Selecting a proper designing technique for a parallel algorithm is the most difficult and important task. Most of the parallel programming problems may have more than one solution. In this chapter, we will discuss the following designing techniques for parallel algorithms –

- Divide and conquer
- Greedy Method
- Dynamic Programming
- Backtracking
- Branch & Bound
- Linear Programming

**Divide and Conquer Method**

In the divide and conquer approach, the problem is divided into several small sub-problems. Then the sub-problems are solved recursively and combined to get the solution of the original problem.

The divide and conquer approach involves the following steps at each level –

- **Divide** – The original problem is divided into sub-problems.
- **Conquer** – The sub-problems are solved recursively.
- **Combine** – The solutions of the sub-problems are combined together to get the solution of the original problem.

The divide and conquer approach is applied in the following algorithms –

- Binary search
- Quick sort
- Merge sort
- Integer multiplication
- Matrix inversion
- Matrix multiplication

**Greedy Method**

In greedy algorithm of optimizing solution, the best solution is chosen at any moment. A greedy algorithm is very easy to apply to complex problems. It decides which step will provide the most accurate solution in the next step.

This algorithm is a called greedy because when the optimal solution to the smaller instance is provided, the algorithm does not consider the total program as a whole. Once a solution is considered, the greedy algorithm never considers the same solution again.

A greedy algorithm works recursively creating a group of objects from the smallest possible component parts. Recursion is a procedure to solve a problem in which the solution to a specific problem is dependent on the solution of the smaller instance of that problem.

**Dynamic Programming**

Dynamic programming is an optimization technique, which divides the problem into smaller sub-problems and after solving each sub-problem, dynamic programming combines all the solutions to get ultimate solution. Unlike divide and conquer method, dynamic programming reuses the solution to the sub-problems many times.
Recursive algorithm for Fibonacci Series is an example of dynamic programming.

**Backtracking Algorithm**

Backtracking is an optimization technique to solve combinatorial problems. It is applied to both programmatic and real-life problems. Eight queen problem, Sudoku puzzle and going through a maze are popular examples where backtracking algorithm is used.

In backtracking, we start with a possible solution, which satisfies all the required conditions. Then we move to the next level and if that level does not produce a satisfactory solution, we return one level back and start with a new option.

**Branch and Bound**

A branch and bound algorithm is an optimization technique to get an optimal solution to the problem. It looks for the best solution for a given problem in the entire space of the solution. The bounds in the function to be optimized are merged with the value of the latest best solution. It allows the algorithm to find parts of the solution space completely.

The purpose of a branch and bound search is to maintain the lowest-cost path to a target. Once a solution is found, it can keep improving the solution. Branch and bound search is implemented in depth-bounded search and depth-first search.

**Linear Programming**

Linear programming describes a wide class of optimization job where both the optimization criterion and the constraints are linear functions. It is a technique to get the best outcome like maximum profit, shortest path, or lowest cost.

In this programming, we have a set of variables and we have to assign absolute values to them to satisfy a set of linear equations and to maximize or minimize a given linear objective function.