1 Introduction

Combinatorial optimization problems. Those are the type of algorithms that arise in countless applications, from billion-dollar operations to everyday computing task; they are used by airline companies to schedule and price their flights, by large companies to decide what and where to s tock in their warehouses, by delivery companies to decide the routes of their delivery trucks, by Netflix to decide which movies to recommend you, by a GPS navigator to come up with driving directions and by word-processors to decide where to introduce blank spaces to justify (align on both sides) a paragraph.[3]

Considering the previous paragraph, we conclude that combinatorial optimization problems arise certainly in all areas of technology and industrial management and in other several applications. Very simple examples is the task of finding the shortest path from point A to B in the road network of C or scheduling exams for given courses at a university.In theoretical computer science and applied mathematics, combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects. Therefore, it is a subset of mathematical optimization, which is related to operational research, algorithm theory, and computational complexity theory. It has important applications in several fields, including AI (artificial intelligence), mathematics, and software engineering.

In this chapter, we introduce combinatorial optimization. Later on, we discuss some of its methods. Before talking about combinatorial optimization let us understand first what do optimization stands for, and check up on its categories and techniques**.**

1.1 What Is Optimization

Optimization is the process of adjusting the inputs to or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or result (Figure 1.1 [4]). The input consists of variables; the process or function is known as the cost function, objective function, or fitness function; and the output is the cost or fitness. If the process is an experiment, then the variables are physical inputs to the experiment.

Figure1.1– Optimization process diagram [1]

1.2 Formal definition of an optimization problem

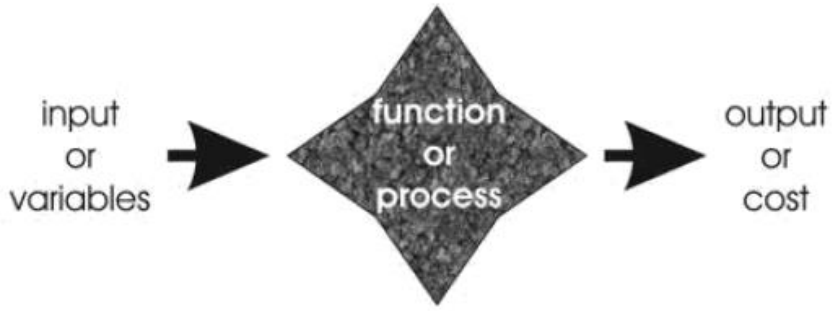

Figure 1.1 Diagram of a function or process that is to be optimized. Optimization varies the input to achieve a desired output.

From a mathematical viewpoint, an optimization problem may be posed as follows [1]:

minimize the function $f: S \to \mathbb{R}$, that is, *compute the pair* $(x, f(x))$ *such that* $f(x)$ *is the minimum value element of the set* $\{x, f(x); x \in S\}$. *We also use the two simplified forms below:*

$$\min_{x \in S} f(x) \text{ormin}\{f(x); x \in S\}.$$

The function $f: S \to \mathbb{R}$ is called the *objective function(cost function)*, and its domain S, the set of feasible solutions, is called the *solution set*.

- We may restrict attention to minimization problems, since any maximization problem can be converted into a minimization problem by just replacing f with $-f$. An optimality condition is a necessary or a sufficient condition for the existence of a solution to an optimization problem.
- An ***optimization problem*** is a set of instances of an optimization problem. Informally, in an ***instance*** we are given, the *input data* and have enough information to obtain a

solution. That is, a ***problem*** is a collection of instances, usually all generated in a similar way.

