Analysis

1. What do you see as the advantages/disadvantages of your task scheduler?

Advantages:
The task scheduler in the Space makes use of a Double ended queue (Dequeue) for storing the tasks. The tasks are stored in depth-first order rather than breadth-first order so that there are not too many subtasks (including successor tasks) lying in Space at any point in time without being processed. This can be better understood when considering the case of the Euclidean TSP task. The tasks get divided into subtasks until the desired level (though it's only level 2 in the assignment). If the scheduler uses a queue (breadth-first traversal) to maintain the ready tasks, then the subtasks of a level 'n' would be waiting until all the other tasks in level 'n' finish executing (the other tasks can again produce subtasks and successor tasks). This is avoided by using a stack-based approach.

2. How might you change the infrastructure to improve your parallel efficiency?

I've used a slightly different strategy while processing the tasks in the Computer. The Computer runs a separate thread to process the task that is obtained from the ready queue in Space. Then, it either runs the task or decomposes it into subtasks and then stores the result of the execution in the Space. Since this runs as a separate thread, the computer doesn't wait for the RMI call to the Space to return. It becomes immediately available to process the next task in the ready queue. This is achieved by using a Space proxy in the Computer.

3. What issues are involved in generalizing your infrastructure to a network of Spaces?

When there a network of spaces, there are certain issues that one needs to address. The data structures that are maintained to hold the ready and waiting tasks need to be replicated across all the Spaces. Or, some kind of shared memory/storage mechanisms need to be used. Otherwise, there could be a situation where a Computer could put the result of execution of one task into one Space and that Space doesn't have the information about the successor of that task. Also, Computers would need a remote reference to all the Spaces.

Another approach is to have a master-slave configuration between the Spaces, wherein one Space acts as the master and all others act as Slaves. The client and the Computers communicate only with the master and the master takes care of finding the slave that has the required data and serving it. This again incurs a lot of communication overhead. Many problems with respect to the co-ordination between the master and slaves that are typical of a distributed environment need to be addressed. Again, if the master goes down, a new master needs to be elected and the control passed.