). Os tempos médios de execução são 0.362 (AD^3) e 0.565 segundos por frase (CPLEX).

Capítulo 5

Conclusões

Neste trabalho, propusémos novas formulações, novos modelos e um novo algoritmo para o problema da análise sintáctica de dependências.

As formulações propostas assentam num novo modelo com fluxos multi-mercadoria que permitem representar o polítopo das árvores de dependência com um número polinomial de restrições lineares, um avanço face a formulações anteriores em que este número era exponencial. Como resultado, formulámos o problema de encontrar a análise sintáctica mais provável para uma dada frase como um programa linear inteiro *conciso*.

Estas formulações abriram caminho para a introdução de novos modelos estatísticos de elevado poder expressivo, os quais incorporam uma grande quantidade de características globais que são relevantes para capturar uma variedade de fenómenos linguísticos. Dado que o respectivo problema de inferência é NP-completo, considerámos uma relaxação linear do problema e mostrámos que essa aproximação tem afinidades com uma conhecida técnica de inferência em modelos gráficos probabilísticos com ciclos, a qual ignora os efeitos globais provocados pelos ciclos e que subjaz os famosos "turbo-códigos" em teoria da comunicação. Por essa razão, denominámos os nossos analisadores sintácticos *turbo-parsers*.

Para resolver o problema linear, introduzimos um novo algoritmo de decomposição dual baseado no "método dos multiplicadores com direcções alternadas", demonstrando importantes vantagens face ao "método do subgradiente projectado" proposto anteriormente na literatura, em particular para decomposições com um elevado número de sobreposições.

Através de validação experimental em 14 línguas diferentes, verificámos que as contribuições introduzidas neste trabalho permitiram avançar o estado-da-arte no problema da análise sintáctica de texto, um problema fundamental na área do processamento de linguagem natural, imprescindível para a qualidade dos sistemas de pesquisa, tradutores automáticos, sistemas de extracção de informação e sistemas de resposta automática a perguntas, entre outras aplicações.

Este trabalho potencia várias avenidas para investigação futura, algumas das quais enumeramos de seguida. No que concerne aos modelos sintácticos, estes podem ser estendidos com novas características e restrições globais, utilizando conhecimento linguístico especializado. Outros formalismos sintácticos podem também ser abordados com técnicas similares, entre os quais gramáticas sintagmáticas com restrições, gramáticas lexicais-funcionais ou gramáticas categoriais combinatórias.

As aproximações propostas, baseadas em relaxações lineares, podem também ser refinadas considerando relaxações poliedrais ou semi-definidas mais apertadas, como as hierarquias de Sherali-Adams, de Lovász-Schrijver e de Lasserre, ou outras utilizadas em modelos gráficos probabilísticos. Algumas estratégias baseadas em "*branch-and-bound*" estão já a ser exploradas.

Finalmente, o algoritmo de decomposição dual aqui introduzido tem aplicações em modelos gráficos arbitrários, sendo relevante para problemas de outras áreas, como visão computacional, bioinformática ou teoria de códigos. Um desafio é encontrar estratégias eficientes para resolver o subproblema quadrático no caso de factores genéricos; parecem particularmente adequados os métodos de conjunto activo, os quais necessitam apenas de uma caixa preta para resolver linearizações do problema (um subproblema idêntico ao do método do subgradiente projectado) e são muito eficientes quando colocados em algoritmos iterativos, pois permitem *warm starts*.

Bibliografia

[1] M. Afonso, J. Bioucas-Dias e M. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19, 2010.

[2] S. Afonso, E. Bick, R. Haber e D. Santos. Floresta sintá (c) tica: a treebank for portuguese. In *Proc. of International Conference on Language Resources and Evaluation*. 2002.

[3] Y. Altun, I. Tsochantaridis e T. Hofmann. Hidden Markov support vector machines. In *Proc. of International Conference on Machine Learning*. 2003.

[4] Y. Bar-Hillel, M. Perles e E. Shamir. On formal properties of simple phrase structure grammars. *Language and Information: Selected Essays on their Theory and Application*, páginas 116–150, 1964.

[5] L.E. Baum e T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[6] C. Berrou, A. Glavieux e P. Thitimajshima. Near Shannon limit error-correcting coding and decoding. In *Proc. of International Conference on Communications*, volume 93, páginas 1064–1070. 1993.

[7] D. Bertsekas, W. Hager e O. Mangasarian. *Nonlinear programming*. Athena Scientific, 1999.

[8] C.M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[9] E. Boros e P.L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato e J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers, 2011.

[11] S. Buchholz e E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of International Conference on Natural Language Learning*. 2006.

[12] X. Carreras. Experiments with a higher-order projective dependency parser. In *Proc. of International Conference on Natural Language Learning*. 2007.

[13] D. Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

[14] N. Chomsky. *Aspects of the Theory of Syntax*, volume 119. The MIT press, 1965.

[15] Y. J. Chu e T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[16] J. Clarke e M. Lapata. Global Inference for Sentence Compression An Integer Linear Programming Approach. *Journal of Artificial Intelligence Research*, 31:399–429, 2008.

[17] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz e Y. Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[18] A. Culotta e J. Sorensen. Dependency tree kernels for relation extraction. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. 2004.

[19] Y. Ding e M. Palmer. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. 2005.

[20] J. Duchi, S. Shalev-Shwartz, Y. Singer e T. Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proc. of International Conference on Machine Learning*. 2008.

[21] J. Eckstein e D. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992.

[22] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.

