Motivation

Celso C. Ribeiro (celso@ic.uff.br)

University of Vienna

Metaheuristics – 2017-10-11

Overview of talk

- Optimization problems
    - Definition
    - Continuous vs. discrete variables: combinatorial optimization
    - Complexity
    - Motivation
        - Shortest path problem
        - Minimum spanning tree problem
        - Steiner tree problem in graphs
        - Maximum clique problem
        - Knapsack problem
        - Traveling salesman problem
- Exact vs. approximate methods
    - Constructive heuristics
    - Local search
    - Metaheuristics
- Concluding remarks

Optimization problems – Definition

An optimization problem can be cast as

$$\text{optimize } f(S) \tag{1}$$

subject to

$$S \in F, \tag{2}$$

where F is a *feasible set* of solutions and f is a real-valued *objective function* that associates each *feasible solution* $S \in F$ to its cost or value $f(S)$.

Optimization problems – Definition

An optimization problem can be cast as

$$\text{optimize } f(S) \tag{1}$$

subject to

$$S \in F, \tag{2}$$

where F is a *feasible set* of solutions and f is a real-valued *objective function* that associates each *feasible solution* $S \in F$ to its cost or value $f(S)$.

Global minimum

A global optimum of a minimization problem is a solution $S^* \in F$ such that $f(S^*) \leq f(S)$, $\forall S \in F$.

Optimization problems – Definition

An optimization problem can be cast as

$$\text{optimize } f(S) \tag{1}$$

subject to

$$S \in F, \tag{2}$$

where F is a *feasible set* of solutions and f is a real-valued *objective function* that associates each *feasible solution* $S \in F$ to its cost or value $f(S)$.

Global minimum

A global optimum of a minimization problem is a solution $S^* \in F$ such that $f(S^*) \leq f(S)$, $\forall S \in F$.

Global maximum

A global optimum of a maximization problem is a solution $S^* \in F$ such that $f(S^*) \geq f(S)$, $\forall S \in F$.

Optimization problems – Complexity

Optimization problems are represented by:

- continuous variables, which in principle can take any real value;
- discrete variables, which can take only a finite (or countable) set of values:
    - Combinatorial optimization problems reduce the search for a solution to a finite (or countable) set (typically formed by binary or integer variables, permutations, paths, trees, cycles, or graphs).

Optimization problems – Complexity

Optimization problems are represented by:

- continuous variables, which in principle can take any real value;
- discrete variables, which can take only a finite (or countable) set of values:
    - Combinatorial optimization problems reduce the search for a solution to a finite (or countable) set (typically formed by binary or integer variables, permutations, paths, trees, cycles, or graphs).

Some problems are intrinsically harder to solve than others: state-of-the-art algorithms to solve them can be very expensive in terms of the computation time needed to find a global optimum and only small problems can be solved in a reasonable amount of time.

Understanding the inner computational complexity of each problem is an absolute need for the identification and development of an appropriate, effective, and efficient algorithm for its solution.

Optimization problems – Motivation

Shortest path problem

Suppose a number of cities are distributed in a region and we want to travel from city s to city t. The distances between each pair of cities are known beforehand. We can either go directly from s to t if there is a road directly connecting these two cities, or start in s, traverse one or more cities, and end up in t. A path from s to t is defined to be a sequence of two or more cities that starts in s and ends in t. The length of a path is defined to be the sum of the distances between consecutive cities in this path. In the *shortest path problem*, we seek, among all paths from s to t, one which has minimum length.

Optimization problems – Motivation

Shortest path problem

Suppose a number of cities are distributed in a region and we want to travel from city s to city t. The distances between each pair of cities are known beforehand. We can either go directly from s to t if there is a road directly connecting these two cities, or start in s, traverse one or more cities, and end up in t. A path from s to t is defined to be a sequence of two or more cities that starts in s and ends in t. The length of a path is defined to be the sum of the distances between consecutive cities in this path. In the *shortest path problem*, we seek, among all paths from s to t, one which has minimum length.

Minimum spanning tree problem

Suppose that a number of points spread out on the plane have to be interconnected. Once again, the distances between each pair of points are known beforehand. Some points have to be made pairwise connected, so as to establish a unique path between any two points. In the *minimum spanning tree problem*, we seek to determine which pairs of points will be directly connected such that the sum of the distances between the selected pairs is minimum.

Optimization problems – Motivation

Steiner tree problem in graphs

Assume that a number of terminals (or clients) have to be connected by optical fibers. The terminals can be connected either directly or using a set of previously located hubs at fixed positions. The distances between each pair of points (be them terminals or hubs) are known beforehand. In the *Steiner tree problem in graphs*, we look for a network connecting terminals and hubs such that there is exactly one unique path between any two terminals and the total distance spanned by the optical fibers is minimum.

Optimization problems – Motivation

Steiner tree problem in graphs

Assume that a number of terminals (or clients) have to be connected by optical fibers. The terminals can be connected either directly or using a set of previously located hubs at fixed positions. The distances between each pair of points (be them terminals or hubs) are known beforehand. In the *Steiner tree problem in graphs*, we look for a network connecting terminals and hubs such that there is exactly one unique path between any two terminals and the total distance spanned by the optical fibers is minimum.

Maximum clique problem

Consider the global friendship network where pairs of people are considered to either be friends or not. In the *maximum clique problem*, we seek to determine the largest set of people for which each pair of people in the set are mutual friends.

Optimization problems – Motivation

Knapsack problem

Consider a hiker who needs to pack a knapsack with a number of items to take along on a hike. The knapsack has a maximum weight capacity. Each item has a given weight and some utility to the hiker. If all of the items fit in the knapsack, the hiker packs them and goes off. However, the entire set of items may not fit in the knapsack and the hiker will need to determine which items to take. The *knapsack problem* consists in finding a subset of items with maximum total utility, among all sets of items that fit in the knapsack.

