ALGORITHM FOR POD SCHEDULING IN KUBERNETES

INPUT:

- PodsQueue: Queue of unscheduled pods
- Nodes: Set of available nodes
- PodsTypeInfo: Information about pod types and their affinities/anti-affinities
- ResourceInfo: Resources available on nodes (CPU, memory, etc.)
- TrafficMatrix: Matrix representing inter-pod communication traffic
- LatencyMatrix: Matrix representing network latency between nodes
- CostConversionFactors: Factors to convert resource usage into cost-equivalent terms (e.g., CPU cost/unit, memory cost/unit)

STEPS

1. Initialization

- Compute PodsTraffic from TrafficMatrix, where PodsTraffic[i][j] represents the communication traffic between pod i and pod j.
- Extract resource requirements and affinity/anti-affinity rules for each pod from PodsQueue.

2. Preprocess Nodes and Pods

- Group nodes by resource capacity (e.g., high-memory, high-CPU, balanced).
- Categorize pods into workload types (e.g., compute-intensive, memory-intensive, balanced).

3. Convert Resource Usage into Cost

For each node N:

- Compute the **cost equivalence** of available resources:

$$\mathbf{CostAvailable}(N) = \mathbf{CPU}(N) \cdot \mathbf{CPUFactor} + \mathbf{Memory}(N) \cdot \mathbf{MemoryFactor}$$

- Similarly, for each pod P, compute the **resource request cost**:

$$\mathbf{CostRequested}(P) = \mathbf{CPU}(P) \cdot \mathbf{CPUFactor} + \mathbf{Memory}(P) \cdot \mathbf{MemoryFactor}$$

4. Calculate Node Scores for Each Pod

For each pod P in PodsQueue:

- **For** each node N in Nodes:
    1. **Compute Resource Cost Score**
       $$\mathbf{ResourceCostScore}(P,N)=\mathbf{CostAvailable}(N)-\mathbf{CostRequested}(P)$$
    2. **Compute Latency and Traffic Cost**
        - For already scheduled pods Q on N:
          $$\mathbf{TrafficCost}(P,Q)=\mathbf{PodsTraffic}[P][Q]\cdot\mathbf{LatencyMatrix}[N][\mathbf{Node}(Q)]$$
        - Sum the traffic cost:
          $$\mathbf{TrafficCostScore}(P,N)=\sum_{Q\in \mathbf{PodsOn}(N)}\mathbf{TrafficCost}(P,Q)$$
    3. **Compute Final Node Score**
       $$\mathbf{NodeScore}(P,N)=\alpha\cdot\mathbf{ResourceCostScore}-\beta\cdot\mathbf{TrafficCostScore}$$
       (where α,β are weight coefficients).

5. Assign Pods to Nodes

- **While** PodsQueue is not empty:
    - Pop the highest-priority pod P from PodsQueue.
    - Sort nodes N by their NodeScore for P in descending order.
    - Assign P to the highest-scoring node N that satisfies all constraints:
        - Sufficient resources.
        - No anti-affinity violations.
        - Affinity rules are satisfied.
    - Update resource availability and node state.
    - Recompute PodsTraffic and LatencyMatrix scores for remaining pods if necessary.

7. Return Result

- Output the final pod-to-node assignments.

MODIFICATIONS SUGGESTED

1. Consider pod qos class type to add separate score for pods.

2. Consider deployments with hpa and vpa .

3. In step 3, consider the actual pod level usage.