A Survey on Algorithm Design Techniques

A.Mahalakshmi[1], Devi selvam[2], L.Viji[3]
[1, 2] Department of CSE
[3] Department of IT
[1, 2] Sri Shakthi Institute of Engineering and Technology,Tamilnadu,India
[3] Velalar College of Engineering and Technology,Tamilnadu,India

Abstract- *Algorithms are recognized as steppingstone of computing. This survey describes different algorithm design techniques such as Brute force, Greedy, Divide and Conquer, Dynamic programming, Backtracking, Branch and Bound and many more. It describes how a particular technique is used for a specific problem by considering various parameters. This survey compares different algorithm design strategies along with applications.*

Keywords- Brute force, Greedy, Divide and Conquer, Dynamic programming, Backtracking, Branch and Bound.

I. INTRODUCTION

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output for any valid input. There exist a number of algorithms, every algorithm is problem specific. The choice of an algorithm may not just depend on computational complexity; it also depends upon the characteristics, advantages and disadvantages. This report shows how an algorithm is best for a particular situation, based upon their advantages and comparison with others. Some algorithm design strategies are problem specific means they are well suited for some specific problem and have disadvantage against another problem. The basic question here is: How to choose the approach? First, by understanding the problem, and second, by knowing various problems and how they are solved using different approaches.

Figure 1.

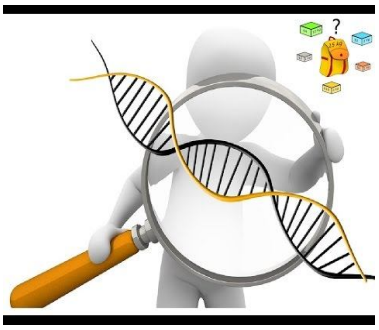

Figure 2.

II. PURPOSE OF ALGORITHM ANALYSIS

There are no well-defined standards for writing algorithms. Rather, it is problem and resource dependent. Algorithms are never written to support a particular programming code.We should know the problem domain, for which we are designing a solution.

The problem domain may belongs to one of the following categories

1. Sorting
2. Searching
3. Combinatorial Problems
4. Numerical Problems
5. Graph Problems

A. problem can be solved in more than one ways.

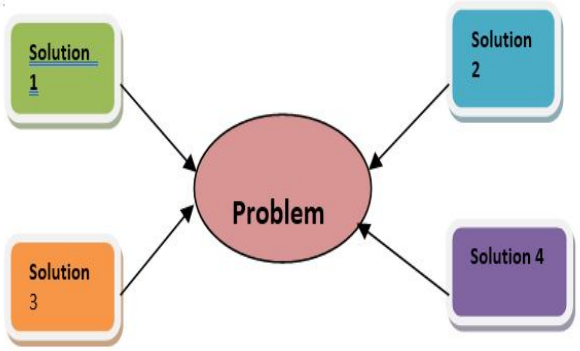

Figure 3.

Thus, we will have many solutions for a single issue. Among that to choose the best solution, we are in need of Analysis.

ALGORITHM COMPLEXITY

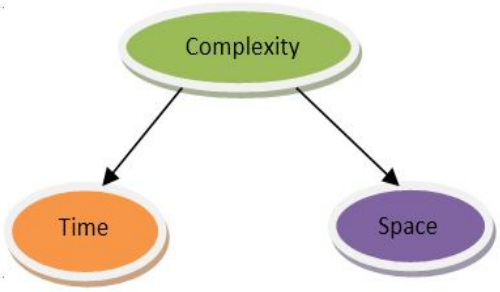

Figure 4.

Suppose A is an algorithm and n is the size of input data, the time and space used by the algorithm A are the two main factors, which decide the efficiency of A.

- Time Factor − Time is measured by counting the number of basic operations, such as comparisons in the searching algorithm.
- Space Factor − Space is measured by counting the maximum memory space required by the algorithm.
- Asymptotic analysis refers to computing the running time of any operation in mathematical units of computation. Usually, the time required by an algorithm falls under three types –
- Best Case − Minimum amount of time required for program execution.
- Average Case − Average time required for program execution.
- Worst Case − Maximum amount of time required for program execution.

III. ALGORITHM DESIGN TECHNIQUES

Algorithm design technique is the systematic approach for solving issues under various domains. They provide templates suited to solving a broad range of diverse problems. They can be translated into common control and data structures provided by most high-level languages.

DESCRIPTION:

A BRUTE FORCE

Brute force is a problem solving approach in which we will "generate all possible solutions and select the optimal one ".It is a straightforward approach proceeds in simple and obvious way usually based directly on problem's statement and definitions of the concepts involved.

A. Exhaustive search

It is a brute-force approach, applied to solve combinatorial problems. The algorithm that tries every possible solution is known as exhaustive search. Also known as the British Museum algorithm. It can be used to reduce the search space.