Optimization problems – Motivation

Knapsack problem

Consider a hiker who needs to pack a knapsack with a number of items to take along on a hike. The knapsack has a maximum weight capacity. Each item has a given weight and some utility to the hiker. If all of the items fit in the knapsack, the hiker packs them and goes off. However, the entire set of items may not fit in the knapsack and the hiker will need to determine which items to take. The *knapsack problem* consists in finding a subset of items with maximum total utility, among all sets of items that fit in the knapsack.

Traveling salesman problem

Consider a traveling salesman who needs to visit all cities in a given sales territory. The salesman must begin and end the tour in a given city and visit each other city in the territory exactly once. Since each city must be visited only once, a tour can be represented by a circular permutation of the cities. Assuming the distances between each pair of cities are known beforehand, the objective of the *traveling salesman problem* is to determine a permutation of the cities that minimizes the total distance traveled.

Exact vs. approximate methods

Exact methods

An *exact method* for solving an optimization problem is one that is guaranteed to produce, in finite time, a global (or exact) optimum for this problem and a proof of its optimality, in case one exists, or otherwise show that no feasible solution exists.

Exact vs. approximate methods

Exact methods

An *exact method* for solving an optimization problem is one that is guaranteed to produce, in finite time, a global (or exact) optimum for this problem and a proof of its optimality, in case one exists, or otherwise show that no feasible solution exists.

Examples: cutting planes, dynamic programming, backtracking, and branch-and-bound.

Some of these paradigms can be viewed as tree search procedures, in the sense that they start from a feasible solution (which corresponds to the root of the tree) and carry out the search for the optimal solution by generating and scanning the nodes of a subtree of the solution space (whose nodes correspond to problem solutions).

Exact vs. approximate methods

Exact methods

An *exact method* for solving an optimization problem is one that is guaranteed to produce, in finite time, a global (or exact) optimum for this problem and a proof of its optimality, in case one exists, or otherwise show that no feasible solution exists.

Examples: cutting planes, dynamic programming, backtracking, and branch-and-bound.

Some of these paradigms can be viewed as tree search procedures, in the sense that they start from a feasible solution (which corresponds to the root of the tree) and carry out the search for the optimal solution by generating and scanning the nodes of a subtree of the solution space (whose nodes correspond to problem solutions).

Complexity theory shows that efficient exact algorithms are unknown (and unlikely to exist) for a broad class of (*NP*-hard) optimization problems, often referred to as *intractable*. Even though the size of the problems that can be solved to optimality (exactly) has been always increasing due to algorithmic and technological developments, there are problems (or instances) that cannot be solved by exact methods. Other approaches are needed to tackle such hard and large optimization problems.

Exact vs. approximate methods

Approximate methods (heuristics)

Opposed to exact methods, *approximate methods* are those that provide feasible solutions that, however, are not necessarily optimal.

Approximate methods usually run faster than exact methods. As a consequence, they are capable of handling larger problem instances than are exact methods. However, they will possibly find lower-quality solutions.

Exact vs. approximate methods

Approximate methods (heuristics)

Opposed to exact methods, *approximate methods* are those that provide feasible solutions that, however, are not necessarily optimal.

Approximate methods usually run faster than exact methods. As a consequence, they are capable of handling larger problem instances than are exact methods. However, they will possibly find lower-quality solutions.

Relevant work on heuristics or approximate algorithms for combinatorial optimization problems can be traced back to the origins of the field of Artificial Intelligence in the 1960s, with the development and applications of A^* search.

Exact vs. approximate methods

Approximate methods (heuristics)

Opposed to exact methods, *approximate methods* are those that provide feasible solutions that, however, are not necessarily optimal.

Approximate methods usually run faster than exact methods. As a consequence, they are capable of handling larger problem instances than are exact methods. However, they will possibly find lower-quality solutions.

Relevant work on heuristics or approximate algorithms for combinatorial optimization problems can be traced back to the origins of the field of Artificial Intelligence in the 1960s, with the development and applications of A^* search.

Constructive heuristics build a feasible solution from scratch and are often based on greedy algorithms.

Exact vs. approximate methods

Approximate methods (heuristics)

Opposed to exact methods, *approximate methods* are those that provide feasible solutions that, however, are not necessarily optimal.

Approximate methods usually run faster than exact methods. As a consequence, they are capable of handling larger problem instances than are exact methods. However, they will possibly find lower-quality solutions.

Relevant work on heuristics or approximate algorithms for combinatorial optimization problems can be traced back to the origins of the field of Artificial Intelligence in the 1960s, with the development and applications of A^* search.

Constructive heuristics build a feasible solution from scratch and are often based on greedy algorithms.

Local search methods start from a feasible solution and improve it by successive small modifications until a solution that cannot be further improved is encountered. Although they often provide high-quality solutions whose values are close to those of optimal values, they can become prematurely trapped in low-quality solutions.

Exact vs. approximate methods

Metaheuristics

Metaheuristics are general high-level procedures that coordinate simple heuristics and rules to find high-quality solutions to computationally difficult optimization problems.

Metaheuristics are based on distinct paradigms and offer different mechanisms to go beyond the first solution obtained that cannot be improved by local search.

They are among the most effective solution strategies for solving combinatorial optimization problems in practice and very frequently produce much better solutions than those obtained by the simple heuristics and rules they coordinate.

Concluding remarks

The material in this talk is taken from

- Chapter 1 – Introduction

of our book, *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures* (Resende & Ribeiro, 2016).