Accelerating Graph-Based Dependency Parsing with Lock-Free Parallel Perceptron

Shuming Ma(✉), Xu Sun, Yi Zhang, and Bingzhen Wei

MOE Key Lab of Computational Linguistics, School of EECS,
Peking University, Beijing, China
{shumingma,xusun,zhangyi16,weibz}@pku.edu.cn

Abstract. Dependency parsing is an important NLP task. A popular approach for dependency parsing is structured perceptron. Still, graph-based dependency parsing has the time complexity of $O(n^3)$, and it suffers from slow training. To deal with this problem, we propose a parallel algorithm called parallel perceptron. The parallel algorithm can make full use of a multi-core computer which saves a lot of training time. Based on experiments we observe that dependency parsing with parallel perceptron can achieve 8-fold faster training speed than traditional structured perceptron methods when using 10 threads, and with no loss at all in accuracy.

Keywords: Dependency parsing · Lock-free · Structured perceptron

1 Introduction

Dependency parsing is an important task in natural language processing. It tries to match head-child pairs for the words in a sentence and forms a directed graph (a dependency tree). Former researchers have proposed various models to deal with this problem [1,11].

Structured perceptron is one of the most popular approaches for graph-based dependency parsing. It is first proposed by Collins [3] and McDonald et al. [9] first applied it to dependency parsing. The model of McDonald is decoded with an efficient algorithm proposed by Eisner [5] and they trained the model with structured perceptron as well as its variant Margin Infused Relaxed Algorithm (MIRA) [4,16]. It proves that MIRA and structured perceptron are effective algorithms for graph-based dependency parsing. McDonald and Pereira [11] extended it to a second-order model while Koo and Collins [6] developed a third-order model. They all used perceptron style methods to learn the parameters.

Recently, many models applied deep learning to dependency parsing. Titov and Henderson [17] first proposed a neural network model for transition-based dependency parsing. Chen and Manning [2] improved the performance of neural network dependency parsing algorithm while Le and Zuidema [7] improved

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 260–268, 2018.
https://doi.org/10.1007/978-3-319-99495-6_22

the parser with Inside-Outside Recursive Neural Network. However, those deep learning methods are very slow during training [13].

To address those issues, we hope to implement a simple and very fast dependency parser, which can at the same time achieve state-of-the-art accuracies. To reach this target, we propose a lock-free parallel algorithm called lock-free parallel perceptron. We use lock-free parallel perceptron to train the parameters for dependency parsing. Although lots of studies implemented perceptron for dependency parsing, rare studies try to implement lock-free parallel algorithms. McDonald et al. [10] proposed a distributed perceptron algorithm. Nevertheless, this parallel method is not a lock-free version on shared memory systems. To the best of our knowledge, our proposal is the first lock-free parallel version of perceptron learning.

Our contribution can be listed as follows:

- The proposed method can achieve 8-fold faster speed of training than the baseline system when using 10 threads, and without additional memory cost.
- We provide theoretical analysis of the parallel perceptron, and show that it is convergent even with the worst case of full delay. The theoretical analysis is for general lock-free parallel perceptron, not limited by this specific task of dependency parsing.

2 Lock-Free Parallel Perceptron for Dependency Parsing

The dataset can be denoted as $\{(x_i, y_i)\}_{i=1}^{n}$ while x_i is input and y_i is correct output. GEN is a function which enumerates a set of candidates $GEN(x)$ for input x. $\Phi(x, y)$ is the feature vector corresponding to the input output pair (x, y). Finally, the parameter vector is denoted as α.

In structured perceptron, the score of an input output pair is calculated as follows:

$$s(x, y) = \Phi(x, y) \cdot \alpha \tag{1}$$

The output of structured perceptron is to generate the structure y' with the highest score in the candidate set $GEN(x)$.

In dependency parsing, the input x is a sentence while the output y is a dependency tree. An edge is denoted as (i, j) with a head i and its child j. Each edge has a feature representation denoted as $f(i, j)$ and the score of edge can be written as follows:

$$s(i, j) = \alpha \cdot f(i, j) \tag{2}$$

Since the dependency tree is composed of edges, the score are as follows:

$$s(x, y) = \sum_{(i,j) \in y} s(i, j) = \sum_{(i,j) \in y} \alpha \cdot f(i, j) \tag{3}$$

$$\Phi(x, y) = \sum_{(i,j) \in y} f(i, j) \tag{4}$$

Algorithm 1. Lock-free parallel perceptron

input: Training examples $\{(x_i, y_i)\}_{i=1}^{n}$
initialize: $\alpha = 0$
repeat
  for all Parallelized threads **do**
    Get a random example (x_i, y_i)
    $y' = argmax_{z \in GEN(x)} \Phi(x, y) \cdot \alpha$
    if $(y' \neq y)$ then $\alpha = \alpha + \Phi(x, y) - \Phi(x, y')$
  end for
until Convergence

