A set function $F : 2^V \rightarrow \mathbb{R}$ is submodular if it satisfies diminishing marginal returns. Applications: sensor placement, image co-segmentation, influence maximization in social networks, document summarization, etc.

Monotone submodular function: $\forall A \subseteq B, F(A) \leq F(B)$

Monotone submodular function: $\forall A \subseteq B, F(A) \leq F(B)$

Double greedy [1] has tight $1/2$ approximation for maximizing non-monotone submodular functions, but is serial $\rightarrow$ Does not scale to big data!

**Objective:** Parallelize (non-monotone) submodular maximization with strong approximation guarantees and high concurrency.

Idea: Operations on data point $\Rightarrow$ Database transaction. Apply concepts of concurrency control from database research.

**Concurrency Control Double Greedy (CC-2g)**

Decentralized maintenance of bounds $A^{min} \subseteq A \subseteq A^{max}$, $B^{min} \subseteq B \subseteq B^{max}$, enables threads to make decisions locally.

**Serial Double Greedy**

Compute marginal gains $\Delta_{\text{rand}}(v|A) = F(A \cup v) - F(A)$, $
\Delta_{\text{rand}}(v|B) = F(B \cup v) - F(B)$.

Sample random decision $P(A \cup v) = \Delta_{\text{rand}}(v|A) / (\Delta_{\text{rand}}(v|A) + \Delta_{\text{rand}}(v|B))$

$P(B \cup v) = \Delta_{\text{rand}}(v|B) / (\Delta_{\text{rand}}(v|A) + \Delta_{\text{rand}}(v|B))$

Update $A, B$

**Concurrency Control Double Greedy (CC-2g)**

Transaction (v)

Update bounds on $A, B$

Compute marginal gains $\Delta_{\text{rand}}(v|A) = F(A \cup v) - F(A)$, $\Delta_{\text{rand}}(v|B) = F(B \cup v) - F(B)$.

Sample random decision $P(A \cup v) = \Delta_{\text{rand}}(v|A) / (\Delta_{\text{rand}}(v|A) + \Delta_{\text{rand}}(v|B))$

$P(B \cup v) = \Delta_{\text{rand}}(v|B) / (\Delta_{\text{rand}}(v|A) + \Delta_{\text{rand}}(v|B))$

Update bounds on $A, B$

Transaction (w)

Estimate marginal gains

Sample random decision

Wait for concurrent txn to finish

Compute true marginal gains

Update bounds on $A, B$

Serializability: CC-2g's output is equivalent to serial double greedy $\rightarrow$ guarantees a tight $1/2$ approximation.

Scalability: We can upper bound the expected number of failed txns using $\tau$ the max inter-processor delay:

Max graph cut: $2 \tau |E|/|V|$, where $|E|$ and $|V|$ are number of edges and vertices respectively.

Set cover: $2 \tau$, assuming disjoint sets.

**Coordination Free Double Greedy (CF-2g)**

Compute using outdated $A^{old}, B^{old}$

$A, B$ change slowly $\Rightarrow$ $A^{old} \approx A$ and $B^{old} \approx B$ with small errors

**Experiments & Results**

Amazon EC2 cc2.8xlarge machines, 16 threads, 10 runs. Synthetic & real graphs, up to 40M vertices & 2B edges.

**Concurrency Control**

<table>
<thead>
<tr>
<th>Concurrency Control</th>
<th>Coordination Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtimes, rel. to serial</td>
<td>Faster on 2-3 threads</td>
</tr>
<tr>
<td>Speedup (16 threads)</td>
<td>$\approx 10x$</td>
</tr>
<tr>
<td>Approximation</td>
<td>= serial</td>
</tr>
<tr>
<td>Failed Transactions</td>
<td>$&lt; 0.01%$</td>
</tr>
<tr>
<td>Adversarial</td>
<td>Slow but correct</td>
</tr>
</tbody>
</table>

Approximation: We can upper bound CF-2g's expected error (relative to serial) using $\tau$

Max graph cut: $\tau |E|/|V|$

Set cover: $2 \tau$, assuming disjoint sets.

Optimal $1/2$ approx. guarantee. Serial $\Rightarrow$ Does not scale!