B. GREEDY TECHNIQUE

A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment. A greedy algorithm always makes the next available best choice. Here at each step, decision made must be

- Feasible - The solution should satisfy the given constraints
- Locally Optimal - Best among all possible feasible solutions
- Irrevocable - Once made, it cannot be changed

Example-Change Making Problem

This problem is to count to a desired value by choosing the least possible coins and the greedy approach forces the algorithm to pick the largest possible coin. If we are provided coins of Rs 1, 5, 10 and 20 and we are asked Rs 48 then the greedy procedure will be −

- 1 − Select 3 coins of 1 Rs, the remaining count is 45
- 5 − Then select one 5 Rs coin, the remaining count is 40
- 20 – And finally, selection of two 20 Rs, solves the problem

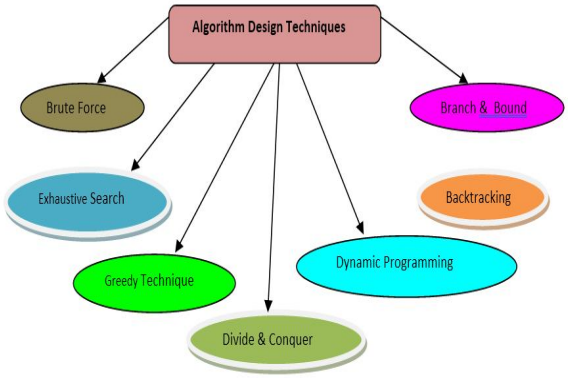

Figure 5. Various algorithm design techniques are

C. DIVIDE AND CONQUER

There are three steps.

- Divide: Breaking the problem into several sub-problems that are similar to the original problem but smaller in size.
- Conquer: Solve the sub-problem recursively (successively and independently). If the sub problems are small enough, solve them in brute force fashion.
- Combine: Combine these solutions to sub-problems to create a solution to the original problem.

Divide and conquer algorithms have several appealing properties that make them a good match for modern parallel machines.

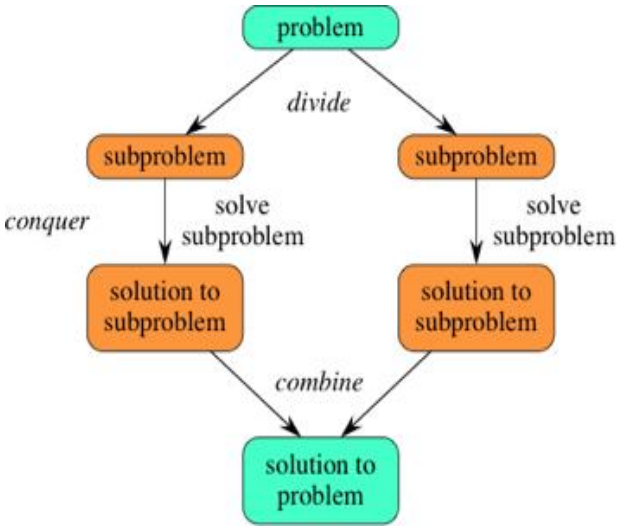

Figure 6.

D. DYNAMIC PROGRAMMING

Dynamic programming is the method of solving the issue by overlapping of sub-problems.

Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. It uses the concept called “Principle of Optimality”, which can be stated as” in a optimal sequences of choices, each subsequence must also be optimal”.

There are two ways of doing this.

1.) Top-Down : Start solving the given problem by breaking it down. If you see that the problem has been solved already, then just return the saved answer. If it has not been solved, solve it and save the answer. This is referred to as Memorization.
2.) Bottom-Up : Analyze the problem and see the order in which the sub-problems are solved and start solving from the trivial subproblem, up towards the given problem. In this process, it is guaranteed that the subproblems are solved before solving the problem. This is referred to as Dynamic Programming.

Knapsack problem

Given a set of n items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

Figure 7.

E. BACKTRACKING

The principle idea of backtracking is to construct solutions on component at a time and evaluate such partially constructed candidates as follows. If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component. If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered. In this case the algorithm backtracks to replace the last component of the partially constructed solution with its next option. Recur¬sion is the key concept in back¬track¬ing.

State space tree

This method is based on construction of state space tree. It is the tree organization of entire solution space(which has complete solution path and dead ends) . Where the root represents the initial state before the search for a solution begins. The nodes at each nth level represent the choices for the nth component.

N Queens Problem

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, following is a solution for 4 Queen Problem.