return The averaged parameters α^*

The proposed lock-free parallel perceptron is a variant of structured perceptron [12,14]. We parallelize the decoding process of several examples and update the parameter vector on a shared memory system. In each step, parallel perceptron finds out the dependency tree y' with the highest score, and then updates the parameter vector immediately, without any lock of the shared memory. In typical parallel learning setting, the shared memory should be locked, so that no other threads can modify the model parameter when this thread is computing the update term. Hence, with the proposed method the learning can be fully parallelized. This is substantially different compared with the setting of McDonald et al. [10], in which it is not lock-free parallel learning.

3 Convergence Analysis of Lock-Free Parallel Perceptron

For lock-free parallel learning, it is very important to analyze the convergence properties, because in most cases lock-free learning leads to divergence of the training (i.e., the training fails). Here, we prove that lock-free parallel perceptron is convergent even with the worst case assumption. The challenge is that several threads may update and overwrite the parameter vector at the same time, so we have to prove the convergence.

We follow the definition in Collins's work [3]. We write $\overline{GEN(x)}$ as all incorrect candidates generated by input x. We define that a training example is separable with margin $\delta > 0$ if $\exists U$ with $\|U\| = 1$ such that

$$\forall z \in \overline{GEN(x)}, U \cdot \Phi(x, y) - U \cdot \Phi(x, z) \geq \delta \tag{5}$$

Since multiple threads are running at the same time in lock-free parallel perceptron training, the convergence speed is highly related to the delay of update. Lock-free learning has update delay, so that the update term may be applied on an "old" parameter vector, because this vector may have already been modified by other threads (because it is lock-free) and the current thread does not know that. Our analysis show that the perceptron learning is still convergent, even with the worst case that all of the k threads are delayed. To our knowledge, this is the first convergence analysis for lock-free parallel learning of perceptrons.

We first analyze the convergence of the worse case (full delay of update). Then, we analyze the convergence of optimal case (minimal delay). In experiments we will show that the real-world application is close to the optimal case of minimal delay.

3.1 Worst Case Convergence

Suppose we have k threads and we use j to denote the j'th thread, each thread updates the parameter vector as follows:

$$y'_j = \underset{z \in GEN(x)}{\operatorname{argmax}} \Phi_j(x, y) \cdot \alpha \tag{6}$$

Recall that the update is as follows:

$$\alpha^{i+1} = \alpha^i + \Phi_j(x, y) - \Phi_j(x, y'_j) \tag{7}$$

Here, y'_j and $\Phi_j(x, y)$ are both corresponding to j^{th} thread while α^i is the parameter vector after i^{th} time stamp.

Since we adopt lock-free parallel setting, we suppose there are k perceptron updates in parallel in each time stamp. Then, after a time step, the overall parameters are updated as follows:

$$\alpha^{t+1} = \alpha^t + \sum_{j=1}^{k} (\Phi_j(x, y) - \Phi_j(x, y'_j)) \tag{8}$$

Hence, it goes to:

$$\begin{aligned} U \cdot \alpha^{t+1} &= U \cdot \alpha^t + \sum_{j=1}^{k} U \cdot (\Phi_j(x, y) - \Phi_j(x, y'_j)) \\ &\geq U \cdot \alpha^t + k\delta \end{aligned}$$

where δ is the separable margin of data, following the same definition of Collins [3]. Since the initial parameter $\alpha = 0$, we will have that $U \cdot \alpha^{t+1} \geq tk\delta$ after t time steps. Because $U \cdot \alpha^{t+1} \leq \|U\| \|\alpha^{t+1}\|$, we can see that

$$\|\alpha^{t+1}\| \geq tk\delta \tag{9}$$

On the other hand, $\|\alpha^{t+1}\|$ can be written as:

$$\begin{aligned} \|\alpha^{t+1}\|^2 &= \|\alpha^t\|^2 + \|\sum_{j=1}^{k} (\Phi_j(x, y) - \Phi_j(x, y'_j))\|^2 \\ &\quad + 2\alpha^t \cdot (\sum_{j=1}^{k} (\Phi_j(x, y) - \Phi_j(x, y'_j))) \\ &\leq \|\alpha^t\|^2 + k^2 R^2 \end{aligned}$$

