Combinatorial Optimization – Tối Ưu Tổ Hợp

Nguyễn Quản Bá Hồng*

Ngày 12 tháng 8 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- .
  PDF: URL: .pdf.
  TEX: URL: .tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Basic Combinatorial Optimization

1.1 [KV18]. Bernhard Korte, Jens Vygen. Combinatorial Optimization: Algorithms & Combinatorics. 6e

- Preface to 6e. Sect. 7.4 is devoted to shallow-light trees. Sect. 14.6 contains recent 2-factor approximation algorithm for submodule function maximization. Sect. 17.5 discusses Nemhauser-Ullmann algorithm & smoothed analysis. In Sect. 20.3, present $(\ln 4 + \epsilon)$-factor approximation algorithm for Steiner tree problem. Sect. 207 contains VPN theorem. There are also small additions, e.g., on integrality ratio in Sect. 5.1 & kernelization in Sect. 15.7.
- Preface to 5e. When preparing 1e of this book, > 10 years ago, tried to accomplish 2 objectives: should be useful as an advanced graduate textbook but also as a reference work for research. With each new edition, have to decide how book can be improved further. Of course, less & less possible to describe growing area comprehensively.

  If included everything that we like, book would grow beyond a single volume. Since book is used for many courses, now even sometimes at undergraduate level, thought: adding some classical material might be more useful than including a selection of latest results.

  In this edition, added a proof of Cayley's formula, more details on blocking flows, new faster b-matching separation algorithm, an approximation scheme for multidimensional knapsack, & results containing multicommodity max-flow min-cut ratio & sparest cut problem. There are further small improvements in numerous places & > 60 new exercises. Of course, also updated refs to point to most recent results & corrected some minor errors that were discovered.

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

- Preface to 4e. Again, have revised, updated, & significantly extended it for 4e. Have added some classical material that may have been missed so far, in particular on linear programming, network simplex algorithm, & max-cut problem. Have also added a number of new exercises & up-to-date references. Hope: these changes serve to make our book an even better basis for teaching & research. At http://www.or.uni-bonn.de/~vygen/co.html, continue to maintain updated information about this book.

- Preface to 3e. Most significant feature is a completely new chap on facility location. No constant-factor approximation algorithms were known for this important class of NP-hard problems until 8 years ago. Today there are several interesting & very different techniques that lead to good approximation guarantees, which makes this area particularly appealing also for teaching. In fact, chap has arisen from a special course on facility location.

  Many of other chaps have also been extended significantly. New material includes Fibonacci heaps, Fujishige's new maximum flow algorithm, flows over time, Schrijver's algorithm for submodular function minimization, & Robins-Zelikovsky Steiner tree approximation algorithm. Several proofs have been streamlined, & many new exercises & references have been added.

- Preface to 1e. Combinatorial optimization is 1 of youngest & most active areas of discrete mathematics & is probably its driving force today. It became a subject in its own right about 50 years ago. This book describes most important ideas, theoretical results, & algorithms in combinatorial optimization. Have conceived it as an advanced graduate text which can also be used as an up-to-date reference work for current research. Book includes essential fundamentals of graph theory, linear & integer optimization as well as very recent ones. Emphasis is on theoretical results & algorithms with provably good performance. Applications & heuristics are mentioned only occasionally.

  – Tối ưu hóa tổ hợp là một trong những lĩnh vực trẻ nhất & năng động nhất của toán học rời rạc & có lẽ là động lực thúc đẩy của nó ngày nay. Nó đã trở thành một môn học độc lập khoảng 50 năm trước. Cuốn sách này mô tả hầu hết các ý tưởng, kết quả lý thuyết, & thuật toán quan trọng nhất trong tối ưu hóa tổ hợp. Chúng tôi đã xây dựng nó như một giáo trình sau đại học nâng cao, đồng thời có thể được sử dụng làm tài liệu tham khảo cập nhật cho các nghiên cứu hiện tại. Sách bao gồm những kiến thức cơ bản thiết yếu về lý thuyết đồ thị, tối ưu hóa tuyến tính & số nguyên cũng như những kiến thức mới nhất. Trọng tâm là các kết quả lý thuyết & thuật toán có hiệu suất tốt đã được chứng minh. Các ứng dụng & phương pháp heuristic chỉ được đề cập đôi khi.

