Parallel Double Greedy Submodular Maximization

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Many machine learning problems can be formulated as maximization of submodular functions. [1] recently proposed a double greedy algorithm for unconstrained submodular maximization that achieves a tight 1/2-approximation. However, double greedy is an inherently sequential, linear time algorithm that does not scale to big data.

We present two approaches to extend the bidirectional greedy algorithm to a parallel setting. The first, 'coordination-free' approach emphasizes speed at the cost of a weaker approximation guarantee – it achieves a $(1/2 - O(1/N))$ approximation for max cut on a complete graph. The second, 'concurrency control' approach guarantees the same tight 1/2-approximation, at the cost of greater coordination.

Our parallel algorithms scale well on synthetic and real datasets, and suffer little or no loss of objective value compared to the sequential algorithm.

1 Introduction

Motivate parallel machine learning from big data setting

Briefly mention two approaches (coordination-free, concurrency control)

In this paper, we focus on unconstrained non-monotone submodular maximization.. and say briefly why this problem is important

Contributions of the paper

1. We propose two parallel algorithms for unconstrained non-monotone submodular maximization, which allows one to choose between speed and stronger approximation guarantees.

2. We prove the serial equivalence of OCC to the sequential algorithm; we analytically bound the amount of coordination of the OCC algorithm for two example problems.

3. We provide approximation guarantees for hogwild; we analytically bound the expected loss in objective value for two example problems.

4. We demonstrate empirically using two synthetic and three real datasets that our parallel algorithms perform well in terms of both speed and objective values.

The bidirectional greedy algorithm [1] gives an approximation of $E[F(A)] = 1/2f(OPT)$, where A is the algorithm output, and OPT is an optimal solution.

The hogwild algorithm can give an approximation of $E[F(A)] = \frac{1}{2}F(OPT) - \frac{1}{4}\sum_i E[\rho_i]$, where ρ_i is the maximum difference in the marginal gain that may result from not knowing the full information when deciding whether to include or exclude element i.

The OCC algorithm [XP: for the lack of a better name] guarantees an outcome that is equivalent to a sequential run of the bidirectional greedy algorithm. Theoretical properties of the bidirectional

greedy algorithm immediately translates to the OCC algorithm – in particular, the OCC algorithm gives the same approximation factor of $1/2$. In contrast to the hogwild approach, OCC introduces more coordination and thus provides less concurrency.

2 Submodular maximization

What is submodular functions?

Why is maximization an important problem?

Describe the bidirectional greedy algorithm: The sequential bidirectional greedy [1] algorithm monotonically grows A^i and shrinks B^i.

3 Approaches for parallel learning

Two approaches that allow us to trade off speed with approximation guarantees.

3.1 Coordination free

Simply run everything in parallel. Optimized for speed, but does not necessarily provide the correct answer. Requires work to prove correctness.

3.2 Concurrency control

Ensures 'serial equivalence' – the outcome of the parallel algorithm is equivalent to some execution of the sequential algorithm. Locally, threads take actions that are guaranteed to be safe (i.e. preserves serial equivalence), and forces additional coordination only when they are unable to execute their action safely. Designed for correctness, but requires coordination that compromises speed. Work is only required to demonstrate that coordination is limited.

4 Hogwild Bidirectional Greedy Algorithm

[XP: Provide more intuition for what the OCC algorithm is doing.]

Algorithm 2[1] is the hogwild parallel bidirectional greedy unconstrained submodular maximization algorithm. We associate with each element e a time T_e at which Algorithm 2 line 7 is executed, and order the elements according to the times T_e. Let $\iota(e)$ be the position of e in this ordering. This total ordering on elements also allows us to define monotonically non-decreasing sets $A^i = \{e' : e' \in A, \iota(e') < i\}$, and monotonically non-increasing sets $B^i = A^i \cup \{e' : \iota(e') \geq i\}$.

Note that in Algorithm 2, lines 5 and 6 may be executed in parallel with lines 9 and 10. Hence, $\Delta_+^{\max}(e)$ and $\Delta_-^{\max}(e)$ (lines 5 and 6) may be computed with different values of $\hat{A}(e')$. We denote by $\hat{A}_e$ and $\hat{B}_e$ respectively the vectors of $\hat{A}$ and $\hat{B}$ that are used in the computation of $\Delta_+^{\max}(e)$ and $\Delta_-^{\max}(e)$. It immediately follows that

$$\Delta_+(e) = F(A^{\iota(e)-1} \cup i) - F(A^{\iota(e)-1}), \qquad \Delta_-(e) = F(B^{\iota(e)-1}\backslash e) - F(B^{\iota(e)-1})$$
$$\Delta_+^{\max}(e) = F(\hat{A}_e \cup e) - F(\hat{A}_e), \qquad \Delta_-^{\max}(e) = F(\hat{B}_e\backslash e) - F(\hat{B}_e).$$

Lemma 4.1. *For any $e \in V$, $\hat{A}_e \subseteq A^{\iota(e)-1}$, $\hat{B}_e \supseteq B^{\iota(e)-1}$.*

Proof. Consider any element $e' \in V$. If $e' \in \hat{A}_e$, it must be the case that the algorithm set $\hat{A}(e')$ to 1 (line 9) before T_e, which implies $\iota(e') < \iota(e)$, and hence $e' \in A^{\iota(e)-1}$. So $\hat{A}_e \subseteq A^{\iota(e)-1}$. Similarly, if $e' \notin \hat{B}_e$, then the algorithm set $\hat{B}(e')$ to 0 (line 10) before T_e, so $\iota(e') < \iota(e)$. Also,

---

[1]We present only the parallelized probabilistic versions of [1]. Both parallel algorithms can be easily extended to the deterministic version of [1]; the hogwild algorithm can also be extended to the multilinear version of [1].

Algorithm 1: Serial submodular maximization

1   $A^0 = \emptyset, B^0 = V$
2   **for** $i = 1$ *to* n **do**
3     $\Delta_+(i) = F(A^{i-1} \cup i) - F(A^{i-1})$
4     $\Delta_-(i) = F(B^{i-1} \backslash i) - F(B^{i-1})$
5   Draw $u_i \sim Unif(0,1)$
6   **if** $u_i < \frac{[\Delta_+(i)]_+}{[\Delta_+(i)]_+ + [\Delta_-(i)]_+}$ **then**
7      $A^i := A^{i-1} \cup i; B^i := B^{i-1}$
8   **else**
9      $A^i := A^{i-1}; B^i := B^{i-1} \backslash i$

Algorithm 2: Hogwild bidirectional greedy

1   **for** $e \in V$ **do** $\hat{A}(e) = 0, \hat{B}(e) = 1$
2   **for** $p \in \{1, \ldots, P\}$ **do in parallel**
3   **while** $\exists$ *element to process* **do**
4      $e = $ next element to process
5      $\Delta_+^{\max}(e) = F(\hat{A} \cup e) - F(\hat{A})$
6      $\Delta_-^{\max}(e) = F(\hat{B} \backslash e) - F(\hat{B})$
7   Draw $u_e \sim Unif(0,1)$
8   **if** $u_e < \frac{[\Delta_+^{\max}(e)]_+}{[\Delta_+^{\min}(e)]_+ + [\Delta_-^{\max}(e)]_+}$ **then**
9       $\hat{A}(e) \leftarrow 1$
10   **else** $\hat{B}(e) \leftarrow 0$

Algorithm 3: Hogwild for separable sums

1   **for** $e \in V$ **do** $\hat{A}(e) = 0$
2   **for** $l = 1, \ldots, L$ **do** $\hat{\alpha}_l = 0, \hat{\beta}_l = \sum_{e \in S_l} w_l(e)$
3   **for** $p \in \{1, \ldots, P\}$ **do in parallel**
4   **while** $\exists$ *element to process* **do**
5      $e = $ next element to process
6      $\Delta_+^{\max}(e) =$
      $-\lambda v(e) + \sum_{S_l \ni e} g(\hat{\alpha}_l + w_l(e)) - g(\hat{\alpha}_l)$
7      $\Delta_-^{\max}(e) =$
      $+\lambda v(e) + \sum_{S_l \ni e} g(\hat{\beta}_l - w_l(e)) - g(\hat{\beta}_l)$
8   Draw $u_e \sim Unif(0,1)$
9   **if** $u_e < \frac{[\Delta_+^{\max}(e)]_+}{[\Delta_+^{\min}(e)]_+ + [\Delta_-^{\max}(e)]_+}$ **then**
10       $\hat{A}(e) \leftarrow 1$
11   **for** $l : e \in S_l$ **do** $\hat{\alpha}_l \leftarrow \hat{\alpha}_l + w_l(e)$
12   **else for** $l : e \in S_l$ **do** $\hat{\beta}_l \leftarrow \hat{\beta}_l - w_l(e)$

Algorithm 4: OCC bidirectional greedy

1   **for** $e \in V$ **do** $\hat{A}(e) = \tilde{A}(e) = 0, \hat{B}(e) = \tilde{B}(e) = 1$
2   **for** $i = 1, \ldots, |V|$ **do** result$(i) = 0$
3   **for** $i = 1, \ldots, |V|$ **do** processed$(i) = false$
4   $\iota = 0$
5   **for** $p \in \{1, \ldots, P\}$ **do in parallel**
6   **while** $\exists$ *element to process* **do**
7      $e = $ next element to process
8      $\tilde{A}(e) \leftarrow 1$
9      $\tilde{B}(e) \leftarrow 0$
10      $i = \iota; \iota \leftarrow \iota + 1$
11      $\Delta_+^{\min}(e) = F(\tilde{A} \cup e) - F(\tilde{A})$
12      $\Delta_+^{\max}(e) = F(\hat{A} \cup e) - F(\hat{A})$
13      $\Delta_-^{\min}(e) = F(\tilde{B} \backslash e) - F(\tilde{B})$
14      $\Delta_-^{\max}(e) = F(\hat{B} \backslash e) - F(\hat{B})$
15   Draw $u_e \sim Unif(0,1)$
16   **if** $u_e < \frac{[\Delta_+^{\min}(e)]_+}{[\Delta_+^{\min}(e)]_+ + [\Delta_-^{\max}(e)]_+}$ **then**
17   result$(i) \leftarrow 1$
18   **else if** $u_e > \frac{[\Delta_+^{\max}(e)]_+}{[\Delta_+^{\max}(e)]_+ + [\Delta_-^{\min}(e)]_+}$ **then**
19   result$(i) \leftarrow -1$
20   **wait until** $\forall j < i$, result$(j) \neq 0$
21   **if** *result*$(i) = 0$ **then** validate(p, e, i)
22   **if** *result*$(i) = 1$ **then**
23       $\hat{A}(e) \leftarrow 1$
24       $\tilde{B}(e) \leftarrow 1$
25   **else**
26       $\tilde{A}(e) \leftarrow 0$
27       $\hat{B}(e) \leftarrow 0$
28   processed$(i) = true$

Algorithm 5: validate(p, e, i)

1   **wait until** $\forall j < i$, processed$(j) = true$
2   $\Delta_+^{exact}(e) = F(\hat{A} \cup e) - F(\hat{A})$
3   $\Delta_-^{exact}(e) = F(\hat{B} \backslash e) - F(\hat{B})$
4   **if** $u_e < \frac{[\Delta_+^{exact}(e)]_+}{[\Delta_+^{exact}(e)]_+ + [\Delta_-^{exact}(e)]_+}$ **then** result$(i) \leftarrow 1$
5   **else** result$(i) \leftarrow -1$

$e' \notin A$ because the execution of line 10 excludes the execution of line 9. Therefore, $e' \notin A^{\iota(e)-1}$, and $e' \notin B^{\iota(e)-1}$. So $\hat{B}_e \subseteq B^{\iota(e)-1}$. $\qquad \square$

Corollary 4.2. *Submodularity of F implies $\Delta_+(e) \leq \Delta_+^{\max}(e)$, and $\Delta_-(e) \leq \Delta_-^{\max}(e)$.*

