we compute the set of maximal cliques $\mathcal{C} = \{C_1, \ldots, C_k\}$ in $G(d, q)$ (i.e., cliques which will lose the clique property if a vertex is added to them). Since $G(d, q)$ is triangulated the maximal cliques can be computed in polynomial time. The set $\mathcal{C}$ will form the DCM1 subproblems. Once subtrees are computed on the DCM1 subproblems, they are merged using the Strict Consensus Merger (SCM).

2.5.1.2 Supertree phase: Strict Consensus Merger (SCM)

The Strict Consensus Merger (SCM), when used in a DCM analysis, combines a set of trees into a single tree. In DCM1, the subproblems are merged sequentially (pairwise at a time) in the order determined by the perfect elimination ordering of the vertices in the triangulated threshold graph $G(d, q)$ (see [37] for details).

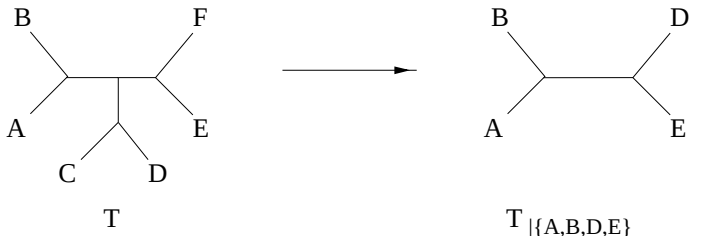

Figure 2.4: Tree T restricted to leaf set $\{A, B, D, E\}$.

We begin with some notation. We let $L(T)$ denote the set of leaves of T, $C(T)$ denote the set of bipartitions of T, and $T_{|X}$, where $X \subseteq L(T)$, denote the tree obtained by restricting the leaf set of T to X and suppressing nodes of degree 2 (see Figure 2.4). SCM takes two trees T_1, T_2 and returns a tree T_{12} on the leaf set $L(T_1) \cup L(T_2)$.

- **Input:** Trees T_1, T_2
- **Output:** A tree T_{12} whose leaf set is $L(T_1) \cup L(T_2)$
- **Algorithm:**
  - Set $X = L(T_1) \cap L(T_2)$ where $L(T)$ is the leaf set of tree T. We call X the *backbone* and it must satisfy $|X| \geq 3$, otherwise the merger is not defined.
  - Compute the strict consensus, T_X, of T_1 and T_2, each restricted to the leaf set X i.e., $T_X = StrictConsensus(T_{1|X}, T_{2|X})$.
  - Add the remaining taxa from T_1 and T_2 into T_X to form T, so as to preserve as much structure as possible. See Figures 2.7, 2.8, and 2.9 for examples of how this merger is accomplished. Note that it is possible that some piece of each tree T_1 and T_2 may attach onto the same edge of T_X; such an event is called a *collision*.

Trees T_1 and T_2 are said to be *compatible* if there exists a tree T such that $T_{|L(T_1)} = T_1$ and $T_{|L(T_2)} = T_2$. If there is no such tree T then T_1 and T_2 are said to be *incompatible*. Figures 2.5 and 2.6 illustrate the SCM algorithm on compatible trees and Figures 2.7, 2.8, and 2.9 demonstrate the SCM algorithm on incompatible trees, both with and without collisions.

Collision of Subtrees In Figures 2.6, 2.8 and 2.9, there is an edge in the backbone to which both trees contribute pieces: this is called a *collision*. In

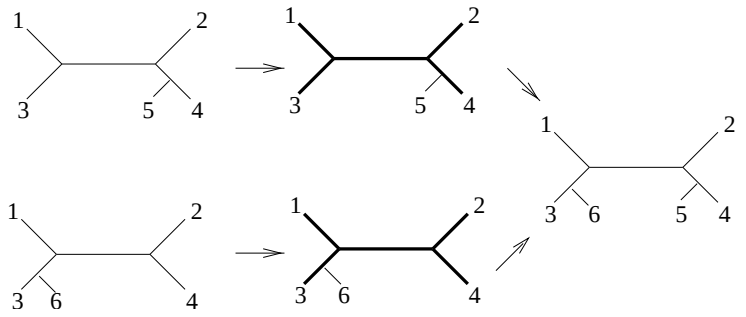

Figure 2.5: SCM on compatible trees without any collisions. We merge the two trees by first computing the strict consensus of the two trees restricted to the backbone (highlighted by thick edges). Note that the strict consensus does not contract edges in this case, since the trees are compatible.

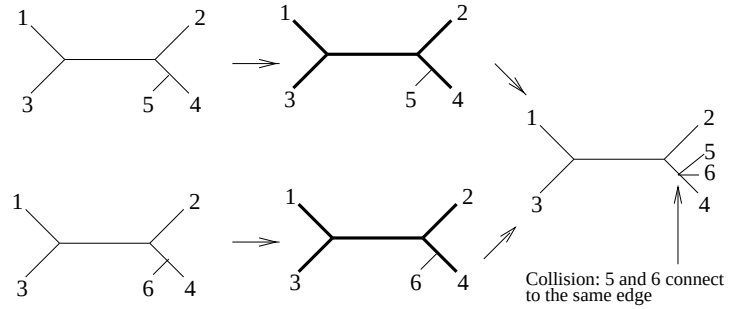

Figure 2.6: SCM on compatible trees with a collision. We merge the two trees by first computing the strict consensus of the two trees restricted to the backbone (backbone highlighted by thick edges).