  Combinatorial optimization has its roots in combinatorics, operations research, & theoretical computer science. A main motivation: thousands of real-life problems can be formulated as abstract combinatorial optimization problems. Focus on detailed study of classical problems which occur in many different contexts, together with underlying theory.

  – Tối ưu hóa tổ hợp bắt nguồn từ toán học tổ hợp, nghiên cứu vận hành và khoa học máy tính lý thuyết. Động lực chính: hàng ngàn bài toán thực tế có thể được xây dựng thành các bài toán tối ưu hóa tổ hợp trừu tượng. Tập trung nghiên cứu chi tiết các bài toán cổ điển xảy ra trong nhiều bối cảnh khác nhau, cùng với lý thuyết nền tảng.

  Most combinatorial optimization problems can be formulated naturally in terms of graphs & as (integer) linear programs. Therefore this book starts, after an introduction, by reviewing basic graph theory & providing those results in linear & integer programming which are most relevant for combinatorial optimization.

  – Hầu hết các bài toán tối ưu hóa tổ hợp đều có thể được xây dựng một cách tự nhiên dưới dạng đồ thị & như các chương trình tuyến tính (số nguyên). Do đó, sau phần giới thiệu, cuốn sách này bắt đầu bằng việc xem xét lại lý thuyết đồ thị cơ bản & cung cấp những kết quả trong quy hoạch tuyến tính & số nguyên có liên quan nhất đến tối ưu hóa tổ hợp.

  Next classical topics in combinatorial optimization are studied: minimum spanning trees, shortest paths, network flows, matchings, & matroids. Most of problems discussed in Chaps. 6–14 have polynomial-time ("efficient") algorithms, while most of problems studied in Chaps. 15–21 are NP-hard, i.e., a polynomial-time algorithm is unlikely to exist. In many cases, one can at least find approximation algorithms that have a certain performance guarantee. Also mention some other strategies for coping with such "hard" problems.

  – Các chủ đề kinh điển tiếp theo trong tối ưu hóa tổ hợp sẽ được nghiên cứu: cây khung nhỏ nhất, đường đi ngắn nhất, luồng mạng, phép so khớp, & matroid. Hầu hết các bài toán được thảo luận trong Chương 6-14 đều có thuật toán thời gian đa thức ("hiệu quả"), trong khi hầu hết các bài toán được nghiên cứu trong Chương 15-21 đều là NP-khó, tức là, thuật toán thời gian đa thức khó có thể tồn tại. Trong nhiều trường hợp, ít nhất ta có thể tìm thấy các thuật toán xấp xỉ có một số đảm bảo hiệu suất nhất định. Ngoài ra, cũng đề cập đến một số chiến lược khác để giải quyết những bài toán "khó" như vậy.

  This book goes beyond scope of a normal textbook on combinatorial optimization in various aspects. E.g., cover equivalence of optimization & separation (for full-dimensional polytopes), $O(n^3)$-implementations of matching algorithms based on ear decompositions, Turing machines, perfect graph theorem, MAXSNP-hardness, Karmarkar-Karp algorithm for bin packing, recent approximation algorithms for multicommodity flows, survivable network design, & Euclidean traveling salesman problem. All results are accompanied by detailed proofs.

  – Cuốn sách này vượt ra ngoài phạm vi của một cuốn sách giáo khoa thông thường về tối ưu hóa tổ hợp ở nhiều khía cạnh. Ví dụ, bao gồm tính tương đương của tối ưu hóa & phân tách (cho đa diện toàn chiều), triển khai $O(n^3)$ của các thuật toán ghép dựa trên phân tích ear, máy Turing, định lý đồ thị hoàn hảo, độ khó MAXSNP, thuật toán Karmarkar-Karp cho đóng gói bin, các thuật toán xấp xỉ gần đây cho luồng đa hàng hóa, thiết kế mạng sống sót, & Bài toán người bán hàng du lịch Euclid. Tất cả các kết quả đều kèm theo chứng minh chi tiết.