where R is the same definition following Collins [3] such that $\Phi(x,y) - \Phi(x,y'_j) \leq R$. The last inequality is based on the property of perceptron update such that the incorrect score is always higher than the correct score (the searched incorrect structure has the highest score) when an update happens. Thus, it goes to:

$$\|\alpha^{t+1}\|^2 \leq tk^2R^2 \tag{10}$$

Combining Eqs. 10 and 9, we have:

$$t^2k^2\delta^2 \leq \|\alpha^{t+1}\|^2 \leq tk^2R^2 \tag{11}$$

Hence, we have:

$$t \leq R^2/\delta^2 \tag{12}$$

This proves that the lock-free parallel perceptron has bounded number of time steps before convergence even with the worst case of full delay, and the number of time steps is bounded by $t \leq R^2/\delta^2$ in the worst case. The worst case means that the parallel perceptron is convergent even if the update is extremely delayed, such that k threads are updating based on the same old parameter vector.

3.2 Optimal Case Convergence

In practice the worst case of extremely delayed update is not probable to happen, or at least not always happening. Thus, we expect that the real convergence speed should be much faster than this worst case bound. The optimal bound is as follows:

$$t \leq R^2/(k\delta^2) \tag{13}$$

This bound is derived when the parallel update is not delayed (i.e., the update of each thread is based on a most recent parameter vector). As we can see, in the optimal case we can get k times speed up by using k threads for lock-free parallel perceptron training. This can achieve full acceleration of training by using parallel learning.

4 Experiments

4.1 Dataset

Following prior work, we use English Penn TreeBank (PTB) [8] to evaluate our proposed approach. We follow the standard split of the corpus, using section 2-21 as training set, Sect. 22 as development set, and Sect. 23 as final test set. We implement two popular model of graph-based dependency parsing: first-order model and second-order model. We tune all of the hyper parameters in development set. The features in our model can be found in McDonald et al. [9, 11]. Our baselines are traditional perceptron, MST-Parser [9][1], and the locked version of parallel perceptron. All of the experiment is conducted on a computer with the Intel(R) Xeon(R) 3.0 GHz CPU.

[1] www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html.

4.2 Results

Table 2 shows that our lock-free method can achieve 8-fold faster speed than the baseline system, which is better speed up when compared with locked parallel perceptron. For both 1st-order parsing and 2nd-order parsing, the results are consistent that the proposed lock-free method achieves the best rate of speed up. The results show that the lock-free parallel peceptron in real-world applications is near the optimal case theoretical analysis of low delay, rather than the worst case theoretical analysis of high delay.

The experimental results of accuracy are shown in Table 1. The baseline MST-Parser [9] is a popular system for dependency parsing. Table 1 shows that our method with 10 threads outperforms the system with single-thread. Our lock system is slightly better than MST-Parser mainly because we use more feature including distance based feature – our distance features are based on larger size of contextual window.

Figure 1 shows that the lock-free parallel perceptron has no loss at all on parsing accuracy on both 1st-order and 2nd-order parsing setting, in spite of the substantial speed up of training.

Figure 2 shows that the method can achieve near linear speed up, and with almost no extra memory cost.

Table 1. Accuracy of baselines and our method.

Models	1st-order	2nd-order
MST Parser	91.60	92.30
Locked Para-Perc	91.68	**92.55**
Lock-free Para-Perc 5-thread	91.70	**92.55**
Lock-free Para-Perc 10-thread	**91.72**	92.53

Table 2. Speed up and time cost per pass of our algorithm.

Models	1st-order	2nd-order
Structured Perc	1.0 × (449 s)	1.0 × (3044 s)
Locked Para-Perc	5.1 × (88 s)	5.0 × (609 s)
Lock-free Para-Perc 5-thread	4.3 × (105s)	4.5 × (672 s)
Lock-free Para-Perc 10-thread	**8.1 × (55 s)**	**8.3 × (367 s)**

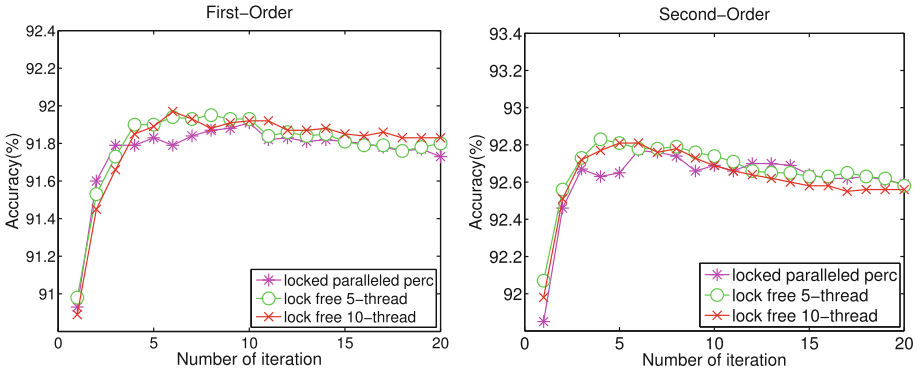

Fig. 1. Accuracy of different methods for dependency parsing.