4.1   Hogwild for separable sums F

For some functions F, we can maintain sketches / statistics to aid the computation of $\Delta_+^{\max}$, $\Delta_-^{\max}$, and obtain the bounds given in Corollary 4.2. In particular, we consider functions of the form $F(X) = \sum_{l=1}^{L} g\left(\sum_{i \in X \cup S_l} w_l(i)\right) - \lambda \sum_{i \in X} v(i)$, where $S_l \subseteq V$ are (possibly overlapping) groups of elements in the ground set, g is a non-decreasing concave scalar function,

3

and $w_l(i)$ and $v(i)$ are non-negative scalar weights. It is easy to see that $F(X \cup e) - F(X) = \sum_{l:e \in S_l} \left[g\left(w_l(e) + \sum_{i \in X \cup S_l} w_l(i)\right) - g\left(\sum_{i \in X \cup S_l} w_l(i)\right) \right] - \lambda v(e)$. Define

$$\hat{\alpha}_l = \sum_{j \in \hat{A} \cup S_l} w_l(j), \qquad \hat{\alpha}_{l,e} = \sum_{j \in \hat{A}_e \cup S_l} w_l(j), \qquad \alpha_l^{\iota(e)-1} = \sum_{j \in A^{\iota(e)-1} \cup S_l} w_l(j).$$

$$\hat{\beta}_l = \sum_{j \in \hat{B} \cup S_l} w_l(j), \qquad \hat{\beta}_{l,e} = \sum_{j \in \hat{B}_e \cup S_l} w_l(j), \qquad \beta_l^{\iota(e)-1} = \sum_{j \in B^{\iota(e)-1} \cup S_l} w_l(j).$$

Algorithm 3 updates $\hat{\alpha}_l$ and $\hat{\beta}_l$, and computes $\Delta_+^{\max}(e)$ and $\Delta_-^{\max}(e)$ using $\hat{\alpha}_{l,e}$ and $\hat{\beta}_{l,e}$. Following arguments analogous to that of Lemma 4.1, we can show:

Lemma 4.3. *For each l and $e \in V$, $\hat{\alpha}_{l,e} \leq \alpha_l^{\iota(e)-1}$ and $\hat{\beta}_{l,e} \geq \beta_l^{\iota(e)-1}$.*

Corollary 4.4. *Concavity of g implies that Δ's computed by Algorithm 3 satisfy*

$$\Delta_+^{\max}(e) \quad \geq \quad \sum_{S_l \ni e} \left[g(\alpha_l^{\iota(e)-1} + w_l(e)) - g(\alpha_l^{\iota(e)-1}) \right] - \lambda v(e) \quad = \quad \Delta_+(e),$$

$$\Delta_-^{\max}(e) \quad \geq \quad \sum_{S_l \ni e} \left[g(\beta_l^{\iota(e)-1} - w_l(e)) - g(\beta_l^{\iota(e)-1}) \right] + \lambda v(e) \quad = \quad \Delta_-(e),$$

5 Optimistic Concurrency Control for Bidirectional Greedy Algorithm

[XP: Provide more intuition for what the OCC algorithm is doing.]

Algorithm 4[2] is the OCC bidirectional greedy algorithm. Unlike the hogwild algorithm, the OCC algorithm ensures serial equivalence by maintaining four sets $\hat{A}$, $\tilde{A}$, $\hat{B}$, $\tilde{B}$, which serve as upper and lower bounds on A and B. Each thread can determine locally if a decision to include / exclude an element can be taken safely. Otherwise, the validation process (Algorithm 5) waits until it is certain about A, B before proceeding.

The serialization order is given by $\iota(e)$, which is the value of ι at line 10 of Algorithm 4. We define $\hat{A}_e$, $\hat{B}_e$, $A^{\iota(e)-1}$, $B^{\iota(e)-1}$ as before with the hogwild algorithm, and additionally let $\tilde{A}_e$ and $\tilde{B}_e$ be the vectors of $\tilde{A}$ and $\tilde{B}$ that are used in the computation of $\Delta_+^{\min}(e)$ and $\Delta_-^{\min}(e)$. We will show that the outcome of Algorithm 4 is equivalent to the sequential algorithm executed with ordering ι.

Lemma 5.1. $\hat{A}_e \subseteq A^{\iota(e)-1} \subseteq \tilde{A}_e \backslash e$, and $\hat{B}_e \supseteq B^{\iota(e)-1} \supseteq \tilde{B}_e \cup e$.

Proof. Clearly, $e \in \tilde{B}_e \cup e$ but $e \notin \tilde{A}_e \backslash e$. By definition, $e \in B^{\iota(e)-1}$ but $e \notin A^{\iota(e)-1}$. The OCC algorithm (Algorithm 4) only modifies $\hat{A}(e)$ and $\hat{B}(e)$ on Lines 23, 27 when processing e, which occurs after the computation of the Δ's on Lines 11 - 14, so $e \in \hat{B}_e$ but $e \notin \hat{A}_e$.

Consider any $e' \neq e$. Suppose $e' \in \hat{A}_e$. This is only possible if we have processed Line 23, which implies (by Line 20) that $\forall j < \iota(e'), \text{result}(j) \neq 0$. But at the time when $\hat{A}_e$ is read (Lines 11 - 14), we have $\text{result}(e) = 0$, so it must be the case that $\iota(e') < \iota(e)$. Thus, $e' \in A^{\iota(e)-1}$.

Now suppose $e' \in A^{\iota(e)-1}$. By definition, this implies $\iota(e') < \iota(e)$ and $e \in A$, which in turn implies $\text{result}(\iota(e')) = 1$. Hence, it must be the case that we have already set $\tilde{A}(e') \leftarrow 1$ (by the ordering imposed by ι on Line 10), but never execute $\tilde{A}(e') \leftarrow 0$, so $e' \in \tilde{A}_e$.

An analogous argument shows $e' \notin \hat{B}_e \implies e' \notin B^{\iota(e)-1} \implies e' \notin \tilde{B}_e \cup e$. $\qquad\square$

Lemma 5.2. *During the validation process for element e, on Lines 2, 3 of Algorithm 5, we have $\hat{A} = A^{\iota(e)-1}$ and $\hat{B} = B^{\iota(e)-1}$.*

---

[2]The synchronization required by the OCC algorithm can be further reduced, for example, by replacing the operations from Line 20 onwards by a non-blocking function call.

Proof. At Line 2, we have guaranteed that $\forall j < \iota(e)$, processed$(j) = true$. Thus, $\forall e' : \iota(e') < \iota(e)$, $e' \in \hat{A} \iff e' \in A^{\iota(e)-1}$ and $e' \in \hat{B} \iff e' \in B^{\iota(e)-1}$. On the other hand, result$(\iota(e)) = 0$, so all e' such that $\iota(e') > \iota(e)$ are blocked on Algorithm 4 Line 20, hence $\hat{A}(e') = 0$ and $\hat{B}(e') = 1$. Thus, $\hat{A} = A^{\iota(e)-1}$ and $\hat{B} = B^{\iota(e)-1}$. □

Algorithm 4 computes

$$\Delta_+^{\min}(e) = F(\tilde{A}_e) - F(\tilde{A}_e \backslash e), \qquad \Delta_+^{\max}(e) = F(\hat{A}_e \cup e) - F(\hat{A})$$
$$\Delta_-^{\min}(e) = F(\tilde{B}_e) - F(\tilde{B}_e \cup e), \qquad \Delta_-^{\max}(e) = F(\hat{B}_e \backslash e) - F(\hat{B}).$$

Corollary 5.3. *Submodularity of F implies that the Δ's computed by Algorithm 4 satisfy:* $\Delta_+^{\min}(e) \leq \Delta_+(e) = \Delta_+^{exact}(e) \leq \Delta_+^{\max}(e)$, *and* $\Delta_-^{\min}(e) \leq \Delta_-(e) = \Delta_+^{exact}(e) \leq \Delta_-^{\max}(e)$.

5.1 Separable sums F

Analogous to the hogwild algorithm, we maintain $\hat{\alpha}_l, \hat{\beta}_l$ and additionally $\tilde{\alpha}_l = \sum_{j \in \tilde{A} \cup S_l} w_l(j)$ and $\tilde{\beta}_l = \sum_{j \in \tilde{B} \cup S_l} w_l(j)$. It can be shown that $\hat{\alpha}_{l,e} \leq \alpha^{\iota(e)-1} \leq \tilde{\alpha}_{l,e} - w_l(e)$ and $\hat{\beta}_{l,e} \geq \beta^{\iota(e)-1} \geq \tilde{\beta}_{l,e} + w_l(e)$, which then allows us to compute our bounds for Δ's.

6 Analysis of Algorithms

[XP: What is the purpose of this section? We want to say that we have a choice between a slower but serially equivalent algorithm and a faster algorithm which is not serially equivalent. Nevertheless, the slower algorithm is not too slow, and the approximation guarantee of the algorithm that is not serially equivalent is not too weak.]

6.1 Approximation of hogwild bidirectional greedy

Let F be submodular and non-negative. We assume for each e, there exists $\rho_e \geq 0$ such that $\Delta_+^{\max}(e) - \rho_e \leq \Delta_+(e)$, and $\Delta_-^{\max}(e) - \rho_e \leq \Delta_-(e)$. This is possible, for example, by defining

$$\rho_e := \max\{\Delta_+^{\max}(e) - \Delta_+(e), \Delta_-^{\max}(e) - \Delta_-(e)\}$$
$$\leq \max_{S,T \subseteq V}\{[F(S \cup e) - F(S)] - [F(S \cup T \cup e) - F(S \cup T)]\} \leq F(e)\kappa_F$$

where κ_F is the total curvature of F. Summing over e then gives us $\sum_e \rho_e \leq \kappa_F \sum_e F(e)$.

Theorem 6.1. *Let F be a non-negative (monotone or non-monotone) submodular function. The hogwild bidirectional greedy algorithm solves the unconstrained problem $\max_{A \subset V} F(A)$ with approximation $E[F(A_{hog})] \geq \frac{1}{2}F^* - \frac{1}{4}\sum_{i=1}^{n} E[\rho_i]$, where A_{hog} is the output of the algorithm, F^* is the optimal value, and ρ_i is a random variable such that $\rho_i \geq \Delta_+^{\max}(i) - \Delta_+(i)$ and $\rho_i \geq \Delta_-^{\max}(i) - \Delta_-(i)$.*

We prove the theorem in Appendix A.

Example: max graph cut. Assuming bounded delay of τ and edges with unit weight, we can bound $\sum_i E[\rho_i] \leq 2\tau \frac{\#\text{edges}}{2N}$ The approximation of the hogwild algorithm is then $E[F(A^n)] \geq = \frac{1}{2}F(OPT) - \tau \frac{\#\text{edges}}{2N}$. In sparse graphs, the hogwild algorithm is off by a small additional term, which albeit grows linearly in τ.

Example: set cover. Consider the simple set cover function, $F(A) = \sum_{l=1}^{L} \min(1, |A \cap S_l|) - \lambda|A| = |\{l : A \cap S_l \neq \emptyset\}| - \lambda|A|$, with $0 < \lambda \leq 1$. We assume that there is some bounded delay τ. Suppose also the S_l's form a partition, so each element e belongs to exactly one set. Then, $\sum_e E[\rho_e] \geq \tau + L(1 - \lambda^\tau)$, which is linear in τ but independent of N.