  Of course, no book on combinatorial optimization can be absolutely comprehensive. Examples of topics mentioned only briefly or do not cover at all are tree decompositions, separators, submodular flows, path matchings, delta-matroids, matroid parity

problem, location & scheduling problems, nonlinear problems, semidefinite programming, average-case analysis of algorithms, advanced data structures, parallel & randomized algorithms, & theory of probabilistically checkable proofs (cite PCP theorem without proof).

– Tất nhiên, không có cuốn sách nào về tối ưu hóa tổ hợp có thể hoàn toàn đầy đủ. Ví dụ về các chủ đề chỉ được đề cập ngắn gọn hoặc không đề cập đến bao gồm phân tích cây, bộ tách, luồng dưới mô-đun, khớp đường đi, delta-matroid, bài toán chẵn lẻ matroid, bài toán vị trí & lập lịch, bài toán phi tuyến tính, lập trình bán xác định, phân tích trường hợp trung bình của thuật toán, cấu trúc dữ liệu nâng cao, thuật toán song song & ngẫu nhiên, & lý thuyết về chứng minh kiểm tra xác suất (trích dẫn định lý PCP mà không cần chứng minh).

This book arose from several courses on combinatorial optimization & from special classes on topics like polyhedral combinatorics or approximation algorithms. Thus material for basic & advanced courses can be selected from this book.

– Cuốn sách này được hình thành từ một số khóa học về tối ưu hóa tổ hợp & từ các lớp học chuyên biệt về các chủ đề như tổ hợp đa diện hoặc thuật toán xấp xỉ. Do đó, tài liệu cho các khóa học cơ bản & nâng cao có thể được chọn từ cuốn sách này.

- 1. Introduction. Start with 2 examples. A company has a machine which drills holes into printed circuit boards. Since it produces many of these boards it wants machine to complete 1 board as fast as possible. Cannot optimize drilling time but can try to minimize time machine needs to move from 1 point to another. Usually drilling machines can move in 2 directions: table moves horizontally while drilling arm moves vertically. Since both movements can be done simultaneously, time needed to adjust machine from 1 position to another is proportional to maximum of horizontal & vertical distance. This is called l_∞-distance. (Older machines can only move either horizontally or vertically at a time; in this case adjusting time is proportional to l_1-distance, sum of horizontal & vertical distance.)

  – Bắt đầu với 2 ví dụ. Một công ty có một máy khoan lỗ trên bảng mạch in. Vì sản xuất nhiều bảng mạch in, công ty muốn máy hoàn thành 1 bảng mạch in càng nhanh càng tốt. Không thể tối ưu hóa thời gian khoan nhưng có thể cố gắng giảm thiểu thời gian máy cần di chuyển từ điểm này sang điểm khác. Thông thường, máy khoan có thể di chuyển theo 2 hướng: bàn máy di chuyển theo chiều ngang trong khi tay khoan di chuyển theo chiều dọc. Vì cả hai chuyển động có thể được thực hiện đồng thời, thời gian cần thiết để điều chỉnh máy từ vị trí này sang vị trí khác tỷ lệ thuận với khoảng cách tối đa theo chiều ngang & chiều dọc. Khoảng cách này được gọi là l_∞-distance. (Các máy cũ hơn chỉ có thể di chuyển theo chiều ngang hoặc chiều dọc tại một thời điểm; trong trường hợp này, thời gian điều chỉnh tỷ lệ thuận với l_1-distance, tổng của khoảng cách theo chiều ngang & chiều dọc.)

  An optimum drilling path is given by an ordering of hole positions $p_1, \ldots, p_n$ s.t. $\sum_{i=1}^{n-1} d(p_i, p_{i+1})$ is minimum, where d is l_∞-distance: for 2 points $p = (x, y), p' = (x', y')$ in plane, write $d(p, p') = \max\{|x - x'|, |y - y'|\}$. An order of holes can be represented by a permutation, i.e., a bijection $\pi : [n] \to [n]$. Which permutation is best of course depends on hole positions; for each list of hole positions we have a different problem instance. Say: 1 instance of our problem is a list of points in plane, i.e., coordinates of holes to be drilled. Then problem can be stated formally as follows:

