Throughout the chapter 4 to chapter 7, we learned several methods to solve the problems. And we did gambler problems with DP and MC. For a problem with accurate model we use DP to be efficient. Then we learned TD which combine the pros of DP and MC. However, when I review these algorithms, I found that they seem to only solve some simple tasks (like maze solver, grid world, the gambler and some single player games).

Can we combine these algorithms to solve a team game problem? For example, in a 5v5 competitive game, each role has different functions and tasks. How can we implement these algorithms to win the game?

In my view, they are two different conditions. First, we know some rules for the whole team to win but we are not informed detailed tasks for each role. I think we can try a 'father agent' with DP method because we know the rules in big view. And we set five 'child agent' with MC method because we don't know their value function. Based on this frame, we make 'child agent' to explore their tasks and we use 'father agent' to supervise these 'child agent', which means we want 'father agent' to check if the 'child agent' actions benefit the final goal.

Second, we may only know the characteristics about roles, but we don't know how to group these roles to win. Then we set 'child agent' with DP because we have accurate environments. We set 'father agent' with MC and let them try sample combinations to explore. Then it may find the best group, which has highest probability to win.

In addition, if we know nothing about the rules and role tasks, we may implement TD to both 'father agent' and 'child agent'. However, we have to start from zero, so I don't think these reinforcement methods are suitable for this kind of condition (team games without any information). It will cost a lot. Over all are my ideas for this question.