6.2 Correctness of OCC

Theorem 6.2. *OCC bidirectional greedy is serially equivalent to bidirectional greedy.*

Proof. We will denote by A_{seq}^i, B_{seq}^i the sets generated by the sequential algorithm, reserving A^i, B^i for sets generated by the OCC algorithm. It suffices to show by induction that $A_{seq}^i = A^i$ and $B_{seq}^i = B^i$. For the base case, $A^0 = \emptyset = A_{seq}^0$, and $B^0 = V = B_{seq}^0$. Consider any element e. The OCC algorithm includes $e \in A$ iff $u_e < [\Delta_+^{\min}(e)]_+([\Delta_+^{\min}(e)]_+ + [\Delta_-^{\max}(e)]_+)^{-1}$ on Algorithm 4 Line 16 or $u_e < [\Delta_+^{\text{exact}}(e)]_+([\Delta_+^{\text{exact}}(e)]_+ + [\Delta_-^{\text{exact}}(e)]_+)^{-1}$ on Algorithm 5 Line 4. In both cases, Corollary 5.3 implies $u_e < [\Delta_+(e)]_+([\Delta_+(e)]_+ + [\Delta_-(e)]_+)^{-1}$. By induction, $A^{\iota(e)-1} = A_{seq}^{\iota(e)-1}$ and $B^{\iota(e)-1} = B_{seq}^{\iota(e)-1}$, so the threshold is exactly that computed by the sequential algorithm. Hence, the OCC algorithm includes $e \in A$ iff the sequential algorithm includes $e \in A$. (An analogous argument works for the case where e is excluded from B.) $\quad\square$

As an immediate consequence, theoretical properties of the sequential algorithm are preserved by the OCC algorithm, including the approximation guarantees:

Theorem 6.3. *Let F be a non-negative (monotone or non-monotone) submodular function. The OCC bidirectional greedy algorithm solves the unconstrained problem $\max_{A \subset V} F(A)$ with approximation $E[F(A_{OCC})] \geq \frac{1}{2}F^*$, where A_{OCC} is the output of the algorithm, and F^* is the optimal value.*

6.3 Scalability of OCC

Whenever an element is validated, it needs to wait for all earlier elements to be processed, and it blocks all later elements from being processed. Each validation effectively constitutes a barrier to the parallel processing. Hence, the scalability of the OCC algorithm is dependent on the number of validated elements. Nevertheless, we show for a couple of example problems that the number of validated elements can be bounded. Full details of the bounds are given in Appendix B.

Example: max graph cut. Assume that there is a bounded delay τ. The expected number of validated elements is upper bounded by $\tau \frac{2\#edges}{N}$.

Example: set cover. Assuming a bounded delay τ and non-overlapping sets S_l's, the expected number of validated elements is upper bounded by 2τ.

7 Evaluation

We implemented the parallel and sequential double greedy algorithms in Java / Scala. Experiments were conducted on Amazon EC2 using one cc2.8xlarge machine, up to 16 threads, for 10 iterations.

We measured the runtime and speedup (ratio of runtime on 1 thread to runtime on p threads). For the hogwild algorithm, we measured $F(A_{hog}) - F(A_{seq})$, the difference between the objective value on the sets returned by hogwild and the sequential algorithms. We verified the correctness of the OCC algorithm by comparing the sets returned by the OCC and sequential algorithms, and also measured the fraction of elements that are validated by OCC.

We tested our parallel algorithms on the max graph cut and set cover problems with two synthetic graphs and three real datasets (Table 1). Graphs were pre-processed to remove self-loops. We found that vertices were typically indexed such that vertices close to each another in the graph were also close in their indices. To reduce this dependency, we randomly permuted the ordering of vertices. For the max graph cut problem, we removed directions on edges to obtain undirected graphs. The set cover problem is reduced to a vertex cover on the directed graph.

Due to space constraints, we only present part of our results in Figure 1, deferring full results to Appendix D. **Runtime, Speedup:** Both parallel algorithms are faster than the sequential algorithm with three or more threads, and show good speedup properties as more threads are added ($\sim$ 10x or more for all graphs and both functions). **Objective value:** The objective value of the hogwild

Graph	# vertices	# edges	Description
Erdos-Renyi	20,000,000	$\approx 2 \times 10^6$	Each edge is included with probability 5×10^{-6}.
ZigZag	25,000,000	2,025,000,000	Expander graph. The 81-regular zig-zag product between the Cayley graph on $\mathbb{Z}_{2500000}$ with generating set $\{\pm 1, \ldots, \pm 5\}$, and the complete graph K_{10}.
Friendster	10,000,000	625,279,786	Subgraph of social network. [2]
Arabic-2005	22,744,080	631,153,669	2005 crawl of Arabic web sites [3, 4, 5].
UK-2005	39,459,925	921,345,078	2005 crawl of the .uk domain [3, 4, 5].
IT-2004	41,291,594	1,135,718,909	2004 crawl of the .it domain [3, 4, 5].

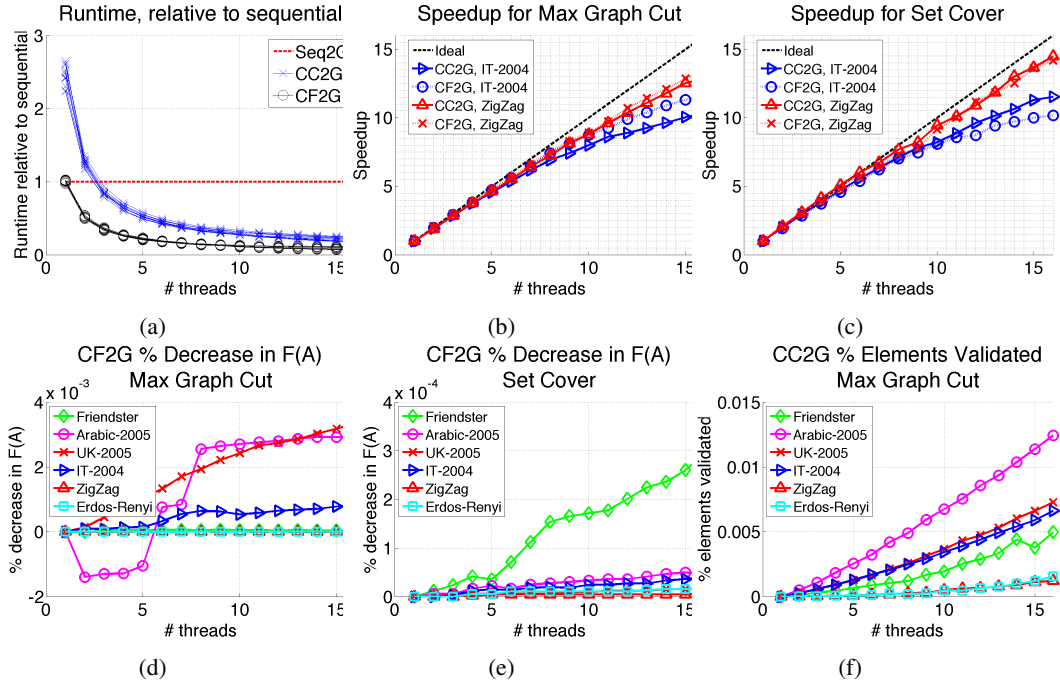Table 1: Synthetic and real graphs used in the evaluation of our parallel algorithms.

Figure 1: Experimental results. Fig 1a – runtime of the parallel algorithms as a ratio to that of the sequential algorithm. Each curve shows the runtime of a parallel algorithm on a particular graph for a particular function F. Fig 1b, 1c – speedup (ratio of runtime on one thread to that on p threads). Fig 1d, 1e – % difference between objective values of the sequential and hogwild algorithms, i.e. $\left[F(A_{hogwild})/F(A_{sequential}) - 1\right] \times 100\%$. Fig 1f – percentage of elements validated by the OCC algorithm on the max graph cut problem.

algorithm decreases with the number of threads, but differs from the sequential objective value by less than 0.01%. **Validations:** The OCC algorithm validates more elements as threads are added, but less than 0.015% are validated with 16 threads, which has negligible effect on the runtime / speedup.

7.1 Adversarial ordering

To highlight the philosophical differences between the two parallel algorithms, we conducted experiments on a ring Cayley graph on $\mathbb{Z}_{10^6}$ with generating set $\{\pm 1, \ldots, \pm 1000\}$. The algorithms are presented with an adversarial ordering, with permutation, so vertices close in the ordering are adjacent to one another, and tend to be processed concurrently. This causes CF2G to make more mistakes, and CC2G to face more uncertainty.

As Figure 2 shows, CC2G sacrifices speed to ensure serial equivalence, eventually validating $> 90\%$ of elements. On the other hand, CF2G focuses on speed, resulting in faster runtime, but delivering an objective value $F(A_{CF2G})$ that is only 20% of $F(A_{Seq2G})$. [3]
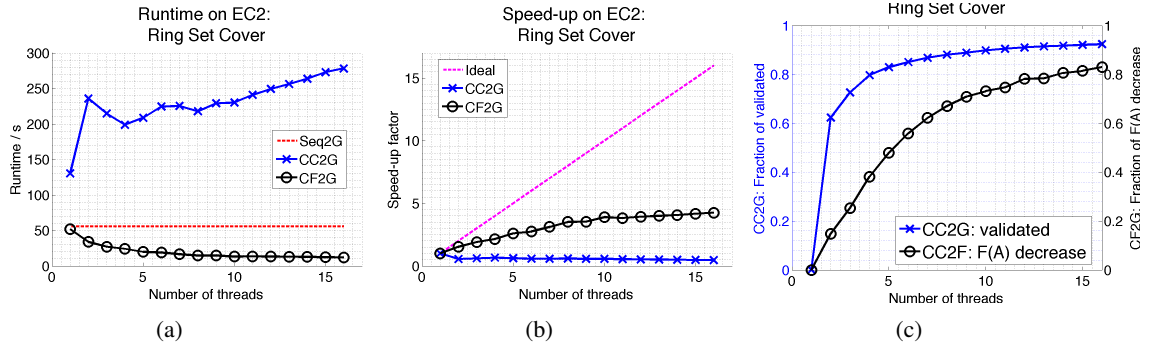

Figure 2: Experimental results for ring graph on set cover problem.

[XP: These experiments were conducted by using an atomic integer to select elements to process. We could instead use a partitioning scheme, which has 2 advantages. Firstly, there is less coordination – for CF2G, we essentially have no coordination. Secondly, when faced with an adversarial ordering, the partitioning scheme allows big jumps / re-orderings, which reduces the number of validations and *increases* the objective value of both parallel algorithms.]

In general, it is possible to avoid such adversarial ordering by randomly permuting the elements when pre-processing, or by using a different partitioning scheme, which we will discuss in a full paper.

8 Related Work

Similar approach, different problem: OCC DP-means [6]; Hogwild SGD [7]; Hogwild LDA [8] / parameter servers [9, 10]

Similar problem, different approach: distributed greedy submodular maximization for monotone functions [11]

9 Discussions

Conclusion: [XP: link back to intro, motivation]; we present two approaches to parallelizing unconstrained submodular maximization, which allows one to choose between speed and tight approximation guarantees.

Future work: constrained maximization, minimization; distributed setting, where communication costs and delays are higher, and function evaluations are challenging.

References

[1] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 649–658, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4874-6. doi: 10.1109/FOCS.2012.73. URL http://dx.doi.org/10.1109/FOCS.2012.73.

[2] Jure Leskovec. Stanford network analysis project, 2011. URL http://snap.stanford.edu/index.html.

---

[3]For the set cover problem, both algorithms have to perform atomic updates to the statistics α_l and β_l. Under the adversarial ordering, there are more concurrent atomic updates, and hence, CF2G does not achieve good speedup. It is possible to compute F directly, which would provide better speedup but slower runtimes.

[3] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

[4] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*. ACM Press, 2011.

[5] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

[6] Xinghao Pan, Joseph E Gonzalez, Stefanie Jegelka, Tamara Broderick, and Michael Jordan. Optimistic concurrency control for distributed unsupervised learning. In *Advances in Neural Information Processing Systems 26*. 2013.

[7] Benjamin Recht, Christopher Re, Stephen J. Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS) 24*, pages 693–701, Granada, 2011.