[23] J.M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of International Conference on Computational Linguistics*, páginas 340–345. 1996.

[24] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A.A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J. Prager e outros. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31–3:59–79, 2010.

[25] J.R. Finkel, A. Kleeman e C.D. Manning. Efficient, feature-based, conditional random field parsing. *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2008.

[26] D. Gabay e B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976.

[27] H.N. Gabow, Z. Galil, T. Spencer e R.E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[28] H. Gaifman. Dependency systems and phrase-structure systems. *Information and Control*, 8(3):304–337, 1965.

[29] R. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, 8(1):21–28, 1962.

[30] J. Giménez e L. Marquez. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proc. of International Conference on Language Resources and Evaluation*. 2004.

[31] A. Globerson e T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *Advances in Neural Information Processing Systems*, 20, 2008.

[32] R. Glowinski e P. Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. Society for Industrial Mathematics, 1989.

[33] R. Glowinski e A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*, 9:41–76, 1975.

[34] D. Goldfarb, S. Ma e K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions (Technical Report UCLA CAM 10-02). 2010.

[35] A. Halevy, P. Norvig e F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[36] M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:302–320, 1969.

[37] L. Huang e K. Sagae. Dynamic programming for linear-time incremental parsing. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. 2010.

[38] R.A. Hudson. *Word grammar*. Blackwell Oxford, 1984.

[39] F. Jelinek. *Statistical methods for speech recognition*. MIT Press, 1997.

[40] R. Johansson e P. Nugues. Dependency-based Semantic Role Labeling of PropBank. In *Proc. of Empirical Methods for Natural Language Processing*. 2008.

[41] D. Koller e N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[42] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1568–1583, 2006.

[43] N. Komodakis, N. Paragios e G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of International Conference on Computer Vision*. 2007.
[44] T. Koo e M. Collins. Efficient third-order dependency parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. 2010.
[45] T. Koo, A. M. Rush, M. Collins, T. Jaakkola e D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proc. of Empirical Methods for Natural Language Processing*. 2010.
[46] F. R. Kschischang, B. J. Frey e H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47, 2001.
[47] S. Lacoste-Julien, B. Taskar, D. Klein e M. I. Jordan. Word alignment via quadratic assignment. In *Annual Meeting of the North American Chapter of the Association for Computational Linguistics*. 2006.
[48] J. Lafferty, A. McCallum e F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of International Conference on Machine Learning*. 2001.
[49] Steffen Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.
[50] T.L. Magnanti e L.A. Wolsey. Optimal Trees. Relatório Técnico 290-94, Massachusetts Institute of Technology, Operations Research Center, 1994.
[51] Christopher Manning e Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
[52] M.P. Marcus, M.A. Marcinkiewicz e B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.
[53] A. McCallum, K. Schultz e S. Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems*. 2009.
[54] R. McDonald, K. Lerman e F. Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of International Conference for Natural Language Learning*. 2006.
[55] R. McDonald e G. Satta. On the complexity of non-projective data-driven dependency parsing. In *Proc. of International Conference on Parsing Technologies*. 2007.
[56] R. T. McDonald, F. Pereira, K. Ribarov e J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of Empirical Methods for Natural Language Processing*. 2005.
[57] R. J. McEliece, D. J. C. MacKay e J. F. Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2), 1998.
[58] I.A. Mel'čuk. *Dependency syntax: theory and practice*. State University of New York Press, 1988.
[59] T.M. Mitchell. *Machine learning*. McGraw Hill, 1997.
[60] J. Nivre, J. Hall, J. Nilsson, G. Eryiğit e S. Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of International Conference on Natural Language Learning*. 2006.
[61] J. Nivre e R. McDonald. Integrating graph-based and transition-based dependency parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. 2008.
[62] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
[63] M. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, páginas 283–298. Academic Press, 1969.
[64] V. Punyakanok, D. Roth, W. Yih e D. Zimak. Semantic role labeling via integer linear programming inference. In *Proc. of International Conference on Computational Linguistics*. 2004.
[65] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
[66] M. Richardson e P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.
[67] S. Riedel e J. Clarke. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of Empirical Methods for Natural Language Processing*. 2006.
[68] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
[69] A. Rush, D. Sontag, M. Collins e T. Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Proc. of Empirical Methods for Natural Language Processing*. 2010.
[70] B. Schölkopf e A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.
[71] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 de *Algorithms and Combinatorics*. Springer, 2003.
[72] D. Smith e J. Eisner. Dependency parsing by belief propagation. In *Proc. of Empirical Methods for Natural Language Processing*. 2008.
[73] M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez e J. Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proc. of International Conference on Natural Language Learning*, 2008.
[74] R.E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–36, 1977.
[75] B. Taskar, C. Guestrin e D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*. 2003.
[76] L. Tesnière. *Eléments de syntaxe structurale*. Libraire C. Klincksieck, 1959.
[77] I. Tsochantaridis, T. Hofmann, T. Joachims e Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. of International Conference on Machine Learning*. 2004.
[78] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.
[79] M. Wainwright, T. Jaakkola e A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.
[80] M. Wainwright e M. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers, 2008.
[81] M. Wang, N. A. Smith e T. Mitamura. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proc. of Empirical Methods for Natural Language Processing*. 2007.
[82] H. Yamada e Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proc. of International Conference on Parsing Technologies*. 2003.
[83] C. Yanover, T. Meltzer e Y. Weiss. Linear programming relaxations and belief propagation–an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.
[84] J.S. Yedidia, W.T. Freeman e Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. Relatório Técnico TR2004-040, MERL, 2004.