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: **for all** $v \in V$ **do**
5: $v.neighbors \leftarrow \{u \mid (u, v) \in E \vee (v, u) \in E\}$
6: $v.unprocessed\_neighbors \leftarrow v.neighbors$
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: ▷ Populate $v.tbl$ with all possible combinations of neighboring configurations. Each entry in $v.tbl$ is a sub-strategy for the nodes $\{v\} \cup v.neighbors$
11: **for all** $v \in V$ **do**
12: $v.tbl \leftarrow \prod_{u \in \{v\} \cup v.neighbors} \{(u, cfg) \mid cfg \in u.configs\}$
13: **end for**

14: $U \leftarrow V$ ▷ Set of unprocessed vertices
15: **while** $U \neq \emptyset$ **do**
16: ▷ Choose a vertex with least no. of unprocessed neighbors from U
17: $v_{min} \leftarrow \arg\min_{v \in U} |v.unprocessed\_neighbors|$

18: ▷ Partition $v_{min}.tbl$ such that in each set of the partition, all the common node configurations are the same. Refer Alg. 2 for the function ISCOMPATIBLE
19: $\mathcal{P} \leftarrow \{S_1, S_2, \ldots \mid \cup_{S_i} = v_{min}.tbl \wedge i \neq j \implies S_i \cap S_j = \emptyset \wedge$
$\textsc{IsCompatible}(v_{min}, S_i, S_i) = \top) \wedge$
$i \neq j \implies \textsc{IsCompatible}(v_{min}, S_i, S_j) = \bot)\}$
20: ▷ 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.
21: $v_{min}.tbl \leftarrow \emptyset$
22: **for all** $S \in \mathcal{P}$ **do**
23: $best \leftarrow \arg\min_{s \in S} \textsc{Cost}(G, s)$
24: $v_{min}.tbl \leftarrow v_{min}.tbl \cup \{best\}$
25: **end for**

26: ▷ Restrict the tables of the neighbors based on the best sub-strategies found for v_{min} in Line 23
27: **for all** $v \in v_{min}.neighbors$ **do**
28: $v.unprocessed\_neighbors \leftarrow v.unprocessed\_neighbors \setminus \{v_{min}\}$
29: $n\_common = |(v_{min}.neighbors \cap v.neighbors) \cup \{v_{min}, v\}|$
30: $v.tbl \leftarrow \{entry_1 \in v.tbl \mid (\exists entry_2 \in v_{min}.tbl)[|entry_1 \cap entry_2| = n\_common]\}$
31: **end for**

32: $v_{min}.processed \leftarrow \top$
33: $U \leftarrow U \setminus \{v_{min}\}$
34: **end while**

35: ▷ Collect all valid strategies for G from $v.tbl$, and pick the one that minimizes the cost.
36: $\mathcal{S}' \leftarrow \textsc{ValidStrategies}(G)$ ▷ Refer Alg. 3
37: $\mathcal{S} \leftarrow \arg\min_{S \in \mathcal{S}'} \textsc{Cost}(G, S)$
38: **return** $\mathcal{S}$
39: **end procedure**

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

procedure IsCompatible(v, S_1, S_2)
    for all $s_1 \in S_1$ **do**
        for all $s_2 \in S_2$ **do**
            $N_1 \leftarrow \{(n, c) \in S_1 \mid n \neq v\}$
            $N_2 \leftarrow \{(n, c) \in S_2 \mid n \neq v\}$

            if $N_1 \neq N_2$ **then**
                return $\perp$
            end if
        end for
    end for

    return $\top$
end procedure

Algorithm 3 Returns a set of valid strategies

procedure ValidStrategies($G = (V, E)$)
    $\mathcal{S} \leftarrow \emptyset$
    for all $v \in V$ **do**
        if $\mathcal{S} = \emptyset$ **then**
            $\mathcal{S} \leftarrow v.tbl$
            Continue
        end if

        $\mathcal{S}' \leftarrow \emptyset$
        for all $S \in \mathcal{S}$ **do**
            for all $T \in v.tbl$ **do**
                if IsCompatible(S, T) $= \top$ **then**
                    $S' \leftarrow S \cup T$
                    $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{S'\}$
                end if
            end for
        end for
        $\mathcal{S} \leftarrow \mathcal{S}'$
    end for

    return $\mathcal{S}$
end procedure