State Space Tree

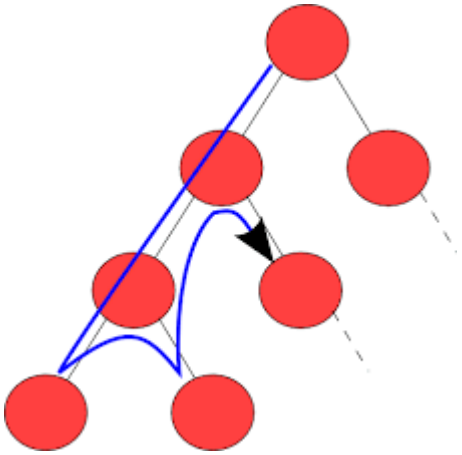

Figure 8.

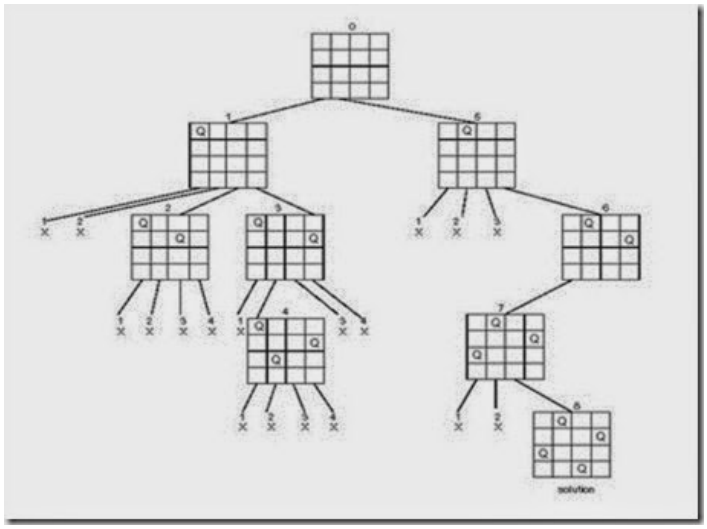

Figure 9.

F. BRANCH AND BOUND

In this method, based on the bound (Lower/Upper) values, the state space tree will be expanded.

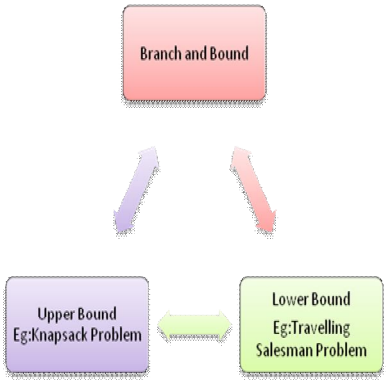

Figure 10.

We know that maximum valuable subset should be identified to solve knapsack problem. Hence, it comes under upper bound. But in case of travelling salesman problem, minimum distance should be identified. Hence it belongs to lower bound.

Figure 11.

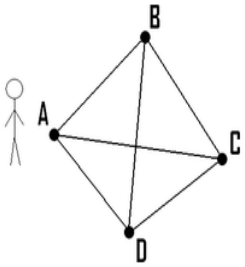

Figure 12.

S salesman wants to visit all cities A, B, C and D. What is the best way to do this (cheapest airline tickets, and minimal travel time)?

Table 1.

Algorithm Design Techniques	**Applications**
Brute Force	Selection sort, Bubble sort, String matching and Convex-hull problems.
Exhaustive Search	Travelling Salesman Problem, Job Assignment Problem.
Greedy Technique	Graph Coloring, Huffman Coding, Dijkstra's Shortest Path Algorithm,Container Loading Problem,Prim's & Kruskal's Minimum Spanning Tree Algorithm.
Divide and	Merge Sort, Quick

Conquer	Sort, Binary Search, Strassen's Matrix Multiplication, Closest pair and Convex Hull Problems.
I. DYNAMIC PROGRAMMING	Computing Binomial Coefficient , Knapsack Problem, Floyd, Warshall, Optimal Binary Search Tree.
Backtracking	N Queens Problem, Subset Sum Problem, Hamiltonian Circuit Problem
Branch and Bound	Knapsack Problem, Travelling Salesman Problem, Job Assignment Problem

Table 2.

Algorithm Design Techniques	Advantages	Disadvantages
Brute Force	- High speed - Very Simple - Fairly simplistic attack that doesn't require a lot of work to setup or initiate.	-Very hardware intensive. -Requires a lot of processing power.
Exhaustive Search	-It's widely applicable. -It's simple. -Reliable and universal method.	-Cost of generating candidate solutions. -Impractical (intractable) on problems that are too big.
Greedy Technique	-Easy to implement. -Repeatedly taking the next available best choice is usually linear work. -Much cheaper than most other algorithms.	-Don't work for some problems. -Much slower. -It is not an automatic.
Divide and Conquer	-Parallelism. -Cache Performance. -Memory Access. -Used to solve difficult problems. -High Efficiency.	-Recursion Overhead. -For efficiency reasons, programs often use pointers into arrays and pointer arithmetic to identify sub problems.
II. DYNAMIC PROGRAMMING	III. **-USED TO AVOID CALCULATING SAME STUFF TWICE.**	-Used only for overlapping of sub problems. -It's very tough task to identify recursive formula.
Backtracking	-It is a step-by-step representation of a solution- very easy to understand. -It is independent of programming language. -Simple to code.	-Multiple function calls are used, so expensive. -Inefficient when there is lots of branching from one state. -Requires large amount of storage space

