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

Shuming Ma (speaker), Xu Sun, Yi Zhang, Bingzhen Wei

Peking University, Beijing, China

Introduction

- Task: **Graph-based Dependency parsing**
  - Match head-child pairs
  - Form a dependency tree
- Algorithm: **Structured perceptron**
  - Popular for dependency parsing
  - Proposed by Collins (2002) and McDonald et al. (2005)
  - Apply to first/second/third order model

Problem: **Slow** in training  Solution: **Lock-free Parallel**

Graph-based Dependency Parsing

- Edge (i, j): head is i, and child is j
- Score of edge (i, j):

$$s(i,j) = \alpha \cdot f(i,j)$$

- Sentence x, dependency tree y
- Score of dependency tree:

$$s(x,y) = \sum_{(i,j)\in y} \alpha \cdot f(i,j)$$
$$= \alpha \cdot \Phi(x,y)$$
$$\text{where } \Phi(x,y) = \sum_{(i,j)\in y} f(i,j)$$

Lock-Free Parallel Perceptron

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^*

Convergence Analysis

- Structured perceptron is proved to be convergent (Collins, 2002)
- Challenge:

  Several threads update and overwrite the parameter vector at the same time -> need to prove the convergence

Worst Case Convergence

Definition: a example is separable with margin $\delta > 0$ *if* $\exists U$ *with* $\|U\| = 1$ *such that:*

$$\forall z \in GEN(x), U \cdot \Phi(x, y) - U \cdot \Phi(x, z) \geq \delta$$

Single thread perceptron update:

$$y_j' = \underset{z \in GEN(x)}{\text{argmax}}\ \Phi_j(x, y) \cdot \alpha$$

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

Multi-thread perceptron update (full delay):

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

$$U \cdot \alpha^{t+1} = U \cdot \alpha^t + \sum_{j=1}^{k} U \cdot \Phi_j(x, y) - U \cdot \Phi_j\left(x, y_j'\right)$$

$$\geq U \cdot \alpha^t + k\delta$$

$$\|a^{t+1}\| \geq tk\delta \qquad (1)$$

Worst Case Convergence

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

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

Joint (1) and (2):

$$t^2 k^2 \delta^2 \leq \|\alpha^{t+1}\|^2 \leq t k^2 R^2$$
$$t \leq R^2 / \delta^2$$

Optimal Case Convergence

Optimal case: the convergence rate is faster than the worst case

Worse case convergence:

$$t \leq R^2/\delta^2$$

Following the previous analysis:

$$t \leq R^2/(k\delta^2)$$

Experiments

Dataset

English Penn TreeBank: we use English Penn TreeBank (PTB) to evaluate our proposed approach. We follow the standard split of the corpus, using section 2-21 as training set, section 22 as development set, and section 23 as final test set.

Baseline

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.

Our baselines are traditional perceptron, MST-Parser, and the locked version of parallel perceptron. All of the experiment is conducted on a computer with the Intel(R) Xeon(R) 3.0GHz CPU.

Results

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 1. Accuracy of baselines and our method.

