There are a number of planners available to use with Moveit!. These include :

1)**OMPL** - an open-source motion planning library

2)**STOMP** -Stochastic Trajectory Optimization for Motion Planning

3)**CHOMP**-Covariant Hamiltonian optimization for motion planning

4)**SBPL** - Search Based Planning library.

As it can be see from the official ROS wiki page :ROS page on planners, currently OMPL is the only planner fully supported leaving the others to be somewhere between partially supported to work in progress. However Moveit! has benchmarking capabilities for it's planners and this will be an exciting area to explore once more planners begin to be supported by Moveit!.

The status of integration of CHOMP is partially supported and hence I was not able to find enough resources to successfully change the planners from OMPL to CHOMP. Hence I will proceed to discuss my research into the inverse kinematic solvers that I worked with as part of this Questionnaire.

The IK solvers that I encountered include track_IK,KDL solver and the custom made IKFast solver. From my research I found out that **KDL solver** is ideally suited for kinematic chains having a DOF greater than 6 and have a restriction where a joint cannot have more than 1 DOF.

track_IK solver runs two different IK methods and combines the result to outperform the popular KDL solver. Also trac_IK handles joint-limited chains better than KDL without increasing solve time. However one should also note that significant improvements over KDL solver only surface when for large chains which is not the case with cool400 robot.

IKFast solver can analytically solve the kinematics equations of any complex kinematics chain, and generate language-specific files (like C++) for later use. The end result is extremely stable solutions that can run as fast as 5 microseconds on recent processors.

The setup I used to benchmark various solvers as follows. Assign an goal state to the robot arm and pass it on to the solver. The solvers are then rated based on time taken to find a solution and generate a simple path.

This was done with help of movegroup_interface_tutorial.cpp file that provided the target pose and commands to visualize the trajectory. The results are as follows:

Solver name	**Solution found in (s)**	**Path simplification time (s)**	**Change in states**
TracIK	0.084903 seconds	0.000601	97 to 2 states
KDL	0.058616 seconds	0.000647	59 to 2 states
IKfast	0.049947 seconds	0.000541	34 to 2 states

It is safe to deduce from the above results that IKfast turns out to the best planner for our robot since it was capable of finding a solution in 0.049947 seconds and also boasts of the least path simplification time of 0.000541. These are in lines with what I found earlier during research that IKfast tends to compute the quicker IK solutions.