[8] Amr Ahmed, Mohamed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J. Smola. Scalable inference in latent variable models. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM)*, 2012.

[9] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola. Parameter server for distributed machine learning. In *Big Learn workshop, at NIPS*, Lake Tahoe, 2013.

[10] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A Gibson, Greg Ganger, and Eric Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in Neural Information Processing Systems 26*. 2013.

[11] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems 26*. 2013.

486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

A Proof of bound for hogwild

We follow the proof outline of [1].

Let OPT be an optimal solution to the problem. Define $O^i := (OPT \cup A^i) \cap B^i$. Note that O^i coincides with A^i and B^i on elements $1, \ldots, i$, and O^i coincides with OPT on elements $i+1, \ldots, n$. Hence,

$$O^i \backslash i + 1 \supseteq A^i$$
$$O^i \cup i + 1 \subseteq B^i.$$

Lemma A.1. *For every* $1 \le i \le n$, $\Delta_+(i) + \Delta_-(i) \ge 0$.

Proof. This is just Lemma II.1 of [1]. $\qquad\square$

Lemma A.2. *For every* $1 \le i \le n$,

$$E[F(O^{i-1}) - F(O^i)] \le \frac{1}{2} E[f(A^i) - f(A^{i-1}) + f(B^i) - f(B^{i-1}) + \rho_i].$$

Proof. We follow the proof outline of [1]. First, note that it suffices to prove the inequality conditioned on knowing A^{i-1} and j, then applying the law of total expectation. Under this conditioning, we also know B^{i-1}, O^{i-1}, $\Delta_+(i)$, $\Delta_+^{\max}(i)$, $\Delta_-(i)$, $\Delta_-^{\max}(i)$, and ρ_i.

We consider the following 9 cases.

Case 1: $0 < \Delta_+(i) \le \Delta_+^{\max}(i)$, $0 \le \Delta_-^{\max}(i)$. Since both $\Delta_+^{\max}(i) > 0$ and $\Delta_-^{\max}(i) > 0$, the probability of including i is just $\Delta_+^{\max}(i)/(\Delta_+^{\max}(i) + \Delta_-^{\max}(i))$, and the probability of excluding i is $\Delta_-^{\max}(i)/(\Delta_+^{\max}(i) + \Delta_-^{\max}(i))$.

$$E[F(A^i) - F(A^{i-1})|A^{i-1}, j] = \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(A^{i-1} \cup i) - F(A^{i-1}))$$

$$= \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\Delta_+(i)$$

$$\ge \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(\Delta_+^{\max}(i) - \rho_i)$$

$$E[F(B^i) - F(B^{i-1})|A^{i-1}, j] = \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(B^{i-1} \backslash i) - F(B^{i-1}))$$

$$= \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\Delta_-(i)$$

$$\ge \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(\Delta_-^{\max}(i) - \rho_i)$$

10

$$E[F(O^{i-1}) - F(O^i)|A^{i-1}, j]$$

$$= \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(O^{i-1}) - F(O^{i-1} \cup i))$$

$$+ \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(O^{i-1}) - F(O^{i-1} \backslash i))$$

$$= \begin{cases} \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(O^{i-1}) - F(O^{i-1} \cup i)) & \text{if } i \notin OPT \\ \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(O^{i-1}) - F(O^{i-1} \backslash i)) & \text{if } i \in OPT \end{cases}$$

$$\leq \begin{cases} \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(B^{i-1} \backslash i) - F(B^{i-1})) & \text{if } i \notin OPT \\ \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}(F(A^{i-1} \cup i) - F(A^{i-1})) & \text{if } i \in OPT \end{cases}$$

$$= \begin{cases} \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\Delta_-(i) & \text{if } i \notin OPT \\ \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\Delta_+(i) & \text{if } i \in OPT \end{cases}$$

$$\leq \begin{cases} \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\Delta_-^{\max}(i) & \text{if } i \notin OPT \\ \frac{\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\Delta_+^{\max}(i) & \text{if } i \in OPT \end{cases}$$

$$= \frac{\Delta_+^{\max}(i)\Delta_-^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}$$

where the first inequality is due to submodularity: $O^{i-1} \backslash i \supseteq A^{i-1}$ and $O^{i-1} \cup i \subseteq B^{i-1}$.

Putting the above inequalities together:

$$E[F(O^{i-1}) - F(O^i)|A^{i-1}, j] - \frac{1}{2}E[f(A^i) - f(A^{i-1}) + f(B^i) - f(B^{i-1}) + \rho_i|A^{i-1}, j]$$

$$\leq \frac{1/2}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\left[2\Delta_+^{\max}(i)\Delta_-^{\max}(i) - \Delta_-^{\max}(i)(\Delta_-^{\max}(i) - \rho_i)\right.$$

$$\left. - \Delta_+^{\max}(i)(\Delta_+^{\max}(i) - \rho_i)\right] - \frac{1}{2}\rho_i$$

$$= \frac{1/2}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}\left[-(\Delta_+^{\max}(i) - \Delta_-^{\max}(i))^2 + \rho_i(\Delta_+^{\max}(i) + \Delta_-^{\max}(i))\right] - \frac{1}{2}\rho_i$$

$$\leq \frac{\frac{1}{2}\rho_i(\Delta_+^{\max}(i) + \Delta_-^{\max}(i))}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)} - \frac{1}{2}\rho_i$$

$$= 0.$$

Case 2: $0 < \Delta_+(i) \leq \Delta_+^{\max}(i)$, $\Delta_-^{\max}(i) < 0$. In this case, the algorithm always choses to include i, so $A^i = A^{i-1} \cup i$, $B^i = B^{i-1}$ and $O^i = O^{i-1} \cup i$:

$$E[F(A^i) - F(A^{i-1})|A^{i-1}, j] = F(A^{i-1} \cup i) - F(A^{i-1}) = \Delta_+(i) > 0$$

$$E[F(B^i) - F(B^{i-1})|A^{i-1}, j] = F(B^{i-1}) - F(B^{i-1}) = 0$$

$$E[F(O^{i-1}) - F(O^i)|A^{i-1}, j] = F(O^{i-1}) - F(O^{i-1} \cup i)$$

$$\leq \begin{cases} 0 & \text{if } i \in OPT \\ F(B^{i-1} \backslash i) - F(B^{i-1}) & \text{if } i \notin OPT \end{cases}$$

$$= \begin{cases} 0 & \text{if } i \in OPT \\ \Delta_-(i) & \text{if } i \notin OPT \end{cases}$$

$$\leq 0$$

$$< \frac{1}{2}E[f(A^i) - f(A^{i-1}) + f(B^i) - f(B^{i-1}) + \rho_i|A^{i-1}, j]$$

where the first inequality is due to submodularity: $O^{i-1} \cup i \subseteq B^{i-1}$.

11

Case 3: $\Delta_+(i) \leq 0 < \Delta_+^{\max}(i), 0 < \Delta_-(i) < \Delta_-^{\max}(i)$. Analogous to Case 1.

Case 4: $\Delta_+(i) \leq 0 < \Delta_+^{\max}(i), \Delta_-(i) \leq 0$. This is not possible, by Lemma A.1.

Case 5: $\Delta_+(i) \leq \Delta_+^{\max}(i) \leq 0, 0 < \Delta_-(i) \leq \Delta_-^{\max}(i)$. Analogous to Case 2.

Case 6: $\Delta_+(i) \leq \Delta_+^{\max}(i) \leq 0, \Delta_-(i) \leq 0$. This is not possible, by Lemma A.1.

$\square$

([XP: Note] If we weaken the assumption of $\Delta_+(i) \leq \Delta_+^{\max}(i)$ to $\Delta_+(i) \leq \Delta_+^{\max}(i) + \epsilon_i$, then in Case 6 above, we can instead bound

$$E[F(O^{i-1}) - F(O^i)|A^{i-1}, j] \leq \frac{\Delta_+^{\max}(i)\Delta_-^{\max}(i) + \epsilon \max(\Delta_+^{\max}(i), \Delta_-^{\max})}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}$$

$$\leq \frac{\Delta_+^{\max}(i)\Delta_-^{\max}(i) + \epsilon(\Delta_+^{\max}(i) + \Delta_-^{\max})}{\Delta_+^{\max}(i) + \Delta_-^{\max}(i)}.$$

The bound of Lemma A.2 becomes

$$E[F(O^{i-1}) - F(O^i)] \leq \frac{1}{2}E[f(A^i) - f(A^{i-1}) + f(B^i) - f(B^{i-1}) + \rho_i + 2\epsilon_i],$$

and the bound of Theorem 6.1 becomes $E[F(A)] \geq \frac{1}{2}F^* - \frac{1}{4}\sum_i E[\rho_i + 2\epsilon_i]$.)

We will now prove Theorem 6.1.

Proof of Theorem 6.1. Summing up the statement of Lemma A.2 for all i gives us a telescoping sum, which reduces to:

$$E[F(O^0) - F(O^n)] \leq \frac{1}{2}E[F(A^n) - F(A^0) + F(B^n) - F(B^0)] + \frac{1}{2}\sum_{i=1}^{n} E[\rho_i]$$

$$\leq \frac{1}{2}E[F(A^n) + F(B^n)] + \frac{1}{2}\sum_{i=1}^{n} E[\rho_i].$$

Note that $O^0 = OPT$ and $O^n = A^n = B^n$, so $E[F(A^n)] \geq \frac{1}{2}F(OPT) - \frac{1}{4}\sum_i E[\rho_i]$. $\square$

A.1   Example: max graph cut

Let $C^{ji} = \{j+1, \ldots, i-1\}, D^i = \{i+1, \ldots, n\}$. Denote $\tilde{A}^j = V \backslash A^j \backslash C^{ji} \backslash D^i = \{1, \ldots, j\} \backslash A^j$ be the elements up to j that are not included in A. Let $w_i(S) = \sum_{j \in S, (i,j) \in E} w(i,j)$. For the max graph cut function, it is easy to see that

$$\Delta_+ \geq -w_i(A^j) - w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)$$
$$\Delta_+^{\max} = -w_i(A^j) + w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)$$
$$\Delta_- \geq +w_i(A^j) - w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j)$$
$$\Delta_-^{\max} = +w_i(A^j) + w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j)$$

Thus, we can set $\rho_i = 2w_i(C^{ji})$.

Suppose we have bounded delay τ, so $|C^{ji}| \leq \tau$. Then $w_i(C^{ji})$ has a hypergeometric distribution with mean $\frac{\deg(i)}{N}\tau$, and $E[\rho_i] = 2\tau\frac{\deg(i)}{N}$. The approximation of the hogwild algorithm is then $E[F(A^n)] \geq \frac{1}{2}F(OPT) - \tau\frac{\#\text{edges}}{2N}$. In sparse graphs, the hogwild algorithm is off by a small additional term, which albeit grows linearly in τ.

A.2 Example: set cover

Consider the simple set cover function, for $\lambda < 1$:

$$F(A) = \sum_{l=1}^{L} \min(1, |A \cap S_l|) - \lambda|A| = |\{l : A \cap S_l \neq \emptyset\}| - \lambda|A|.$$

We assume that there is some bounded delay τ.

Suppose also the S_l's form a partition, so each element e belongs to exactly one set. Let n_l denote $|S_l|$ the size of S_l. Given any ordering π, let e_l^t be the tth element of S_l in the ordering, i.e. $|\{e' : \pi(e') \leq \pi(e_l^t) \wedge e' \in S_l\}| = t$.

