Construtive Heuristics

Nicolas Von Dolinger Moreira Rocha
2022035571
Federal University of Minas Gerais - UFMG

September 1, 2025

Abstract

This document presents a resume about the chapters 15 (15.1, 15.2, 15.3), 16 (16.1, 16.2) and 35 (35.1, 35.2, 35.3) of Cormen et al., "Introduction to Algorithms", 2009; and about chapter 8 (8.1, 8.2) Arts and Lenstra J. K.,"Local Search in Combinatorial Optimization", John Wiley and Suns, 1997.

1 Introduction to Algorithms

1.1 Chapter 15: Dynamic Programming

1.1.1 Rod cutting

This problem is the following: given rod of length n inches and a table of prices, determine the maximum revenue obtainable by cutting up the rod and selling the pieces according to its price on the table.

The author shows that this problem can be solved by an recursive method, but the complexity of this algorithm is exponential, because it recompute the same subproblems a lot of times.

So on, the dynamic method works by saving the solution of a subproblem and solving it only once, and in the future we just need to get it result instead of compute it again. This method uses additional memory to save computational time, serving as an example of time-memory trade-off.

The author presents the most common ways to implement dynamic-programming approach: the top-down with memoization, that writes the procedure recursively and saves the result of each subproblems, so that's no need to compute it again; and bottom-up method, that solve the smaller problems first and when solving a particular subproblem, all the smaller subproblem need to get this solution have already been computed.

1.1.2 Matrix-chain multiplication

This problem in the following: given a sequence (chain) of matrices to be multiplied , we wish to compute the product of this matrices fully parenthesize in a way that minimizes the number of scalar multiplications. The matrix multiplication is associative, so all the parenthesizations yield the same product and how we parenthesize a chain of matrices can have a dramatic impact on the cost of evaluating the product. It is important to notice

that the problem is not multiplying the matrices, just giving the order it will make the minimum cost.

To apply dynamic programming method to this problem it also follow the same ideia: characterize the structure of an optimal solution, recursively define the value of an optimal solution, compute the value of an optimal solution and construct an optimal solution from computed information.

1.1.3 Elements of dynamic programming

In this subsection, the author shows how an exponential problem becomes a polynomial problem when using dynamic programming with memoization. This is possible when the subproblems of a problem are overlapping; you can then use memoization with a table to store the answers to these problems.

1.2 Chapter 16: Greedy Algorithms

1.2.1 An activity-selection problem

The author presents a kind of algorithm approach that is called "greedy", that makes the best choice at the moment, and in a large amount of problems, gives the optimal solution.

To exemplify that, it presents the following problem, we have a set of activities that use a resource, which can serve only one activity at a time, each activity has a start and finish time, and each activity is compatible if they do no overlap. This problem is called activity-selection problem.

The greedy approach to solve this problem is to choice the activity with the earliest finish time, that will be included in the answer, following the rules of the problem. The author also proves that this strategy leads to the optimal solution.

1.2.2 Elements of the greedy strategy

A greedy approach is useful, but not always leads to a optimal solution depending on the problem, so it needs to follows two properties. The first property is the greedy-choice, in greedy approach that choice that we make in a moment doesn't depends on the solution of the subproblems, that's what differs it from dynamic programming. The second property is a optimal substructure, that implies in affirm that the greedy choice made before, combined with the greedy choice already made, yields an optimal solution to the original problem.

1.3 Chapter 35: Approximation Algorithms

1.3.1 The vertex-cover problem

The vertex-cover is version of minimization of a NP-complete problem. Here is needed to find a the vertex cover of minimum size in a given undirected graph. The author proves that this problem has a 2 approximation factor, i.e., using this algorithm, that is works on polinomial time, the answer given it's always at most 2 times worse than the optimal answer.

1.3.2 The traveling-salesman problem

In this problem, we are given a complete undirected graph and we must find a hamiltonian cycle of minimum cost. The author shows that the algorithm using the triangle inequality(computing a MST and using the triangle inequality to build the path) have a 2 aproximation factor. The author also shows that if Pis different of NP, the for any constant p >= 1, there is no polynomial-time approximation algorithm with approximation ratio p for the general traveling-salesman problem.

1.3.3 The set-covering problem

This problem in an optimization problem that models many problems that require resources to be allocated. The author shows two algorithms: one with p(n)(the size of the maximum subset that covers the main set (X) elements) approximation factor and another with ln(X) + 1 approximation factor.

2 Local Search in Combinatorial Optimization

2.1 Chapter 8: The traveling salesman problem: a case study

2.1.1 Introduction

This section presents what is the propourse of the author in this chapter. The problem that is the main topic of study is the Travelling Salesman Problem (TPS), specifically the symmetric version. The author is concerned about the results that some famous papers on literature presents and he is interested in the relationship between what is known theoretically about the algorithms and what can be observed empirically.

2.1.2 Tour Construction Heuristics

Each TSP heuristic can be evaluated in terms of two key parameters: its running time and the quality of the tours it produces. The author will focus on the best known heuristics in terms of results and execution time.

The author focus on four heuristics: Nearest Neighbor(always go next to the nearest unvisited location), Greedy(repeatedly choose the shortest edge if it is not yet in the tour and if adding it would not create a degree-3 vertex on the path), Clarke-Wright(it start's with a multigraph in which every nonhub vertex is connected by two edges of the hub and for each pair o nunhub cities let the savings be the amount by wich the tour would be shortened if the salesman went directly from one city to the other passing by the hub, apply the Greedy heuristic to this metric) and Christofides(find a MST for the set of cities and compute a minimum length matching on the vertices of odd degreee in the MST, combine this two and get a a graoh in which vertex has even degree, this graph must contain an Euler tour and a TS tour can be constructed by traversing this cycle while takin shortcuts to avoid multiply visited cities). Each one has a particular significance in the context of local search.

References

[1] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009.

[2] AARTS, E.; LENSTRA, J. K. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.