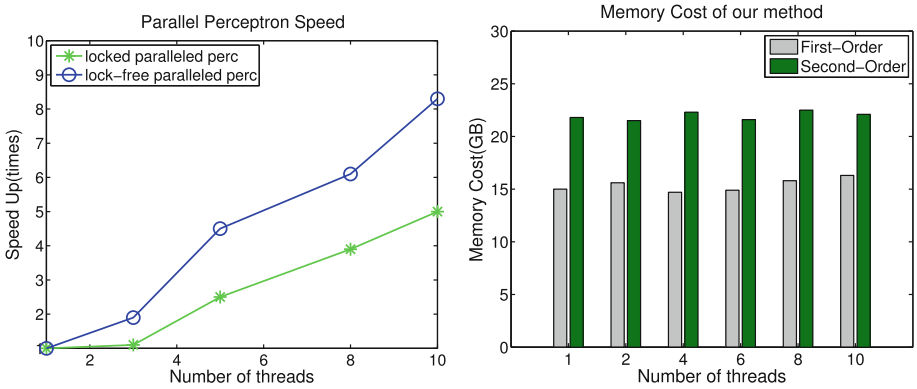

Fig. 2. Speed up and memory cost of different methods for dependency parsing.

5 Conclusions

We propose lock-free parallel perceptron for graph-based dependency parsing. Our experiment shows that it can achieve more than 8-fold faster speed than the baseline when using 10 running threads, and with no loss in accuracy. We also provide convergence analysis for lock-free parallel perceptron, and show that it is convergent in the lock-free learning setting. The lock-free parallel perceptron can be directly used for other structured prediction NLP tasks.

Acknowledgements. The authors would like to thank the anonymous reviewers for insightful comments and suggestions on this paper. This work was supported in part by National Natural Science Foundation of China (No. 61673028).

References

1. Bohnet, B.: Very high accuracy and fast dependency parsing is not a contradiction. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 89–97 (2010)
2. Chen, D., Manning, C.D.: A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 740–750 (2014)
3. Collins, M.: Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the ACL-2002 Conference on Empirical Methods in Natural Language Processing-Volume 10, pp. 1–8. Association for Computational Linguistics (2002)
4. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. **2**, 265–292 (2002)
5. Eisner, J.: Three new probabilistic models for dependency parsing: an exploration. In: Proceedings of the 16th Conference on Computational Linguistics, pp. 340–345 (1996)
6. Koo, T., Collins, M.: Efficient third-order dependency parsers. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 1–11 (2010)
7. Le, P., Zuidema, W.: The inside-outside recursive neural network model for dependency parsing. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 729–739 (2014)
8. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of English: the penn treebank. Comput. Linguist. **19**(2), 313–330 (1993)
9. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 91–98 (2005)
10. McDonald, R., Hall, K., Mann, G.: Distributed training strategies for the structured perceptron. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 456–464. Association for Computational Linguistics (2010)
11. McDonald, R.T., Pereira, F.C.N.: Online learning of approximate dependency parsing algorithms. In: 11st Conference of the European Chapter of the Association for Computational Linguistics (2006)
12. Sun, X.: Towards shockingly easy structured classification: a search-based probabilistic online learning framework. Technical report, arXiv:1503.08381 (2015)
13. Sun, X.: Asynchronous parallel learning for neural networks and structured models with dense features. In: COLING (2016)
14. Sun, X., Matsuzaki, T., Okanohara, D., Tsujii, J.: Latent variable perceptron algorithm for structured classification. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 1236–1242 (2009)
15. Sun, X., Ren, X., Ma, S., Wang, H.: meProp: sparsified back propagation for accelerated deep learning with reduced overfitting. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, pp. 3299–3308 (2017)
16. Taskar, B., Klein, D., Collins, M., Koller, D., Manning, C.D.: Max-margin parsing. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP), vol. 1, p. 3 (2004)

17. Titov, I., Henderson, J.: A latent variable model for generative dependency parsing. In: Proceedings of the 10th International Conference on Parsing Technologies, pp. 144–155 (2007)
18. Wei, B., Sun, X., Ren, X., Xu, J.: Minimal effort back propagation for convolutional neural networks. CoRR **abs/1709.05804** (2017)