For any $e \in S_l$, we get

$$\Delta_+(e) = -\lambda + 1\{A^{\iota(e)-1} \cap S_l = \emptyset\}$$
$$\Delta_+^{\max}(e) = -\lambda + 1\{\hat{A}_e \cap S_l = \emptyset\}$$
$$\Delta_-(e) = +\lambda - 1\{B^{\iota(e)-1}\backslash e \cap S_l = \emptyset\}$$
$$\Delta_-^{\max}(e) = +\lambda - 1\{\hat{B}_e\backslash e \cap S_l = \emptyset\}$$

Let η be the position of the first element of S_l to be accepted, i.e. $\eta = \min\{t : e_l^t \in A \cap S_l\}$. (For convenience, we set $\eta = n_l$ if $A \cap S_l = \emptyset$.) We first show that η is independent of π: for $\eta < n_l$,

$$P(\eta|\pi) = \frac{\Delta_+^{\max}(e_l^\eta)}{\Delta_+^{\max}(e_l^\eta) + \Delta_-^{\max}(e_l^\eta)} \prod_{t=1}^{\eta-1} \frac{\Delta_-^{\max}(e_l^t)}{\Delta_+^{\max}(e_l^t) + \Delta_-^{\max}(e_l^t)}$$

$$= \frac{1-\lambda}{1-\lambda+\lambda} \prod_{t=1}^{\eta-1} \frac{\lambda}{1-\lambda+\lambda}$$

$$= (1-\lambda)\lambda^{\eta-1},$$

and $P(\eta = n_l|\pi) = \lambda^{\eta-1}$.

Note that, $\Delta_-^{\max}(e) - \Delta_-(e) = 1$ iff $e = e_l^{n_l}$ is the last element of S_l in the ordering, there are no elements accepted up to $\hat{B}_{e_l^{n_l}}\backslash e_l^{n_l}$, and there is some element e' in $\hat{B}_{e_l^{n_l}}\backslash e_l^{n_l}$ that is rejected and not in $B^{\iota(e_l^{n_l})-1}$. Denote by $m_l \leq \min(\tau, n_l - 1)$ the number of elements before $e_l^{n_l}$ that are inconsistent between $\hat{B}_{e_l^{n_l}}$ and $B^{\iota(e_l^{n_l})-1}$. Then $\mathbb{E}[\Delta_-^{\max}(e_l^{n_l}) - \Delta_-(e_l^{n_l})] = P(\Delta_-^{\max}(e_l^{n_l}) \neq \Delta_-(e_l^{n_l}))$ is

$$\lambda^{n_l-1-m_l}(1-\lambda^{m_l}) = \lambda^{n_l-1}(\lambda^{-m_l}-1) \leq \lambda^{n_l-1}(\lambda^{-\min(\tau,n_l-1)}-1) \leq 1-\lambda^\tau.$$

If $\lambda = 1$, $\Delta_+^{\max}(e) \leq 0$, so no elements before $e_l^{n_l}$ will be accepted, and $\Delta_-^{\max}(e_l^{n_l}) = \Delta_-(e_l^{n_l})$.

On the other hand, $\Delta_+^{\max}(e) - \Delta_+(e) = 1$ iff $(A^{\iota(e)-1}\backslash\hat{A}_e) \cap S_l \neq \emptyset$, that is, if an element has been accepted in A but not yet observed in $\hat{A}_e$. Since we assume a bounded delay, only the first τ elements after the first acceptance of an $e \in S_l$ may be affected.

$$\mathbb{E}\left[\sum_{e \in S_l} \Delta_+^{\max}(e) - \Delta_+(e)\right]$$

$$= \mathbb{E}[\#\{e : e \in S_l \wedge e_l^\eta \in A^{\iota(e)-1} \wedge e_l^\eta \notin \hat{A}_e\}]$$

$$= \mathbb{E}[\mathbb{E}[\#\{e : e \in S_l \wedge e_l^\eta \in A^{\iota(e)-1} \wedge e_l^\eta \notin \hat{A}_e\} \mid \eta = t, \pi(e_l^t) = k]]$$

$$= \sum_{t=1}^{n_l} \sum_{k=t}^{N-n+t} P(\eta = t, \pi(e_l^t) = k)\mathbb{E}[\#\{e : e \in S_l \wedge e_l^\eta \in A^{\iota(e)-1} \wedge e_l^\eta \notin \hat{A}_e\} \mid \eta = t, \pi(e_l^t) = k]$$

$$= \sum_{t=1}^{n_l} P(\eta = t) \sum_{k=t}^{N-n+t} P(\pi(e_l^t) = k)\mathbb{E}[\#\{e : e \in S_l \wedge e_l^\eta \in A^{\iota(e)-1} \wedge e_l^\eta \notin \hat{A}_e\} \mid \eta = t, \pi(e_l^t) = k].$$

13

Under the assumption that every ordering π is equally likely, and a bounded delay τ, conditioned on $\eta = t, \pi(e_l^t) = k$, the random variable $\#\{e : e \in S_l \wedge e_l^\eta \in A^{\iota(e)-1} \wedge e_l^\eta \notin \hat{A}_e\}$ has hypergeometric distribution with mean $\frac{n_l - t}{N - k}\tau$. Also, $P(\pi(e_l^t) = k) = \frac{n_l}{N}\binom{n-1}{t-1}\binom{N-n}{k-t}/\binom{N-1}{k-1}$, so the above expression becomes

$$\mathbb{E}\left[\sum_{e \in S_l} \Delta_+^{\max}(e) - \Delta_+(e)\right]$$

$$= \sum_{t=1}^{n_l} P(\eta = t) \sum_{k=t}^{N-n+t} \frac{n_l}{N} \frac{\binom{n-1}{t-1}\binom{N-n}{k-t}}{\binom{N-1}{k-1}} \frac{n-t}{N-k}\tau$$

$$= \frac{n_l}{N}\tau \sum_{t=1}^{n_l} P(\eta = t) \sum_{k=t}^{N-n+t} \frac{\binom{k-1}{t-1}\binom{N-k}{n-t}}{\binom{N-1}{n-1}} \frac{n-t}{N-k} \qquad \text{(symmetry of hypergeometric)}$$

$$= \frac{n_l}{N}\tau \sum_{t=1}^{n_l} \frac{P(\eta = t)}{\binom{N-1}{n-1}} \sum_{k=t}^{N-n+t} \binom{k-1}{t-1}\binom{N-k-1}{n-t-1}$$

$$= \frac{n_l}{N}\tau \sum_{t=1}^{n_l} \frac{P(\eta = t)}{\binom{N-1}{n-1}} \binom{N-1}{n-1} \qquad \text{(Lemma C.1, } a = N-2, b = n_l - 2, j = 1\text{)}$$

$$= \frac{n_l}{N}\tau \sum_{t=1}^{n_l} P(\eta = t)$$

$$= \frac{n_l}{N}\tau.$$

Now if we define $\rho_e = \Delta_+^{\max}(e) - \Delta_+(e) + \Delta_-^{\max}(e) - \Delta_-(e)$, we get

$$\mathbb{E}\left[\sum_e \rho_e\right] = \mathbb{E}\left[\sum_e \Delta_+^{\max}(e) - \Delta_+(e) + \Delta_-^{\max}(e) - \Delta_-(e)\right]$$

$$= \sum_l \mathbb{E}\left[\sum_{e \in S_l} \Delta_+^{\max}(e) - \Delta_+(e)\right] + \mathbb{E}\left[\sum_{e \in S_l} \Delta_-^{\max}(e) - \Delta_-(e)\right]$$

$$\leq \tau \frac{\sum_l n_l}{N} + L(1 - \lambda^\tau)$$

$$= \tau + L(1 - \lambda^\tau).$$

Note that $\mathbb{E}\left[\sum_e \rho_e\right]$ does not depend on N and is linear in τ. Also, if $\tau = 0$ in the sequential case, we get $\mathbb{E}\left[\sum_e \rho_e\right] \leq 0$.

B   Upper bound on expected number of elements sent for validation

Let N be the number of elements, i.e. the cardinality of the ground set. Let P be the number of processors.

We assume that the total ordering assigns elements to processors in a round robin fashion. Thus, we assume $C^{ji} = \{i - p + 1, \ldots, i - 1\}$ has $p - 1$ elements.

We call element i *dependent* on i' if $\exists A, F(A \cup i) - F(A) \neq F(A \cup i' \cup i) - F(A \cup i')$ or $\exists B, F(B \backslash i) - F(B) \neq F(B \cup i' \backslash i) - F(B \cup i')$, i.e. the result of the transaction on i' will affect the computation of Δ's for i. For example, for the graph cut problem, every vertex is dependent on its neighbors; for the separable sums problem, i is dependent on $\{i' : \exists S_l, i \in S_l, i' \in S_l\}$.

Let n_i be the number of elements that i is dependent on.

Now, we note that if C^{ji} does not contain any elements on which i is dependent, then $\Delta_+^{\max}(i) = \Delta_+(i) = \Delta_+^{\min}(i)$ and $\Delta_-^{\max}(i) = \Delta_-(i) = \Delta_-^{\min}(i)$, so i will not be validated (in either deterministic or probabilistic versions). Conversely, if i is validated, there must be some element $i' \in C^{ji}$ such that i is dependent on i'.

$$
\begin{aligned}
E&(\text{number of validated elements}) \\
&= \sum_i P(i \text{ validated}) \\
&\leq \sum_i P(\exists i' \in C^{ji}, i \text{ depends on } i') \\
&= \sum_i 1 - P(\forall i' \in C^{ji}, i \text{ does not depend on } i') \\
&= \sum_i 1 - \prod_{k=1}^{P-1} \frac{N - k - n_i}{N - k} \\
&= \sum_i 1 - \prod_{k=1}^{P-1} \left(1 - \frac{n_i}{N - k}\right) \\
&\leq \sum_i 1 - \left(1 - \sum_{k=1}^{P-1} \frac{n_i}{N - k}\right) && \text{(Weierstrass inequality)} \\
&= \left(\sum_i n_i\right)\left(\sum_{k=1}^{P-1} \frac{1}{N - k}\right) \\
&\leq \frac{P - 1}{N - P + 1} \sum_i n_i.
\end{aligned}
$$

The key quantity in the above inequality is $\sum_i n_i$. Typically, we expect each element i to depend on a small fraction of the ground set. For example, in the graph cut problem, $\sum_i n_i = 2|E|$ is twice the number of edges. If the graph is sparse with $|E| \approx s|V| \log |V|$, where $0 \leq s \ll 1$ and $P \ll N$, then $\frac{P-1}{N-P+1} \sum_i n_i \approx 2s(P - 1) \log N$, which grows sublinearly with N.

Note that the bound established above is generic to all algorithms that follow the basic transactional model we proposed (round-robin optimistic concurrency control), and is not specific to F or even submodular maximization. Thus, while our bounds provide a fundamental limit, additional knowledge of F can lead to better analyses on the algorithm's concurrency.

B.1   Tighter general bound?

Define $\rho_i = \max_{S \subseteq V} \{[F(S \cup i) - F(S)] - [F(S \cup C^{ji} \cup i) - F(S \cup C^{ji})]\} \leq F(i) - F(V) + F(V \backslash i)$

[XP: Is there theory along these lines?]

Then, we can bound

$$\Delta_+^{\min} \le \Delta_+^{\max} \le \Delta_+^{\min} + \rho_i \qquad\qquad \text{(choosing } S = A^j)$$
$$\Delta_-^{\min} \le \Delta_-^{\max} \le \Delta_-^{\min} + \rho_i \qquad\qquad \text{(choosing } S = A^j \cup D^i)$$

Thus,

$$
\begin{aligned}
& E(\text{number of validated elements}) \\
&= \sum_i P(i \text{ validated}) \\
&= \sum_i P\left(\frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} \le u_i \le \frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} \right) \\
&= \sum_i \frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} \\
&\le \sum_i \frac{\Delta_+^{\min} + \rho_i}{\Delta_+^{\min} + \rho_i + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \rho_i + \Delta_-^{\min}} \\
&= \sum_i \frac{\rho_i}{\Delta_+^{\min} + \rho_i + \Delta_-^{\min}}
\end{aligned}
$$

B.2 Upper bound for max graph cut

Denote $\tilde{A}^j = V \setminus A^j \setminus C^{ji} \setminus D^i = \{1, \ldots, j\} \setminus A^j$ be the elements up to j that are not included in A. Let $w_i(S) = \sum_{j \in S, (i,j) \in E} w(i,j)$. For the max graph cut function, it is easy to see that

$$
\begin{aligned}
\Delta_+^{\min} &= \max(0, -w_i(A^j) - w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)) \\
\Delta_+^{\max} &= \max(0, -w_i(A^j) + w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)) \\
\Delta_-^{\min} &= \max(0, +w_i(A^j) - w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j)) \\
\Delta_-^{\max} &= \max(0, +w_i(A^j) + w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j))
\end{aligned}
$$

Consider the following cases.