  – Đường khoan tối ưu được đưa ra bằng cách sắp xếp các vị trí lỗ $p_1, \ldots, p_n$ s.t. $\sum_{i=1}^{n-1} d(p_i, p_{i+1})$ là nhỏ nhất, trong đó d là khoảng cách l_∞: với 2 điểm $p = (x, y), p' = (x', y')$ trong mặt phẳng, hãy viết $d(p, p') = \max\{|x - x'|, |y - y'|\}$. Thứ tự các lỗ có thể được biểu diễn bằng một phép hoán vị, tức là một song ánh $\pi : [n] \to [n]$. Tất nhiên, phép hoán vị nào là tốt nhất phụ thuộc vào vị trí lỗ; với mỗi danh sách các vị trí lỗ, chúng ta có một trường hợp bài toán khác nhau. Giả sử: 1 trường hợp bài toán của chúng ta là một danh sách các điểm trong mặt phẳng, tức là tọa độ của các lỗ cần khoan. Khi đó, bài toán có thể được phát biểu một cách hình thức như sau:

  Problem 1 (Drilling problem)**.**

  Instance. *A set of points $p_1, \ldots, p_n \in \mathbb{R}^2$.*

  Task. *Find a permutation $\pi : [n] \to [n]$ s.t. $\sum_{i=1}^{n-1} d(p_{\pi(i)}, p_{\pi(i+1)})$ is minimum.*

  2nd example: Have a set of jobs to be done, each having a specified processing time. Each job can be done by a subset of employees, & assume: all employees who can do a job are equally efficient. Several employees can contribute to same job at same time, & 1 employee can contribute to several jobs (but not at same time). Objective: get all jobs done as early as possible.

  – VD2: Có một tập hợp các công việc cần thực hiện, mỗi công việc có thời gian xử lý cụ thể. Mỗi công việc có thể được thực hiện bởi một nhóm nhân viên, & giả định: tất cả nhân viên có thể thực hiện một công việc đều có hiệu suất như nhau. Nhiều nhân viên có thể cùng lúc thực hiện một công việc, & 1 nhân viên có thể thực hiện nhiều công việc (nhưng không cùng lúc). Mục tiêu: hoàn thành tất cả các công việc càng sớm càng tốt.

  In this model it suffices to prescribe for each employee how long he or she should work on which job. Order in which employees carry out their jobs is not important, since time when all jobs are done obviously depends only on maximum total working time we have assigned to 1 employee. Hence have to solve problem:

  – Trong mô hình này, chỉ cần quy định cho mỗi nhân viên thời gian làm việc cho từng công việc là đủ. Thứ tự nhân viên thực hiện công việc không quan trọng, vì thời gian hoàn thành tất cả công việc rõ ràng chỉ phụ thuộc vào tổng thời gian làm việc tối đa mà chúng ta đã phân bổ cho 1 nhân viên. Do đó, chúng ta phải giải quyết vấn đề:

  Problem 2 (Job assignment problem)**.**

  Instance. *A set of numbers $t_1, \ldots, t_n \in \mathbb{R}_+$ (processing times for n jobs), a number $m \in \mathbb{N}$ of employees, & a nonempty subset $S_i \subset [m]$ of employees for each job $i \in [n]$.*

Task. *Find numbers* $x_{ij} \in \mathbb{R}_+$, $\forall i \in [n], j \in S_i$ *s.t.* $\sum_{j \in S_i} x_{ij} = t_i$ *for* $i \in [n]$ *&* $\max_{j \in [m]} \sum_{i: j \in S_i} x_{ij}$ *is minimum.*

These are 2 typical problems arising in combinatorial optimization. How to model a practical problem as an abstract combinatorial optimization problem is not described in this book; indeed there is no general recipe for this task. besides giving a precise formulation of input & desired output, often important to ignore irrelevant components (e.g., drilling time which cannot be optimized or order in which employees carry out their jobs).