1.3 Categories of Optimization [5]

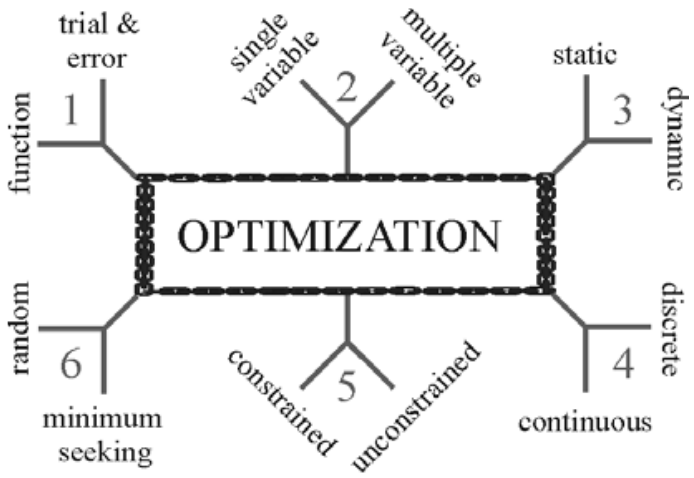

Figure 1.2 – Optimization categories [1]

Figure 1.2 divides optimization algorithms into six categories. None of these six views or their branches are necessarily mutually exclusive. For instance, some of the variables may be discrete and others continuous.

1. Trial-and-error optimization refers to the process of adjusting variables that affect the output without knowing much about the process that produces the output. A simple example is adjusting the rabbit ears on a TV to get the best picture and audio reception. . In contrast, a mathematical formula describes the objective function in function optimization.
2. If there is only one variable, the optimization is one-dimensional. A problem having more than one variable requires multidimensional optimization. Optimization becomes increasingly difficult as the number of dimensions increases. Many multidimensional optimization approaches generalize to a series of one-dimensional approaches.
3. Dynamic optimization means that the output is a function of time, while static means that the output is independent of time. For example, there were several ways to drive back and forth to work. What was the best route? From a distance point of view, the problem is static, and the solution can be found using a map or the odometer of a car. In practice, this problem is not simple because of the myriad of variations in the routes. The shortest route isn't necessarily the fastest route. Finding the fastest route is a dynamic problem whose solution depends on the time of day, the weather, accidents, and so on.

4. Optimization can also be distinguished by either discrete or continuous variables. Discrete variables have only a finite number of possible values, whereas continuous variables have an infinite number of possible values. If we are deciding in what order to attack a series of tasks on a list, discrete optimization is employed. Discrete variable optimization is also known as combinatorial optimization, because the optimum solution consists of a certain combination of variables from the finite pool of all possible variables. However, if we are trying to find the minimum value of *f(x)* on a number line, it is more appropriate to view the problem as continuous.

5. Variables often have limits or constraints. Constrained optimization incorporates variable equalities and inequalities into the cost function. Unconstrained optimization allows the variables to take any value. A constrained variable, often converts into an unconstrained variable through a transformation of variables. Most numerical optimization routines work best with unconstrained variables. Consider the simple constrained example of minimizing ***f(x)*** over the interval ***-1< x< 1***.The variable converts x into an unconstrained variable u by letting ***x = sin(u)*** and minimizing ***f(sin(u))*** for any value of *u*. When constrained optimization formulates variables in terms of linear equations and linear constraints, it is called a linear program. When the cost equations or constraints are nonlinear, the problem becomes a nonlinear programming problem.

6. Some algorithms try to minimize the cost by starting from an initial set of variable values. These minimum seekers easily get stuck in local minima, but tend to be fast. They are the traditional optimization algorithms and are generally based on calculus methods. Moving from one variable set to another is based on some determinant sequence of steps. On the other hand, random methods use some probabilistic calculations to find variable sets. They tend to be slower but have greater success at finding the global minimum.

1.4 Optimizations techniques

The goal of an optimization algorithm is to look for the best of all possible solutions to a problem. Research in optimization has been very important in computer science from the beginning. Optimization is a very dynamic research field because new challenges appear every day: new engineering problems, new industrial situations, new services that need optimization, and a long list of other situations that constantly challenge the existing optimization techniques [7].