- $\Delta_+^{\max} = 0$. Then $\Delta_+^{\min} = 0$ and also

$$w_i(A^j) > w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j) \quad \Longrightarrow \quad w_i(A^j) + w_i(D^i) > w_i(C^{ji}) + w_i(\tilde{A}^j)$$

so $\Delta_-^{\min} > 0$ and $\Delta_-^{\max} > 0$. Thus $\frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} = 0 - 0 = 0$.

- $\Delta_-^{\max} = 0$. Then $\Delta_-^{\min} = 0$ and also

$$w_i(\tilde{A}^j) > w_i(C^{ji}) + w_i(D^i) + w_i(A^j) \quad \Longrightarrow \quad w_i(\tilde{A}^j) + w_i(D^i) > w_i(C^{ji}) + w_i(A^j)$$

so $\Delta_+^{\min} > 0$ and $\Delta_+^{\max} > 0$. Thus $\frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} = 1 - 1 = 0$.

- $\Delta_+^{\max} > 0$ and $\Delta_-^{\max} > 0$. Then,

$$\frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}}$$

$$= \frac{-w_i(A^j) + w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)}{-w_i(A^j) + w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j) + \max(0, +w_i(A^j) - w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j))}$$

$$- \frac{\max(0, -w_i(A^j) - w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j))}{\max(0, -w_i(A^j) - w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)) + w_i(A^j) + w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j)}$$

$$= \min\left(1, \frac{-w_i(A^j) + w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)}{2w_i(D^i)}\right)$$

$$- \max\left(0, \frac{-w_i(A^j) - w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)}{2w_i(D^i)}\right)$$

$$= \min\left(1, \frac{w_i(C^{ji})}{w_i(D^i)}\right)$$

Thus,

$$E(\# \text{ of validated elements}) = \sum_i \frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} \le \sum_i \min\left(1, \frac{w_i(C^{ji})}{w_i(D^i)}\right)$$

[XP: Not sure how to sum this over i.]

$$\sum_\pi \sum_i \min(1, w_i(C)/w(D^i)) \le E\left(\sum_i w_i(C)\right) = c * \sum_i deg(i)/n$$

B.3   Upper bound for set cover

We make the same assumptions as before in the hogwild analysis, i.e. the sets S_l form a partition of V, there is a bounded delay τ.

Observe that for any $e \in S_l$, $\Delta_-^{\min}(e) \ne \Delta_-^{\max}(e)$ if $\hat{B}_e \backslash e \cap S_l \ne \emptyset$ and $\tilde{B}_e \backslash e \cap S_l = \emptyset$.

This is only possible if $e_l^{n_l} \notin \tilde{B}_e$ and $\tilde{B}_e \supset \hat{A}_e \cap S_l = \emptyset$, that is $\pi(e) \ge \pi(e_l^{n_l}) - \tau$ and $\forall e' \in S_l, (\pi(e') < \pi(e_l^{n_l}) - \tau) \implies (e' \notin A)$. The latter condition is achieved with probability $\lambda^{n_l - m_l}$, where $m_l = \#\{e' : \pi(e') \ge \pi(e_l^{n_l}) - \tau\}$. Thus,

$$\mathbb{E}\left[\#\{e : \Delta_-^{\min}(e) \ne \Delta_-^{\max}(e)\}\right] = \mathbb{E}[m_l \, 1(\forall e' \in S_l, (\pi(e') < \pi(e_l^{n_l}) - \tau) \implies (e' \notin A))]$$

$$= \mathbb{E}[\mathbb{E}[m_l \, 1(\forall e' \in S_l, (\pi(e') < \pi(e_l^{n_l}) - \tau) \implies (e' \notin A))|u_{1:N}]]$$

$$= \mathbb{E}[m_l \, \mathbb{E}[1(\forall e' \in S_l, (\pi(e') < \pi(e_l^{n_l}) - \tau) \implies (e' \notin A))|u_{1:N}]]$$

$$= \mathbb{E}[m_l \lambda^{n_l - m_l}]$$

$$\le \lambda^{(n_l - \tau)_+} \mathbb{E}[m_l]$$

$$= \lambda^{(n_l - \tau)_+} \mathbb{E}[\mathbb{E}[m_l | \pi(e_l^{n_l}) = k]]$$

$$= \lambda^{(n_l - \tau)_+} \sum_{k=n_l}^N P(\pi(e_l^{n_l}) = k)\mathbb{E}[m_l | \pi(e_l^{n_l}) = k].$$

17

Conditioned on $\pi(e_l^{n_l}) = k$, m_l is a hypergeometric random variable with mean $\frac{n_l-1}{k-1}\tau$. Also $P(\pi(e_l^{n_l}) = k) = \frac{n_l}{N}\binom{n_l-1}{0}\binom{N-n_l}{N-k}/\binom{N-1}{N-k}$. The above expression is therefore

$$\mathbb{E}\left[\#\{e : \Delta_-^{\min}(e) \neq \Delta_-^{\max}(e)\}\right]$$

$$= \lambda^{(n_l-\tau)_+}\sum_{k=n_l}^{N}\frac{n_l}{N}\frac{\binom{n_l-1}{0}\binom{N-n_l}{N-k}}{\binom{N-1}{N-k}}\frac{n_l-1}{k-1}\tau$$

$$= \lambda^{(n_l-\tau)_+}\frac{n_l}{N}\tau\sum_{k=n_l}^{N}\frac{\binom{N-k}{0}\binom{k-1}{n_l-1}}{\binom{N-1}{n_l-1}}\frac{n_l-1}{k-1} \qquad \text{(symmetry of hypergeometric)}$$

$$= \lambda^{(n_l-\tau)_+}\frac{n_l}{N}\frac{\tau}{\binom{N-1}{n_l-1}}\sum_{k=n_l}^{N}\binom{N-k}{0}\binom{k-2}{n_l-2}$$

$$= \lambda^{(n_l-\tau)_+}\frac{n_l}{N}\frac{\tau}{\binom{N-1}{n_l-1}}\binom{N-1}{n_l-1} \qquad \text{(Lemma C.1, } a = N-2,\, b = n_l-2,\, j = 2,\, t = n_l)$$

$$= \lambda^{(n_l-\tau)_+}\frac{n_l}{N}\tau.$$

Now we consider any element $e \in S_l$ with $\pi(e) < \pi(e_l^{n_l}) - \tau$ that is validated. (Note that $e_l^{n_l} \in \hat{B}_e$ and $\tilde{B}_e$, so $\Delta_-^{\min}(e) = \Delta_-^{\max}(e) = \lambda$.) It must be the case that $\hat{A}_e \cap S_l = \emptyset$, for otherwise $\Delta_+^{\min}(e) = \Delta_+^{\max}(e) = -\lambda$ and we do not need to validate. This implies that $\Delta_+^{\max}(e) = 1 - \lambda \geq u_i$. At validation, if $A^{\iota(e)-1} \cap S_l = \emptyset$, we accept e into A. Otherwise, $A^{\iota(e)-1} \cap S_l \neq \emptyset$, which implies that some other element $e' \in S_l$ has been accepted. Thus, we conclude that every element $e \in S_l$ that is validated must be within τ of the first accepted element e_l^η $in S_l$. The expected number of such elements is exactly as we computed in the hogwild analysis: $\frac{n_l}{N}\tau$.

Hence, the expected number of elements that we need to validate is upper bounded as

$$\mathbb{E}[\#\text{validated}] \leq \sum_l (1 + \lambda^{(n_l-\tau)_+})\frac{n_l}{N}\tau$$

$$\leq \sum_l 2\frac{n_l}{N}\tau$$

$$= 2\tau.$$

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

C   Lemma

Lemma C.1. $\sum_{k=t}^{a-b+t} \binom{k-j}{t-j}\binom{a-k+j}{b-t+j} = \binom{a+1}{b+1}$.

Proof.

$$\sum_{k=t}^{a-b+t} \binom{k-j}{t-j}\binom{a-k+j}{b-t+j}$$

$$= \sum_{k'=0}^{a-b} \binom{k'+t-j}{t-j}\binom{a-k'-t+j}{b-t+j}$$

$$= \sum_{k'=0}^{a-b} \binom{k'+t-j}{k'}\binom{a-k'-t+j}{a-b-k'} \qquad \text{(symmetry of binomial coeff.)}$$

$$= (-1)^{a-b}\sum_{k'=0}^{a-b} \binom{-t+j-1}{k'}\binom{-b+t-j-1}{a-b-k'} \qquad \text{(upper negation)}$$

$$= (-1)^{a-b}\binom{-b-2}{a-b} \qquad \text{(Chu-Vandermonde's identity)}$$

$$= \binom{a+1}{a-b} \qquad \text{(upper negation)}$$

$$= \binom{a+1}{b+1} \qquad \text{(symmetry of binomial coeff.)}$$

$\square$

19

D   Full experiment results

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
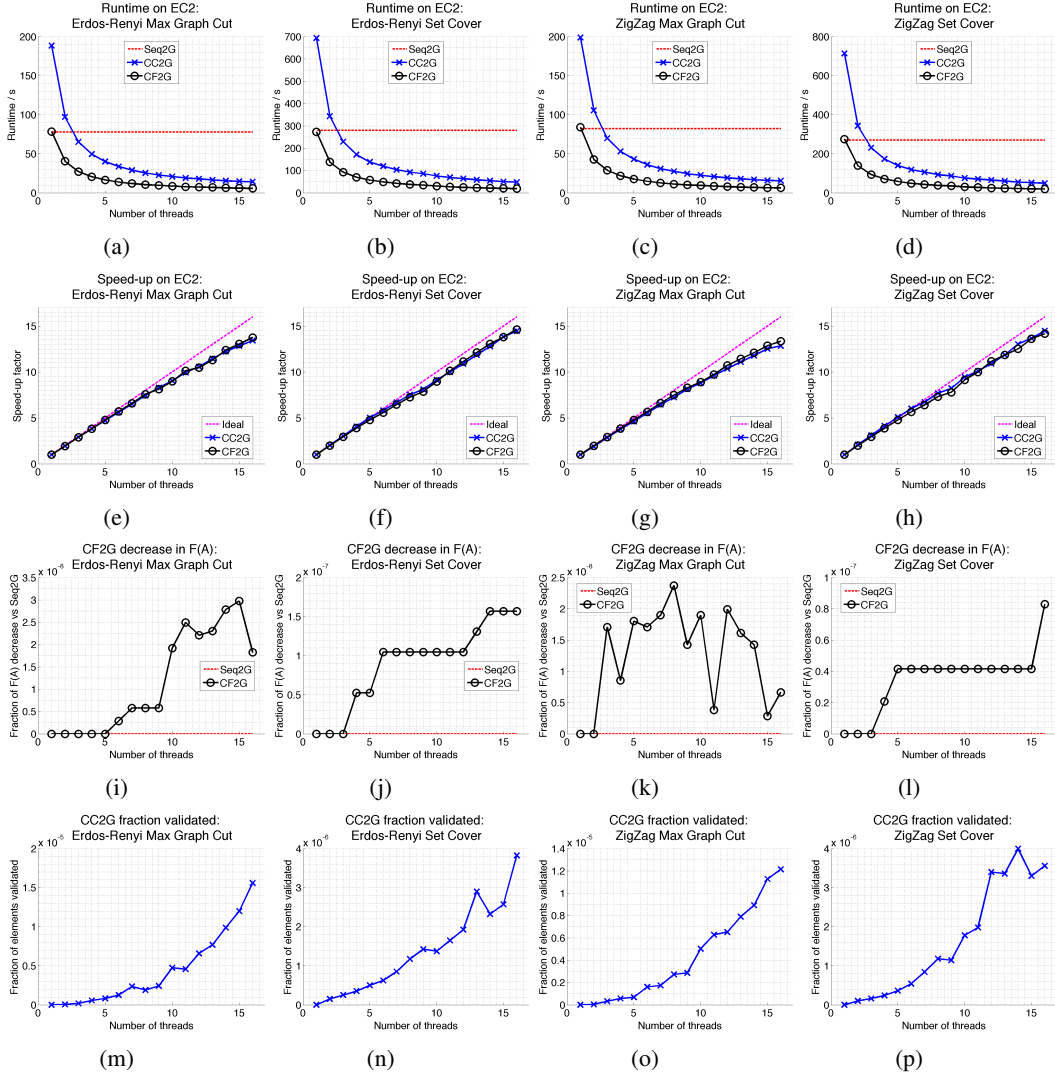1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Figure 3: Experimental results on Erdos-Renyi and ZigZag synthetic graphs.