– Đây là 2 bài toán điển hình phát sinh trong tối ưu hóa tổ hợp. Sách này không mô tả cách mô hình hóa một bài toán thực tế thành một bài toán tối ưu hóa tổ hợp trừu tượng; thực tế là không có công thức chung nào cho nhiệm vụ này. Bên cạnh việc đưa ra công thức chính xác về đầu vào & đầu ra mong muốn, việc bỏ qua các thành phần không liên quan (ví dụ: thời gian khoan không thể tối ưu hóa hoặc thứ tự thực hiện công việc của nhân viên) thường rất quan trọng.

Of course not interested in a solution to a particular drilling problem or job assignment problem in some company, but rather looking for a way how to solve all problems of these types. 1st consider Drilling prob.

– Tất nhiên là không quan tâm đến giải pháp cho một vấn đề khoan cụ thể hay vấn đề phân công công việc ở một công ty nào đó, mà là tìm cách giải quyết mọi vấn đề thuộc loại này. Trước tiên hãy xem xét vấn đề khoan.

- 1.1. Enumeration. How can a solution to Drilling problem look like? There are infinitely many instances (finite sets of points in plane), so cannot list an optimum permutation for each instance. Instead, what we look for is an algorithm which, given an instance, computes an optimum solution. Such an algorithm exists: Given a set of n points, just try all possible $n!$ orders, & for each compute l_∞-length of corresponding path.

  – Giải pháp cho bài toán Khoan có thể trông như thế nào? Có vô số trường hợp (tập hợp hữu hạn các điểm trên mặt phẳng), nên không thể liệt kê một hoán vị tối ưu cho từng trường hợp. Thay vào đó, chúng ta cần tìm một thuật toán, cho trước một trường hợp, tính toán một giải pháp tối ưu. Một thuật toán như vậy tồn tại: Cho một tập hợp n điểm, chỉ cần thử tất cả các bậc $n!$ có thể, & với mỗi trường hợp, hãy tính độ dài l_∞ của đường đi tương ứng.

  There are different ways of formulating an algorithm, differing mostly in level of detail & formal language they use. Certainly would not accept following as an algorithm: "Given a set of n points, find an optimum path & output it." Not specified at all how to find optimum solution. Above suggestion to enumerate all possible $n!$ orders is more useful, but still is not clear how to enumerate all orders. Here is 1 possible way: Enumerate all n-tuples of numbers $1, \dots, n$, i.e., all n^n vectors of $[n]^n$. This can be done similarly to counting: start with $(1, 1, \dots, 1), (1, \dots, 1, 2)$ up to $(1, \dots, 1, n)$ then switch to $(1, \dots, 1, 2, 1)$, & so on. At each step, increment last entry unless it is already n, in which case go back to last entry that is smaller than n, increment it & set all subsequent entries to 1. This techniques is sometimes called backtracking. Order in which vectors of $[n]^n$ are enumerated is called lexicographical order:

  – Có nhiều cách khác nhau để xây dựng một thuật toán, chủ yếu khác nhau ở mức độ chi tiết & ngôn ngữ hình thức mà họ sử dụng. Chắc chắn sẽ không chấp nhận sau đây là một thuật toán: "Cho một tập hợp n điểm, tìm một đường đi tối ưu & xuất ra nó." Không chỉ rõ cách tìm giải pháp tối ưu. Đề xuất ở trên để liệt kê tất cả các thứ tự $n!$ có thể hữu ích hơn, nhưng vẫn không rõ cách liệt kê tất cả các thứ tự. Sau đây là 1 cách có thể: Liệt kê tất cả các bộ n số $1, \dots, n$, tức là tất cả các vectơ n^n của $[n]^n$. Điều này có thể được thực hiện tương tự như đếm: bắt đầu với $(1, 1, \dots, 1), (1, \dots, 1, 2)$ đến $(1, \dots, 1, n)$ sau đó chuyển sang $(1, \dots, 1, 2, 1)$, & vân vân. Ở mỗi bước, tăng phần tử cuối cùng trừ khi nó đã là n, trong trường hợp đó, quay lại phần tử cuối cùng nhỏ hơn n, tăng phần tử đó & đặt tất cả các phần tử tiếp theo thành 1. Kỹ thuật này đôi khi được gọi là quay lui. Thứ tự liệt kê các vectơ của $[n]^n$ được gọi là thứ tự từ điển:

