Plan Construction Implementation Details

Information from the Paper

The function ConstructCompositionPlan (Algorithm 1, line 6) discards all the unnecessary services in planSet to minimize the number of component services in the final solution plan, and then arranges these service sets in sequence.

Algorithm

Input: Composition request, plan sets generated by the backward search algorithm

Output: List of composition plans

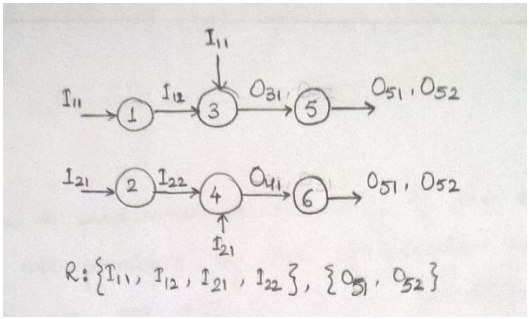
1. For each input plan set,
    a. Get the maximum layer index, say m, from the search nodes in the current plan set.
    b. Construct a plan with (m + 1) blank service layers.
    c. Add the search nodes in the plan service layers according to their respective layer index.
    d. Repeat until no more search nodes can be removed from the current plan:
        i. For each layer (from 0 to m - 1) in the current plan,
            A. For each search node in the current layer,
                I. Create a set of outputs produced by the predecessors of the current search node. These predecessor nodes must be present in the current plan.
                II. Take a union of the predecessor outputs with the set of composition request inputs. Let's call it *availableInputSet*.
                III. If the *availableInputSet* does not contain all the inputs required by the current search node, remove the current node from the current plan.
        ii. For each layer (from m - 1 to 0) in the current plan,
            A. For each search node in the current layer,
                I. Check if the current search node has any successors in the current plan.
                II. If not, check if the current search node produces at least 1 output requested in the composition request.
                III. If not, remove the current node from the current plan.
    e. If the number of search nodes in the current plan <= 1, discard the current plan. Otherwise,
        i. Create a set of all the outputs produced by all the remaining search nodes in the current plan. Let's call it *planOutputSet.*
        ii. If the *planOutputSet* does not contain all the composition request outputs, discard the current plan. Otherwise,

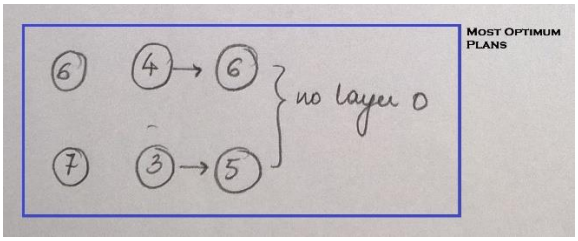
A. Check if the current plan already exists in the list of composition plans.
B. If not, add it to the list. Otherwise, discard it.

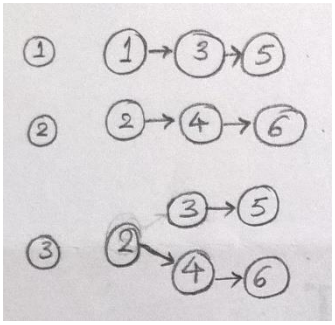
2. Return the list of composition plans.

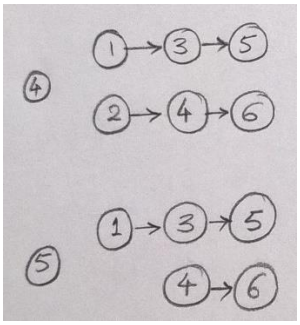
Limitations

- Even after the pruning and validation performed by the above algorithm, some composition plans may contain superfluous services.
  Solution: For pruning such plans further, another stage dedicated to optimizing the services in a composition plan can be added to the service composition process.









Notes

- Since a service can appear only once in a search graph and all the plans for a composition request are generated from the same search graph, the layer index of each search node, and hence the relative order of services in all the plans, remains the same. Therefore, to check if a plan has already been included in the list of composition plans, we just compare the set of services included in that plan with the ones of the existing plans. There is no need to specifically compare the order of these services in the plans as it always remains the same.

- Some composition plans may have some empty service layers (including layer 0) due to removal of invalid search nodes. These layers can be removed while constructing constraint-aware plans, if required.

- There could be cases where all the required inputs for a service w4 have been provided in the composition request, and yet if another service w2 in layer 0 produces one of w4's inputs as its output, w4 will be placed in layer 1 (not 0). In such cases, w2 would become a predecessor to w4. However, during plan set generation, depending on the composition request and the plan sets, the pruning process could generate plans that have w4 (layer 1) in them but not w2 (layer 0), although this might not always be the case in such scenarios. If such plans are generated, they will be considered as valid if they pass all the validations.
  The point to be noted here is that the inputs of w4 that were being supplied by w2 at the search graph stage could instead be supplied by the composition request itself at the composition plan stage.

Differences between Algorithm and Implementation

- The loop for invoking the plan construction algorithm for each plan set generated by the backward search algorithm is executed by the service composition algorithm. However, in the implementation, this loop has also been incorporated in the plan construction method itself.
  Also, the check for adding only unique plans to the list of plans has been moved from the service composition algorithm to the plan construction method in the implementation.

  Reason: This has been done for better modularity and lower coupling. Since the complete plan construction process is now contained within its own class, any future modifications in the process (if required) would not affect the service composition algorithm.