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

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

Results

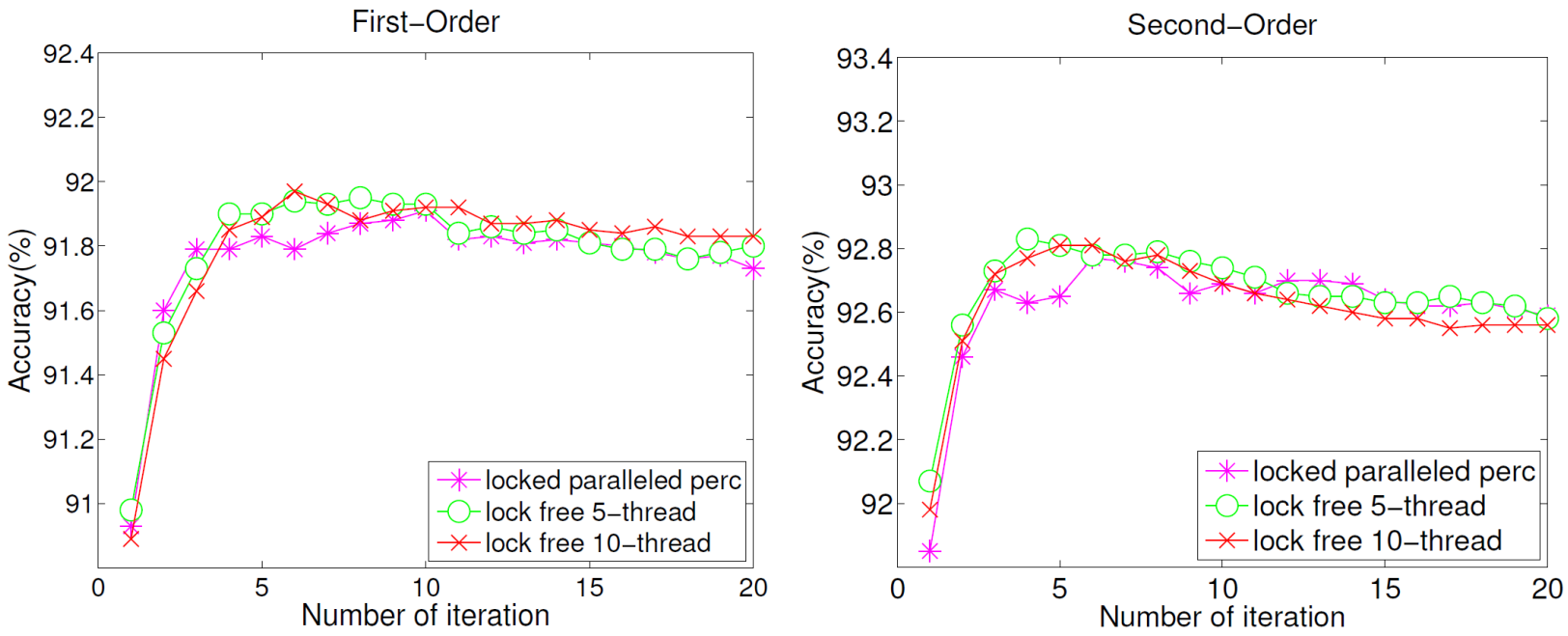

Fig. 1. Accuracy of different methods for dependency parsing.

Results

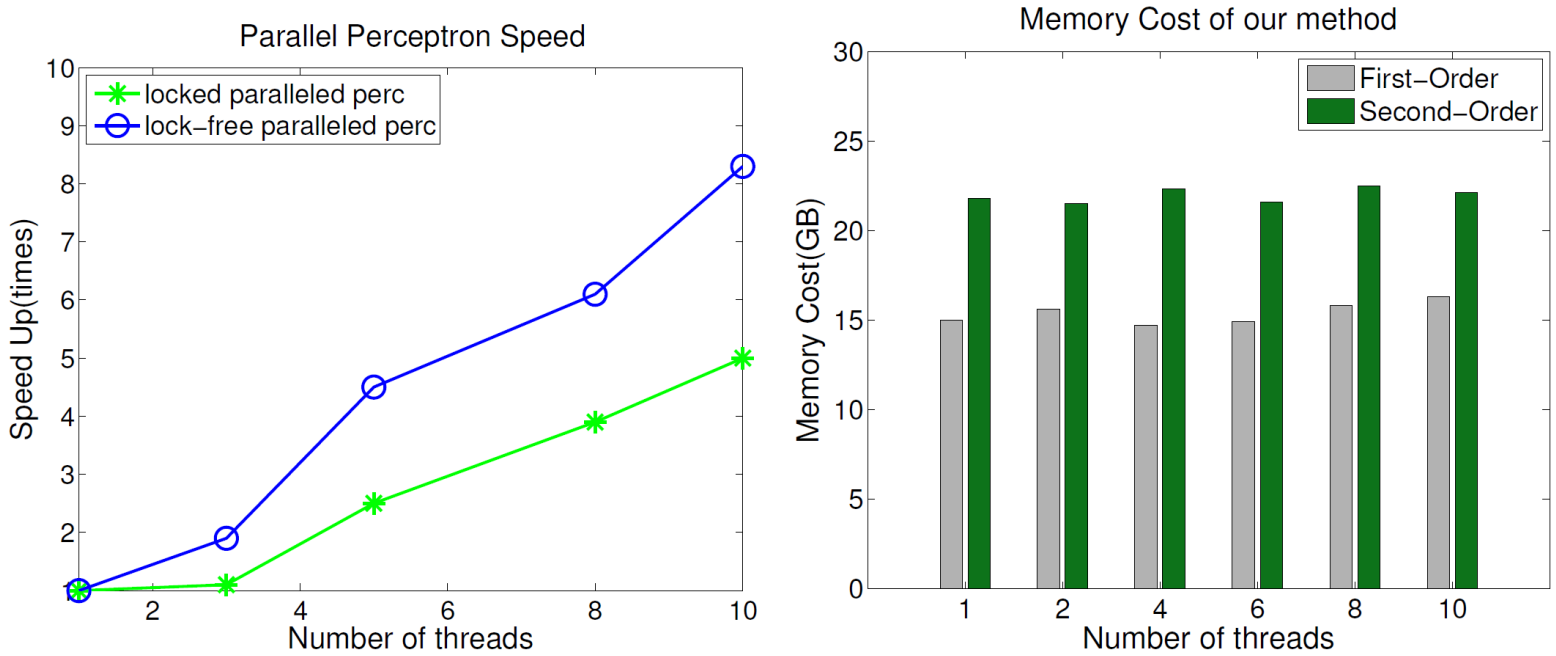

Fig. 2. Speed up and memory cost of different methods for dependency parsing.

Conclusion and Future Work

- 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.

Thank you!

北京大学 PEKING UNIVERSITY