Algorithm 1 Algorithm to determine optimal parallelization strategy for an operator DAG

1: **procedure** FINDOPTSTRATEGY($G = (V, E)$, $nprocs$)
2: **Input(s):** $G = (V, E)$: Operator DAG.
$(\forall v \in V)[v.dim]$: Dimension of the iteration domain of the operation of v.
$nprocs$: Number of processors.
3: **Output(s):** $\mathcal{S}$: Optimal parallelization starategy for G
4:
5: **for all** $v \in V$ **do**
6: $v.neighbors \leftarrow \{u \mid (u, v) \in E \vee (v, u) \in E\}$
7: $v.processed \leftarrow \bot$
8: $v.configs \leftarrow \{(c_1, c_2, \ldots, c_{v.dim}) \mid (\forall i \in [1, v.dim])[c_i \in \mathbb{Z}^+] \wedge \prod_{i=1}^{v.dim} c_i \leq nprocs\}$
9: **end for**
10:
11: ▷ Populate $v.tbl$ with all possible combinations of neighboring configurations. Each entry in $v.tbl$ is a sub-strategy for the nodes $\{v\} \cup v.neighbors$
12: **for all** $v \in V$ **do**
13: $v.tbl \leftarrow \prod_{u \in \{v\} \cup v.neighbors} \{(u, cfg) \mid cfg \in u.configs\}$
14: **end for**
15:
16: $U \leftarrow \{v \in V \mid v.processed = \bot\}$ ▷ Set of unprocessed vertices
17: **while** $U \neq \emptyset$ **do**
18: ▷ Choose a vertex with least no. of unprocessed neighbors from U
19: **for all** $v \in U$ **do**
20: $v.unprocessed\_neighbors \leftarrow \{u \mid u, v \in U \wedge u \in v.neighbors\}$
21: **end for**
22: $v_{min} \leftarrow \arg\min_{v \in U} |v.unprocessed\_neighbors|$
23:
24: ▷ Partition $v_{min}.tbl$ such that in each set of the partition, all the neighboring configurations are the same. Refer Alg. 2 for the function HASSAMENEIGHCONFIG
25: $\mathcal{P} \leftarrow \{S_1, S_2, \ldots \mid \cup_{S_i} = v_{min}.tbl \wedge i \neq j \implies S_i \cap S_j = \emptyset \wedge$
$\text{HASSAMENEIGHCONFIGS}(v_{min}, S_i, S_i) = \top) \wedge$
$i \neq j \implies \text{HASSAMENEIGHCONFIGS}(v_{min}, S_i, S_j) = \bot)\}$
26:
27: ▷ Iterate over each set $S \in \mathcal{P}$ and update $v_{min}.tbl$ so that it only contains the sub-strategy in S that has the least cost.
28: $v_{min}.tbl \leftarrow \emptyset$
29: **for all** $S \in \mathcal{P}$ **do**
30: $best \leftarrow \arg\min_{s \in S} \text{COST}(G, s)$
31: $v_{min}.tbl \leftarrow v_{min}.tbl \cup \{best\}$
32: **end for**
33:
34: ▷ Restrict the tables of the neighbors based on the best sub-strategies found for v_{min} in Line 30
35: **for all** $v \in v_{min}.neighbors$ **do**
36: $n\_common = |(v_{min}.neighbors \cap v.neighbors) \cup \{v_{min}, v\}|$
37: $v.tbl \leftarrow \{entry_1 \in v.tbl \mid (\exists entry_2 \in v_{min}.tbl)[|entry_1 \cap entry_2| = n\_common]\}$
38: **end for**
39:
40: $v_{min}.processed \leftarrow \top$
41: $U \leftarrow U \setminus \{v_{min}\}$
42: **end while**
43:
44: ▷ Once all the vertices are processed, $v.tbl$ of all the vertices should contain just a single entry.
45: $\mathcal{S} \leftarrow \emptyset$
46: **for all** $v \in V$ **do**
47: $\mathcal{S} \leftarrow \mathcal{S} \cup v.tbl$
48: **end for**
49: **return** $\mathcal{S}$
50: **end procedure**

Algorithm 2 Returns $\top$ if the two sets of sub-strategies of a vertex have same configurations for the neighboring vertices.

```
procedure HasSameNeighConfigs(v, S_1, S_2)
    for all s_1 ∈ S_1 do
        for all s_2 ∈ S_2 do
            N_1 ← {(n, c) ∈ S_1 | n ≠ v}
            N_2 ← {(n, c) ∈ S_2 | n ≠ v}

            if N_1 ≠ N_2 then
                return ⊥
            end if
        end for
    end for

    return ⊤
end procedure
```