In this dissertation, we concentrate on combinatorial optimization problems, which is by far the most popular type of optimization that has attracted the interest of many researchers in the recent years.

Combinatorial optimization problems are intriguing because they are often easy to state, but often quite difficult to solve, which is captured by the fact that many of them are NP-hard. An NP-hard problem is one for which an optimal solution is conjectured by computer scientists to be unknown in the polynomial time complexity related to the increasing problem size, that is, an NP-hard problem requires more steps than can be grounded by a polynomial. Thus, the main difficulty in solving combinatorial optimization problems arises from the fact that the number of feasible solutions from which the best solutions needs to be selected grows exponentially with the size of the problem instances.

Because of their difficulty and enormous practical importance, a large number of solution techniques for tackling them have been proposed. The available algorithms can roughly be classified into two main categories: exact and approximate algorithms [8].

1.4.1 Exact algorithms

Exact algorithms are guaranteed to find the optimal solution and to prove its optimality for every finite size instance of a combinatorial optimization problem within an instance-dependent, finite run-time, or prove that no feasible solution exists. The most basic exact method that always works is called complete enumeration. It lists all possibilities and chooses the best one. Clearly, this is simple and it will work. However, this method is generally impractical as one might see at first sight because the list would take too long to compute in most cases.

The challenge is thus to exclude many possible configurations from an enumeration *priori*, We might, for instance, break up our solution space into sections and these sections into further sections. We therefore have a hierarchy of sections to search.

It may seem paradoxical to exclude certain locations from a search without looking there, but in mathematical problems, we usually have some knowledge about the problem at hand that allows us to make such inferences a priori.One particular strategies for using such information is branch-and-bound method [9].

1.4.2 Approximate algorithms

When optimal solutions cannot be computed efficiently in practice, the only possibility is to trade optimality for efficiency. In other words, the guarantee of finding optimal solutions can be sacrificed for the sake of getting very good solutions in polynomial time. Heuristics and metaheuristics are two classes of approximate algorithms that seek to obtain this goal.

1.5 Heuristic Methods

The term "heuristic" originates from the Greek word "heuriskein" which means "to discover" or "to find".

A heuristic is a technique which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is [10].

Dedicated heuristic solution approaches have been developed which aim at providing good solutions in reasonable time for a given problem. However, such methods have two major drawbacks [11]:

1) They are tailored to a specific problem and their adaptation to other problems is difficult and in many cases even impossible.
2) They are typically designed to "build" one single solution in the most effective way, whereas most decision problems have a vast number of feasible solutions

1.5.1 Heuristic searches

Two different types of heuristics can be identified.

- **Constructive Heuristics**: constructive heuristics are mainly problem specific and try to construct one single solution with best possible quality by carefully selecting promising solution elements.
- **Search Heuristics** (Heuristic Search Strategies): Search heuristics implement a search in the solution space of a given problem, during which they examine many different solutions in order to find the best possible one.

1.5.2 Taxonomy of heuristic search strategies

In fact, we can identify three, problem-independent, basic principles of heuristic search:

a) Repeated solution construction: a new solution is obtained by constructing a new one from scratch.

b) **Repeated solution modification**: a new solution is obtained by modifying an existing one.

c) **Repeated solution recombination**: a new solution is obtained by recombining two or more existing solutions.

1.6 Metaheuristics

The term metaheuristic generally refers to approximate algorithms for optimization that are not specifically expressed for a particular problem.

A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration [12].

A metaheuristic can be also defined as a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework, which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem [13].

In recent studies, metaheuristic algorithms are widely used, and organized as the most promising approach for tackling hard combinational optimization problems. Examples of such metaheuristics include simulated annealing, tabu search, iterated local search, variable neighbourhood search algorithms, greedy randomized adaptive search procedures, genetic algorithms and ant colony optimization.

Although metaheuristics are widely used techniques, the how and why they work effectively for specific problems and for others not, is still not well understood.