Figure 4: Set cover on 4 real graphs.

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
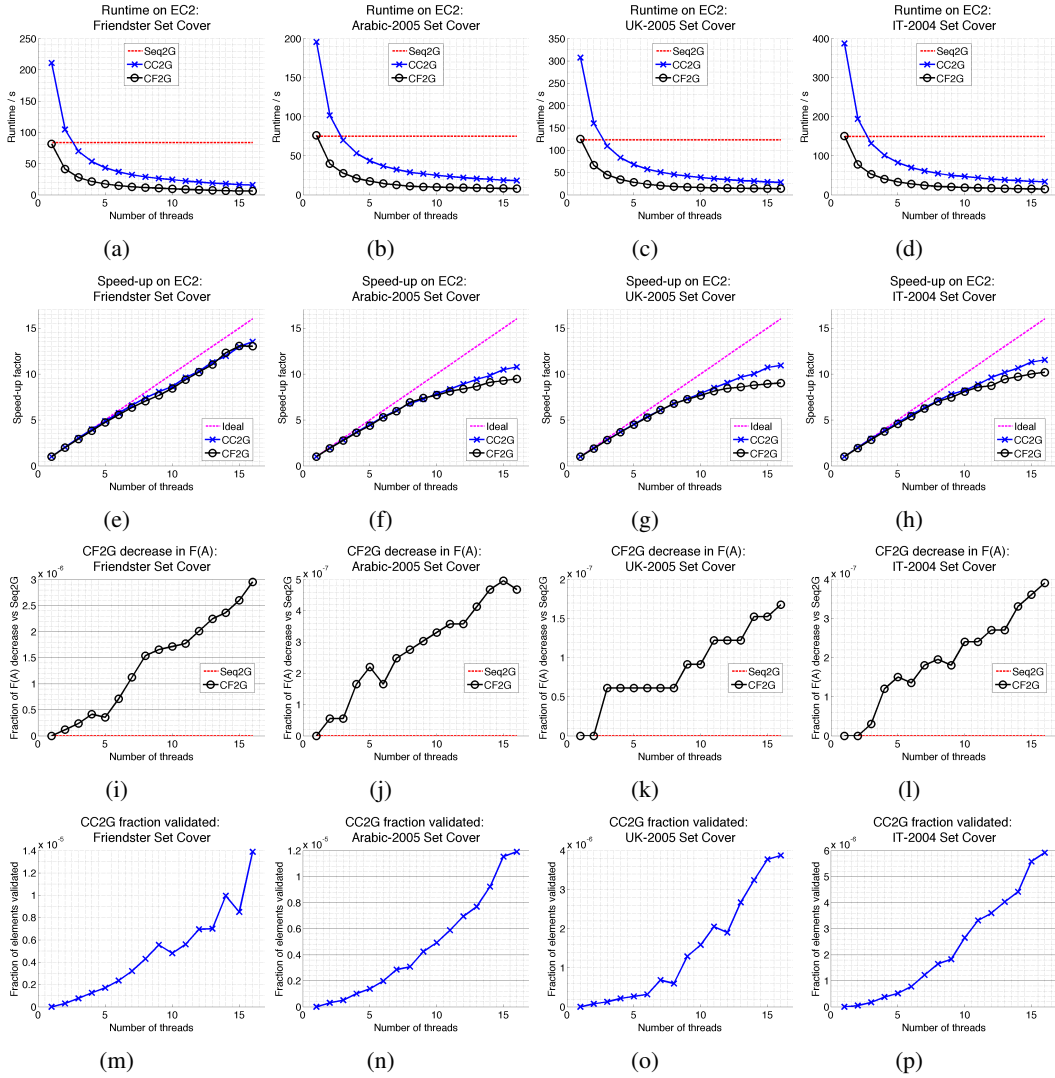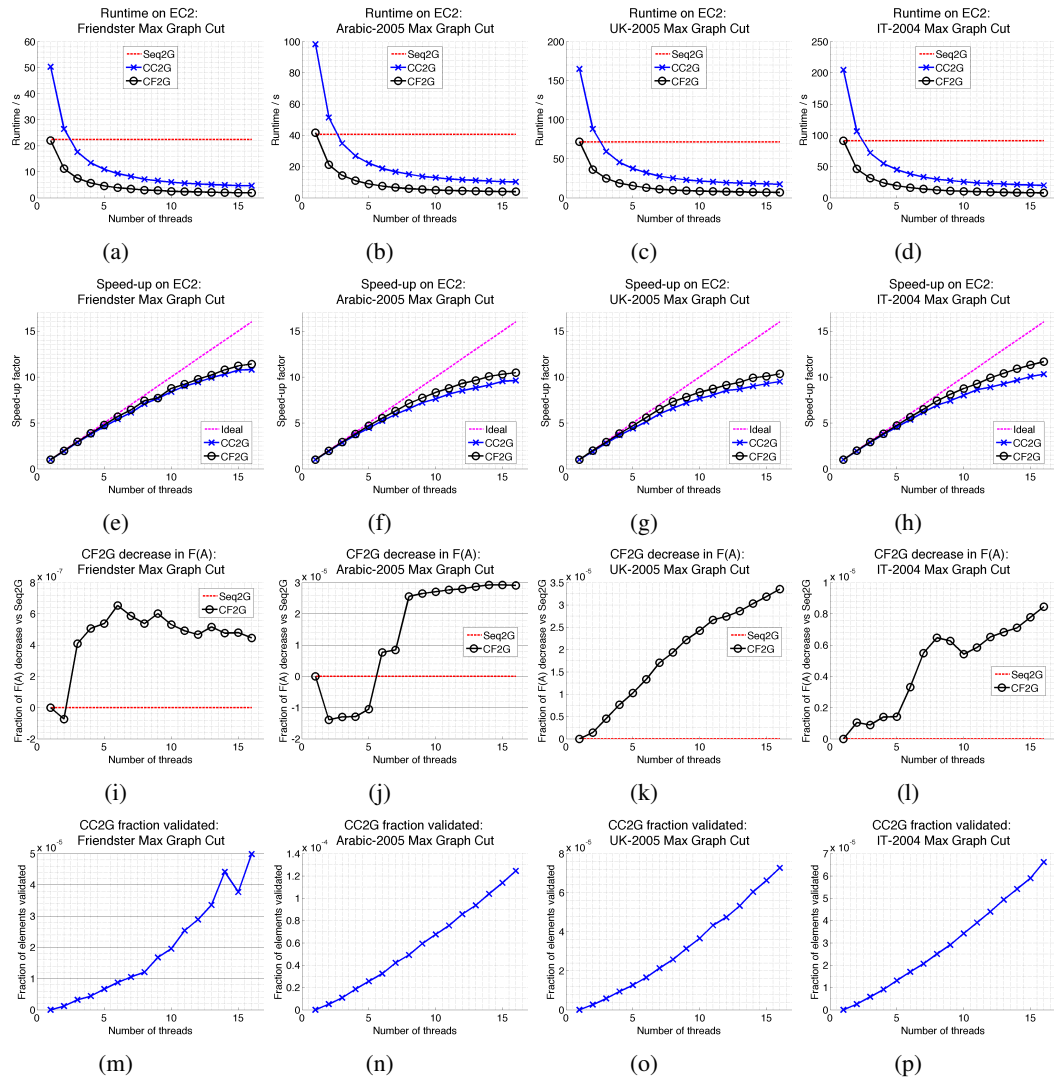
Figure 5: Max graph cut on 4 real graphs.
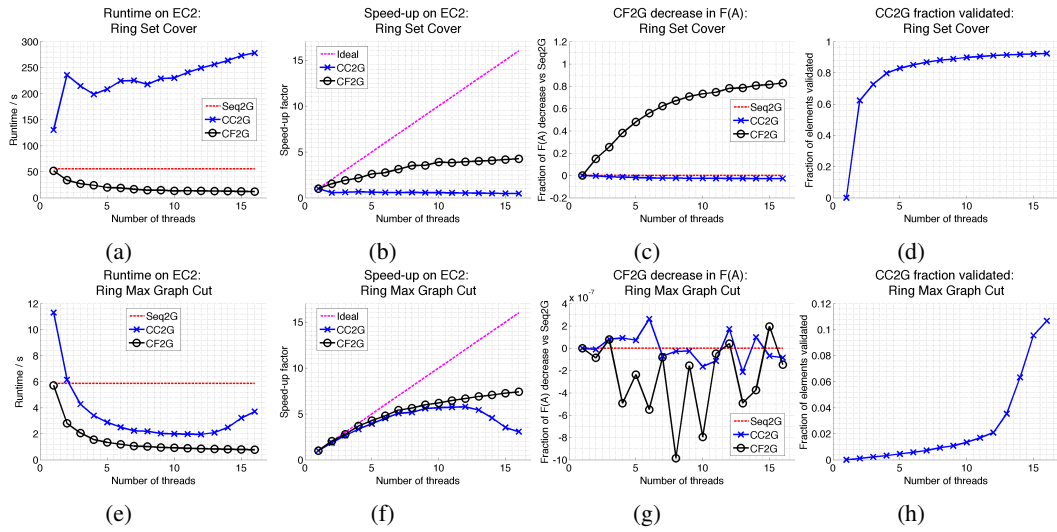
Figure 6: Experimental results for ring graph on set cover problem.

