branch_and_bound

Function solve_with_container

```
pub fn solve_with_container<Node, Container>(
    container: Container,
) -> Option<Node>
where
    Node: Subproblem,
    Container: BnbAwareContainer<Node>,
```

Solve a problem with branch-and-bound / backtracking using a custom subproblem container with a custom strategy.

Until the container is empty, a subproblem is popped from the container and evaluated; when a subproblem is branched, the generated subnodes are put into the container to be retrieved in following iterations.

A container is, thus, responsible for the order in which subproblems will be examined, and can also implement additional features, such as early termination based on the current best value, early termination based on the number of iterations, eager or lazy evaluation, etc.

`solve_with_container` should be preferred for advanced use cases (e.g., custom order or unusual early terination conditions). If you want one of the basic options, use `solve`.