  Definition 1. *Let* $x, y \in \mathbb{R}^n$*: 2 vectors. Say: a vector* x *is* lexicographically smaller *than* y *if there exists an index* $j \in [n]$ *s.t.* $x_i = y_i$ *for* $i \in [j-1]$ *&* $x_j < y_j$*.*

  – Cho $x, y \in \mathbb{R}^n$*: 2 vectơ. Giả sử: một vectơ* x nhỏ hơn về mặt từ điển *so với* y *nếu tồn tại một chỉ số* $j \in [n]$ *s.t.* $x_i = y_i$ *với* $i \in [j-1]$ *&* $x_j < y_j$*.*

  Knowing how to enumerate all vectors of $[n]^n$, can simply check for each vector whether its entries are pairwise distinct &, if so, whether path represented by this vector is shorter than best path encountered so far.

  – Biết cách liệt kê tất cả các vectơ của $[n]^n$, có thể dễ dàng kiểm tra từng vectơ xem các mục của nó có phân biệt từng cặp hay không &, nếu có, liệu đường dẫn được biểu diễn bởi vectơ này có ngắn hơn đường dẫn tốt nhất đã gặp cho đến nay hay không.

  Since this algorithm enumerates n^n vectors it will take at least n^n steps (in fact, even more). This is not best possible. There are only $n!$ permutations of $[n]$, & $n!$ is significantly smaller than n^n. (By Stirling's formula $n! \approx \sqrt{2\pi n}\frac{n^n}{e^n}$ (Stirling [1730])). Shall show how to enumerate all paths in approximately $n^2 n!$ steps. Consider following algorithm which enumerates all permutations in lexicographical order:

  Problem 3 (Path enumeration algorithm)**.**

  Input. *A natural number* $n \geq 3$*. A set* $\{p_1, \dots, p_n\}$ *of points in plane.*

  Output. *A permutation* $\pi^* : [n] \to [n]$ *with* $\text{cost}(\pi^*) \coloneqq \sum_{i=1}^{n-1} d(p_{\pi^*(i)}, p_{\pi^*(i+1)})$ *minimum.*

  p. 3+++

- 2. Graphs.
- 3. Linear Programming.

- 4. Linear Programming Algorithms.
- 5. Integer Programming.
- 6. Spanning Trees & Arborescences.
- 7. Shortest Paths.
- 8. Network Flows.
- 9. Minimum Cost Flows.
- 10. Maximum Matchings.
- 11. Weighted Matching.
- 12. b-Matchings & T-Joins.
- 13. Matroids.
- 14. Generalizations of Matroids.
- 15. NP-Completeness.
- 16. Approximation Algorithms.
- 17. Knapsack Problem.
- 18. Bin-Packing.
- 19. Multicommodity Flows & Edge-Disjoint Paths.
- 20. Network Design Problems.
- 21. Traveling Salesman Problem.
- 22. Facility Location.

2 Wikipedia's

2.1 Wikipedia/combinatorial optimization

Combinatorial optimization is a subfield of mathematical optimization that consists of finding an optimal object from a finite set of objects, where set of feasible solutions is discrete or can be reduced to a discrete set. Typical combinatorial optimization problems are traveling salesman problem (TSP), minimum spanning tree problem (MST), & knapsack problem. In many such problems, e.g. ones previously mentioned, exhaustive search is not tractable, & so specialized algorithms that quickly rule out large parts of search space or approximation algorithms must be resorted to instead.

– Tối ưu hóa tổ hợp là một nhánh của tối ưu hóa toán học, bao gồm việc tìm kiếm một đối tượng tối ưu từ một tập hợp hữu hạn các đối tượng, trong đó tập hợp các nghiệm khả thi là rời rạc hoặc có thể được rút gọn thành một tập rời rạc. Các bài toán tối ưu hóa tổ hợp điển hình bao gồm bài toán người bán hàng du lịch (TSP), bài toán cây bao trùm nhỏ nhất (MST), và bài toán ba lô. Trong nhiều bài toán như vậy, ví dụ như các bài toán đã đề cập trước đó, tìm kiếm vét cạn không khả thi, do đó, các thuật toán chuyên biệt loại trừ nhanh chóng các phần lớn không gian tìm kiếm hoặc các thuật toán xấp xỉ phải được sử dụng thay thế.