	- Easy to debug .	
Branch and Bound	-It is used to solve optimization problem. -It may traverse the tree in any manner, DFS or BFS. -It completely searches the state space tree to get more number of optimal solutions.	-Finding pruning strategies require clever thinking technologies

III. METHODOLOGY

The main key of the study to make an attempt of solving the material handling station with the implementation of the discrete even simulation i.e. modeling the entire procedural steps exactly by using queues and the probability and statistic functions the simple model of 3 station material handling system is considered and the steps which as the intermediate process elements such as the machining centers inspection stations, packing, conveyor belts etc. are part of this study.

IV. COMPARISON BETWEEN A LGORITHM DESIGN TECHNIQUES

Table 3.

GREEDY TECHNIQUE	BRUTE FORCE
Very Fast	Slower than Greedy method.
Tries to find a localized optimum solution, which may eventually lead to globally optimized solutions	Used to solve small class of problems
BACKTRACKING	**BRUTE FORCE**
In each step, this method checks- if it satisfies all the conditions. If it does : continues generating subsequent solutions. If not : goes one step backward to check for another path	This method generates all possible combinations and then checks if any of them is the exact solution

Table 4.

DIVIDE AND CONQUER	DYNAMIC PROGRAMMING
Inputs are divided at prespecified deterministic points.	It tries every possible split points.
Principle of Optimality concept is not used.	Principle of Optimality concept is used.
Solutions are combined to achieve an overall solution	Use the output of a smaller sub-problem and then try to optimize a bigger sub-problem.
1) Memory access Divide-and-conquer algorithms naturally tend to make efficient use of memory caches.	Here, each sub-problem is solved only once. There is no recursion. The key in dynamic programming is remembering.
2) The sub-problems are independent of each other.	Here, the sub-problems are not independent of each other.
D&C algorithms that are time-efficient often have relatively small recursion depth. *3)*	DP gives a better performance for many NP complete problems like TSP. Though the space needed is large, it reduces the complexity well.

Table 5.

BACKTRACKING	BRANCH AND BOUND
Applicable to both Optimization and Non-optimization problems.	Applicable only to Optimization problems.
State space tree is generated by using depth first search method.	State space tree is generated by using breadth first search method.
It don't consider any bound values.	State space tree is generated based on bound values.

V. CONCLUSION

The main goal of this survey is to compare various algorithm design techniques based on the advantages, disadvantages and applications. Usually a given problem can be solved using various approaches however it is not wise to settle for the first that comes to mind. More often than not, some approaches result in much more efficient solutions than others. In future, the best algorithm design technique suitable for all types of problems can be identified.

REFERENCES

[1] Shailendra Nigam, Dr. Deepak Garg "Choosing Best Algorithm Design Strategies For a Particular Problem", In Proceedings of the IEEE International Advance Computing Conference (IACC 09), Thapar University Patiala, India (6-7 March 2009)

[2] Exhaustive Search, Chapter 38 in Algorithms by Robert Sedgewick; Addison- Wesley, 1983.

[3] A New Road Map of Algorithm Design Techniques by Anany Levitin in Dr. Dobb's Journal, April 2008.

[4] Greedy Algorithms, Chapter 17 in Introduction to Algorithms by Thomas Cormen, Charles Leiserson, and Ronald Rivest, MIT Press, 1999.

[5] Greedy Algorithms, Chapter 5 in Algorithms by Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vizirani, McGraw-Hill, 2008

[6] http://ignou.ac.in/userfiles/SandeepFINAL_Unit1_Intro_21-03-2013.pdf

[7] http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250_Weiss/L28- Design.htm

[8] https://gradeup.co/algorithm-design-techniques-i-6c2dec3e-c0da-11e5-abd0-9e294d44e4af

[9] http://gdeepak.com/thesisme/Thesis-Choosing%20Best%20Algorithm%20Design%20Strategies%20For%20a%20Particular%20Problem.pdf

[10] https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm

[11] http://stackoverflow.com/questions/30025421/difference-between-backtracking-and-branch-and-bound

[12] http://www.algorithmist.com/index.php/Exhaustive_Search

[13] https://people.csail.mit.edu/rinard/divide_and_conquer/

[14] [https://en.wikipedia.org/wiki/Bellman_equation