E   Full experiment results, partitioning with work stealing

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
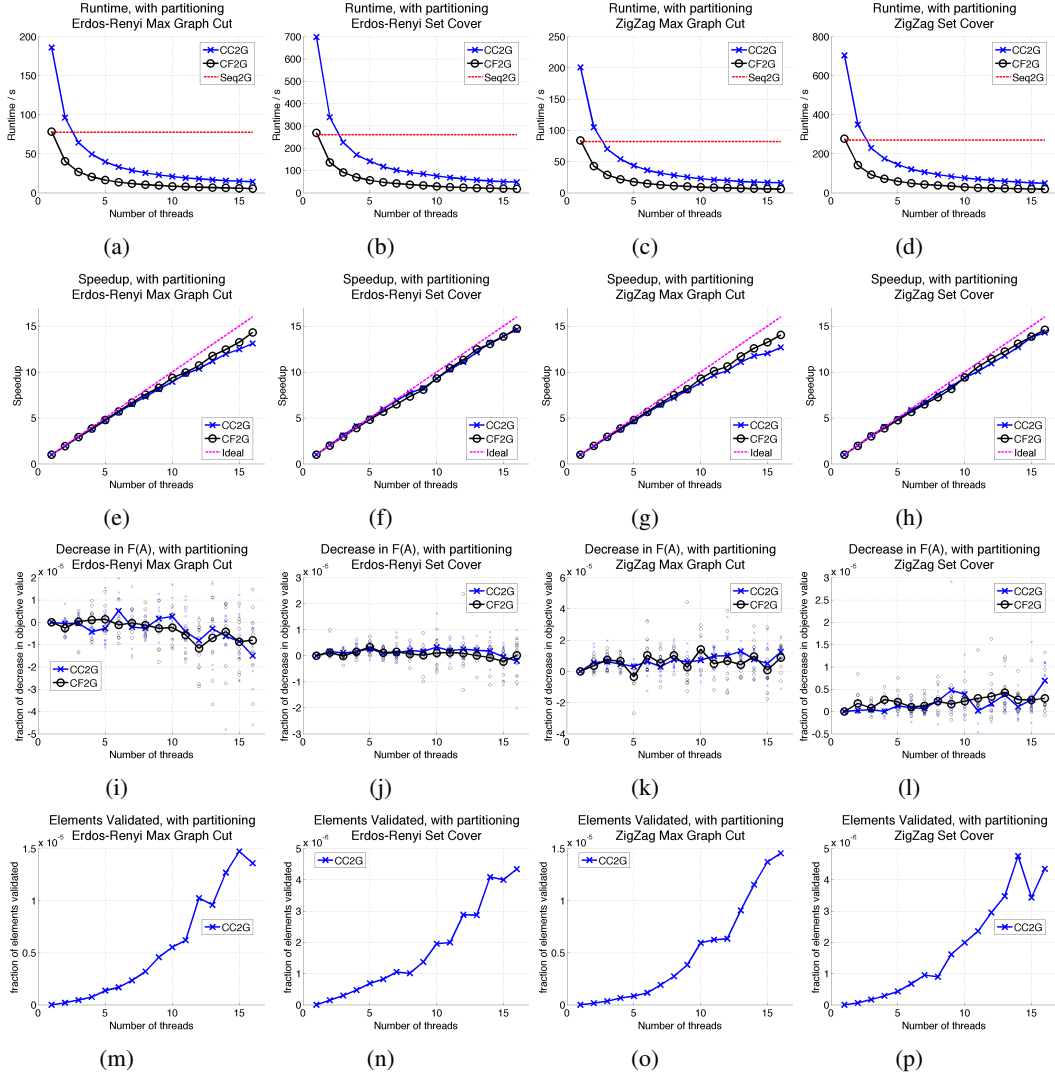
Figure 7: Experimental results (with partitioning) on Erdos-Renyi and ZigZag synthetic graphs.

25

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
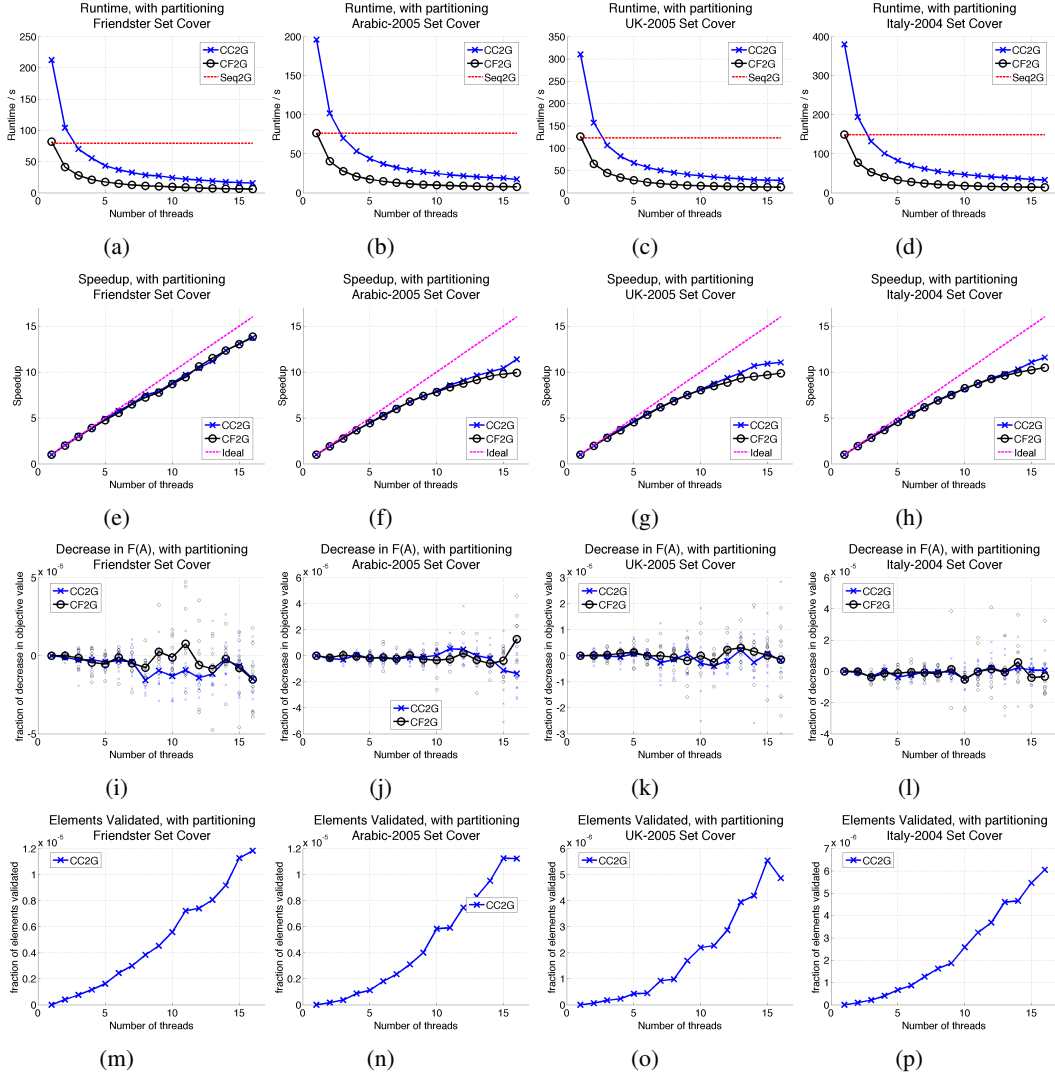
Figure 8: Set cover (with partitioning) on 4 real graphs.

1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
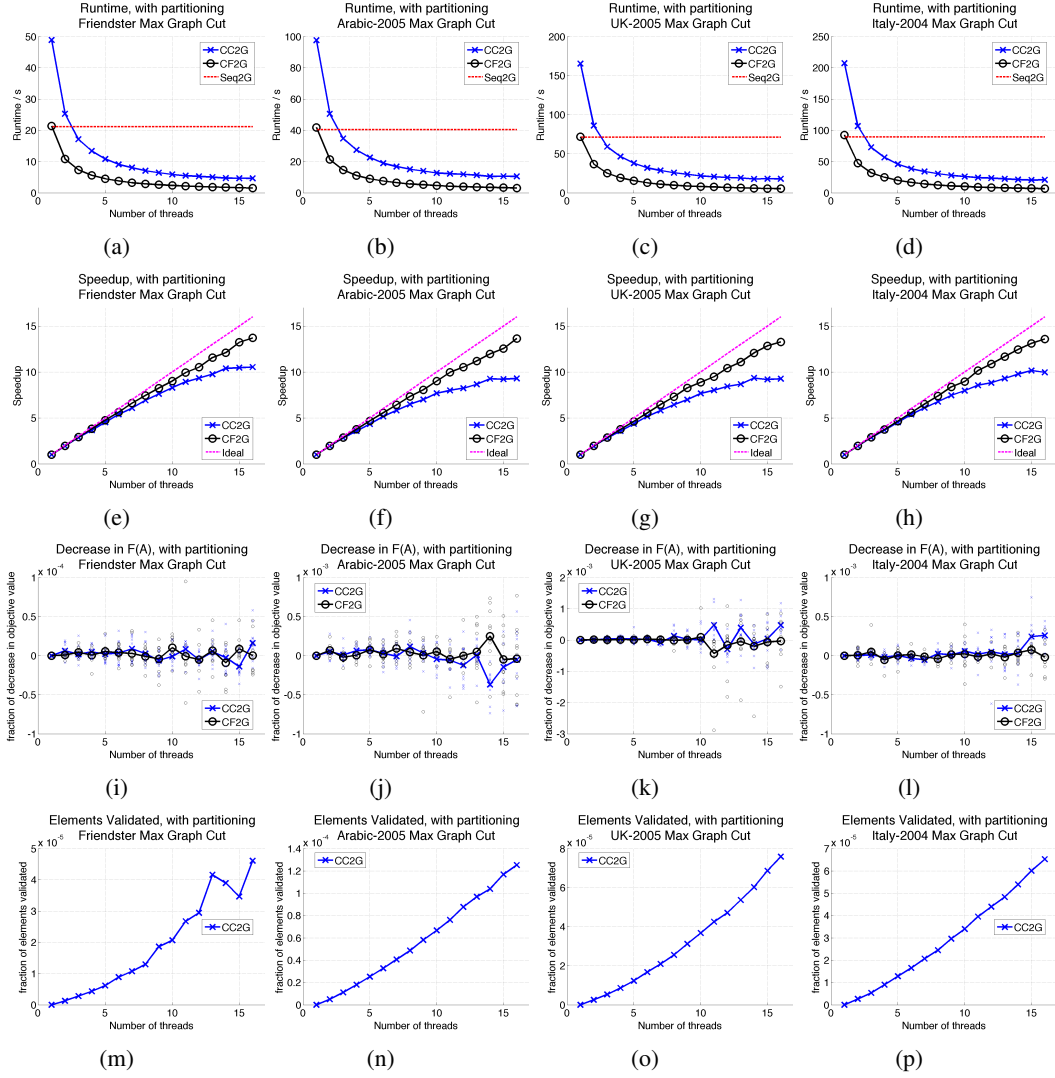
Figure 9: Max graph cut (with partitioning) on 4 real graphs.

1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
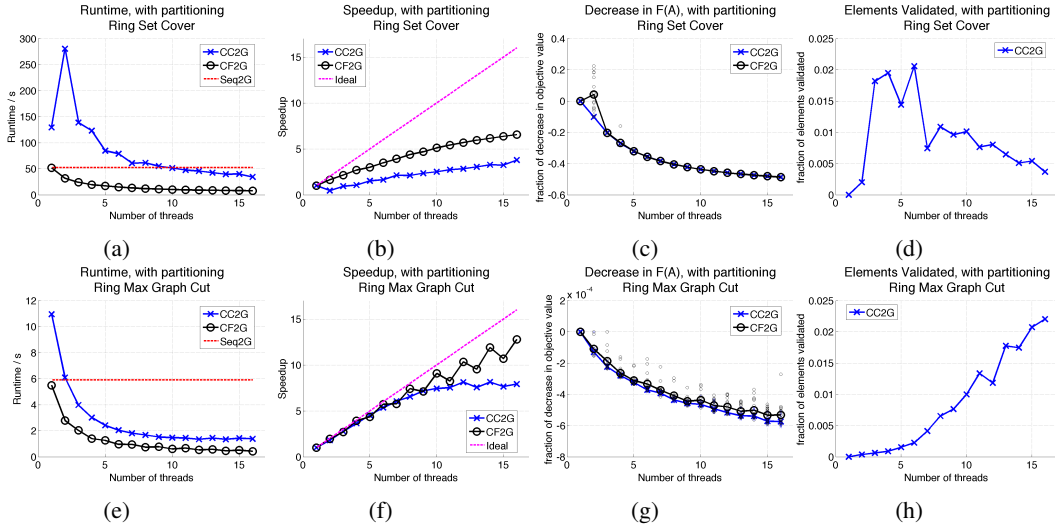1504
1505
1506
1507
1508
1509
1510
1511

Figure 10: Experimental results (with partitioning) for ring graph on set cover problem.