Fig: A minimum spanning tree of a weighted planar graph. Finding a minimum spanning tree is a common problem involving combinatorial optimization.

Combinatorial optimization is related to operations research, algorithm theory, & computational complexity theory. It has important applications in several fields, including AI, ML, auction theory, software engineering, VLSI, applied mathematics & theoretical CS.

– Tối ưu hóa tổ hợp liên quan đến nghiên cứu hoạt động, lý thuyết thuật toán và lý thuyết độ phức tạp tính toán. Nó có những ứng dụng quan trọng trong nhiều lĩnh vực, bao gồm AI, ML, lý thuyết đấu giá, kỹ thuật phần mềm, VLSI, toán ứng dụng và khoa học máy tính lý thuyết.

2.1.1 Applications

Basic applications of combinatorial optimization include, but are not limited to: logistics, supply chain optimization, developing best airline network of spoles & destinations, deciding which taxis in a fleet to route to pick up fares, determining optimal way to deliver packages, allocating jobs to people optimally, designing water distribution networks, earth science problems (e.g. reservoir flow-rates).

– Các ứng dụng cơ bản của tối ưu hóa tổ hợp bao gồm nhưng không giới hạn ở: hậu cần, tối ưu hóa chuỗi cung ứng, phát triển mạng lưới hàng không tốt nhất về điểm đến & điểm đến, quyết định tuyến taxi nào trong đội bay để đón khách, xác định cách tối ưu để giao hàng, phân bổ công việc cho mọi người một cách tối ưu, thiết kế mạng lưới phân phối nước, các vấn đề khoa học về trái đất (ví dụ: lưu lượng dòng chảy của hồ chứa).

2.1.2 Methods

There is a large amount of literature on polynomial-time algorithms for certain special classes of discrete optimization. A considerable amount of it is unified by theory of linear programming. Some examples of combinatorial optimization problems that are covered by this framework are shortest paths & shortest-path trees, flows & circulations, spanning trees, matching, & matroid problems.

– Có rất nhiều tài liệu về thuật toán thời gian đa thức cho một số lớp tối ưu hóa rời rạc đặc biệt. Phần lớn trong số đó được thống nhất bởi lý thuyết quy hoạch tuyến tính. 1 số ví dụ về các bài toán tối ưu hóa tổ hợp được đề cập trong khuôn khổ này bao gồm đường đi ngắn nhất & cây đường đi ngắn nhất, luồng & tuần hoàn, cây khung, bài toán ghép nối & matroid.

For NP-complete discrete optimization problems, current research literature includes following topics:

- polynomial-time exactly solvable special cases of problem at hand (e.g., fixed-parameter tractable problems)
- algorithms that perform well on "random" instances (e.g., for TSP)
- approximation algorithms that run in polynomial time & find a solution that is close to optimal
- solving real-world instances that arise in practice & do not necessarily exhibit worst-case behavior of in NP-complete problems (e.g. real-world TSP instances with 10s of thousands of nodes).

– Đối với các bài toán tối ưu rời rạc NP-đầy đủ, các tài liệu nghiên cứu hiện tại bao gồm các chủ đề sau:

- các trường hợp đặc biệt có thể giải chính xác trong thời gian đa thức của bài toán đang xét (ví dụ: các bài toán có thể giải được với tham số cố định)
- các thuật toán hoạt động tốt trên các trường hợp "ngẫu nhiên" (ví dụ: đối với TSP)
- các thuật toán xấp xỉ chạy trong thời gian đa thức & tìm ra giải pháp gần tối ưu
- giải các trường hợp thực tế phát sinh trong thực tế & không nhất thiết thể hiện hành vi xấu nhất của các bài toán NP-đầy đủ (ví dụ: các trường hợp TSP thực tế với hàng chục nghìn nút).