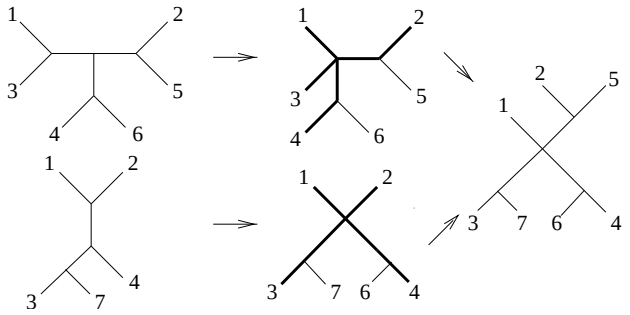

Figure 2.7: SCM on incompatible trees without any collisions. We merge the two trees by first computing the strict consensus of the two trees restricted to the backbone (highlighted by thick edges). Since the trees are incompatible, the strict consensus contracts edges in this example.

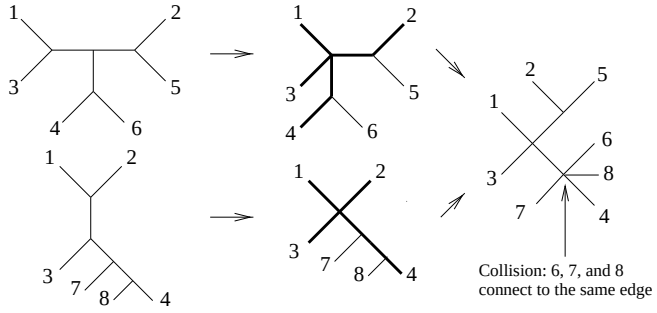

Figure 2.8: SCM on incompatible trees with a collision. We merge the two trees by computing the strict consensus of the two trees restricted to the backbone (highlighted by thick edges).

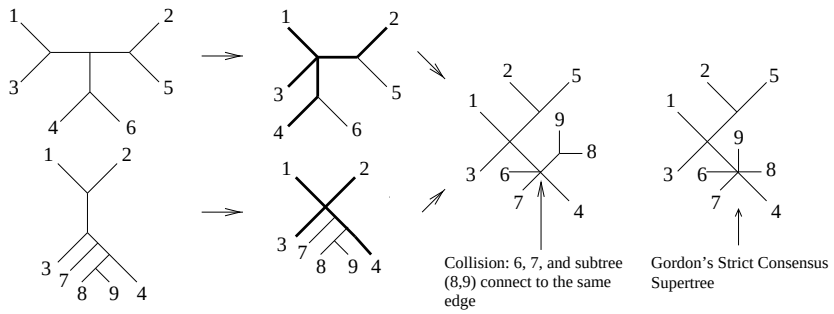

Figure 2.9: Handling collisions in the SCM and Gordon's Strict Consensus Supertree. We merge the two trees by computing the strict consensus of the two trees restricted to the backbone (highlighted by thick edges). We get a *collision* when adding the remaining leaves; however, note that the bipartition {{1,2,3,4,5,6,7},{8,9}} is present in the supertree. By contrast, in Gordon's Strict Consensus Supertree that edge would be contracted.

each of these figures, we see more than one leaf or subtree trying to connect to the edge incident to leaf 4. The Strict Consensus Merger handles collisions in the following way. If an edge e of the backbone has a collision, then we subdivide the edge e, producing a new node v_e, to which all contributions will be attached. Now, in a subtree t contributing to this edge, we identify all the pieces of t (that is, sub-subtrees) that should attach to that edge, and each of these pieces is then attached directly to v_e. Equivalently, we will contract edges in t between the different pieces that will attach onto e at the new node v_e, and then attach the resultant single subtree of t to v_e. Figures 2.6, 2.8, and 2.9 provide an example of this process.

Running time of SCM Let T_1 and T_2 be the trees to be merged, and let $m = \max(L(T_1), L(T_2))$. Since the strict consensus of two trees can be

computed in linear time [23], the SCM can also be computed in $O(m)$ time. Our implementation, however, can take $O(m^2 \log(m))$ time because we convert each tree into its set of bipartitions, where each bipartition is represented by two sets of integers. We explain how this time complexity is computed. Operations which we use, such as intersection of two sets, can take $O(gm \log(m))$ where m is the size of the larger set and g is the time taken to test for equality of two objects in the set. We are working with sets of bipartition and our implementation takes $O(m)$ time to test if two bipartitions are equal, where m is the size of the larger bipartition. Thus, total time taken is $O(m^2 \log(m))$.

Comparison of SCM against Strict Consensus Supertree method (SCS) The Strict Consensus Merger of two trees is very similar to the Strict Consensus Supertree (SCS) [34]. In fact, the only difference is how collisions are handled. (That is, if collisions do not occur, then the resultant supertrees are identical.) The example given in [34] illustrates the difference in SCS and SCM when there are collisions: the SCS tree is a strict contraction of the SCM tree, because it contracts additional edges located within sub-subtrees involved in the collision.

Conditions under which SCM returns the true tree when applied to DCM1 subproblems We can now state the conditions under which SCM will return the true tree given a DCM1 decomposition [37].

Theorem 2 *Let $(T, \{\lambda(e)\})$ be a model tree in the General Markov model and*