Algorithm 1: Generalized transactions

for $p \in \{1, \ldots, P\}$ **do in parallel**
  while $\exists$ *element to process* **do**
    e = next element to process
    $\mathfrak{S}_e$ = getSnapshot
    (op, $\mathfrak{A}_e$) = propose(e, $\mathfrak{S}_e$)
    validate(e, op, $\mathfrak{A}_e$)

Algorithm 2: validate

if *op = fail* $\vee \neg \mathfrak{A}_e(\mathfrak{S})$ **then**
  `// Reject, and re-compute`
  op = repropose(e, $\mathfrak{S}$)
`// Apply op to current state`
$\mathfrak{S} \leftarrow$ op($\mathfrak{S}$)

Figure 1: Algorithm for generalized transactions. Each transaction makes a copy $\mathfrak{S}_e$ of the current state $\mathfrak{S}$, and proposes an operation op and a set of associated assumptions $\mathfrak{A}_e$. The valdiation procedure verifies that the assumptions hold on the current state, and ultimately applies op to the $\mathfrak{S}$.

Algorithm 3: Serial submodular maximization

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

Algorithm 4: CF2G bidirectional greedy

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

Algorithm 5: CC2G bidirectional greedy

for $e \in V$ **do** $\hat{A}(e) = \tilde{A}(e) = 0$, $\hat{B}(e) = \tilde{B}(e) = 1$
for $i = 1, \ldots, |V|$ **do** processed(i) $= false$
$\iota = 0$
for $p \in \{1, \ldots, P\}$ **do in parallel**
  while $\exists$ *element to process* **do**
    e = next element to process
    $(\hat{A}_e, \tilde{A}_e, \hat{B}_e, \tilde{B}_e, i)$ = getSnapshot(e)
    (result, u_e) = propose(e, $\hat{A}_e$, $\tilde{A}_e$, $\hat{B}_e$, $\tilde{B}_e$)
    validate(e, i, u_e, result)

Algorithm 6: CC2G getSnapshot(e)

$\tilde{A}(e) \leftarrow 1$; $\tilde{B}(e) \leftarrow 0$
$i = \iota$; $\iota \leftarrow \iota + 1$
$\hat{A}_e = \hat{A}$; $\hat{B}_e = \hat{B}$
$\tilde{A}_e = \tilde{A}$; $\tilde{B}_e = \tilde{B}$
return $(\hat{A}_e, \tilde{A}_e, \hat{B}_e, \tilde{B}_e, i)$

Algorithm 7: CC2G propose

$\Delta_+^{\min}(e) = F(\tilde{A}_e) - F(\tilde{A}_e \backslash e)$
$\Delta_+^{\max}(e) = F(\hat{A}_e \cup e) - F(\hat{A}_e)$
$\Delta_-^{\min}(e) = F(\tilde{B}_e) - F(\tilde{B}_e \cup e)$
$\Delta_-^{\max}(e) = F(\hat{B}_e \backslash e) - F(\hat{B}_e)$
Draw $u_e \sim Unif(0, 1)$
if $u_e < \frac{[\Delta_+^{\min}(e)]_+}{[\Delta_+^{\min}(e)]_+ + [\Delta_-^{\max}(e)]_+}$ **then**
  result $\leftarrow 1$
else if $u_e > \frac{[\Delta_+^{\max}(e)]_+}{[\Delta_+^{\max}(e)]_+ + [\Delta_-^{\min}(e)]_+}$ **then**
  result $\leftarrow -1$
else result $\leftarrow fail$
return (result, u_e)

Algorithm 8: CC2G: validate(e, i, u_e, result)

wait until $\forall j < i$, processed(j) $= true$
if *result* $= fail$ **then**
  $\Delta_+^{exact}(e) = F(\hat{A} \cup e) - F(\hat{A})$
  $\Delta_-^{exact}(e) = F(\hat{B} \backslash e) - F(\hat{B})$
  if $u_e < \frac{[\Delta_+^{exact}(e)]_+}{[\Delta_+^{exact}(e)]_+ + [\Delta_-^{exact}(e)]_+}$ **then** result $\leftarrow 1$
  else result $\leftarrow -1$
if *result* $= 1$ **then** $\hat{A}(e) \leftarrow 1$; $\tilde{B}(e) \leftarrow 1$
else $\tilde{A}(e) \leftarrow 0$; $\hat{B}(e) \leftarrow 0$
processed(i) $= true$

4 CF2G Bidirectional Greedy Algorithm

[XP: Provide more intuition for what the CF2G algorithm is doing.]