Combinatorial optimization problems can be viewed as searching for best element of some set of discrete items; therefore, in principle, any sort of search algorithm or metaheuristic can be used to solve them. Widely applicable approaches include branch-&-bound (an exact algorithm which can be stopped at any point in time to serve as heuristic), branch-&-cut (uses linear optimization to generate bounds), dynamic programming (a recursive solution construction with limited search window) & tabu search (a greedy-type swapping algorithm). However, generic search algorithms are not guaranteed to find an optimal solution 1st, nor are they guaranteed to run quickly (in polynomial time). Since some discrete optimization problems are NP-complete, e.g. traveling salesman (decision) problem, this is expected unless P = NP.

– Các bài toán tối ưu hóa tổ hợp có thể được xem như việc tìm kiếm phần tử tốt nhất của một tập hợp các phần tử rời rạc; do đó, về nguyên tắc, bất kỳ loại thuật toán tìm kiếm hoặc siêu thuật toán nào cũng có thể được sử dụng để giải chúng. Các phương pháp được áp dụng rộng rãi bao gồm branch-&-bound (một thuật toán chính xác có thể dừng tại bất kỳ thời điểm nào để làm thuật toán tìm kiếm), branch-&-cut (sử dụng tối ưu hóa tuyến tính để tạo ra các giới hạn), lập trình động (một cấu trúc giải pháp đệ quy với cửa sổ tìm kiếm giới hạn) và tìm kiếm & tabu (một thuật toán hoán đổi kiểu tham lam). Tuy nhiên, các thuật toán tìm kiếm chung không đảm bảo sẽ tìm ra giải pháp tối ưu đầu tiên, cũng như không đảm bảo chúng chạy nhanh (trong thời gian đa thức). Vì một số bài toán tối ưu hóa rời rạc là NP-đầy đủ, ví dụ như bài toán người bán hàng du lịch (quyết định), điều này là bình thường trừ khi P = NP.

For each combinatorial optimization problem, there is a corresponding decision problem that asks whether there is a feasible solution for some particular measure m_0. E.g., if there is a graph G which contains vertices u, v, an optimization problem might be "find a path from u to v that uses fewest edges". This problem might have an answer of, say, 4. A A corresponding decision problem would be "is there a path from u to v that uses 10 or fewer edges?" This problem can be answered with a simple yes or no.

– Với mỗi bài toán tối ưu tổ hợp, có một bài toán quyết định tương ứng đặt ra câu hỏi liệu có một giải pháp khả thi cho một số đo m_0 cụ thể nào đó hay không. Ví dụ, nếu có một đồ thị G chứa các đỉnh u, v, một bài toán tối ưu hóa có thể là "tìm đường đi từ u đến v sử dụng ít cạnh nhất". Bài toán này có thể có đáp án là, chẳng hạn, 4. 1 bài toán quyết định tương ứng sẽ là "có đường đi từ u đến v sử dụng 10 cạnh trở xuống không?" Bài toán này có thể được trả lời đơn giản bằng có hoặc không.

Field of approximation algorithms deals with algorithms to find near-optimal solutions to hard problems. Usual decision version is then an inadequate definition of problem since it only specifies acceptable solutions. Even though could introduce suitable decision problems, problem is then more naturally characterized as an optimization problem.

– Lĩnh vực thuật toán xấp xỉ tập trung vào các thuật toán tìm kiếm các giải pháp gần tối ưu cho các bài toán khó. Phiên bản quyết định thông thường khi đó là một định nghĩa không đầy đủ về bài toán vì nó chỉ xác định các giải pháp chấp nhận được. Mặc dù có thể đưa ra các bài toán quyết định phù hợp, nhưng bài toán khi đó được mô tả một cách tự nhiên hơn là bài toán tối ưu hóa.

2.1.3 NP optimization problem

An *NP-optimization problem* (NPO) is a combinatorial optimization problem with following additional conditions. Note: below referred polynomials are functions of size of respective functions' inputs, not size of some implicit set of input instances.

- size of every feasible solution $y \in f(x)$ is polynomially bounded in size of given instance x,
- languages $\{x; x \in I\}$ & $\{(x, y); y \in f(x)\}$ can be recognized in polynomial time, &
- m is polynomial-time computable.

This implies: corresponding decision problem is in NP. In computer science, interesting optimization problems usually have above properties & are therefore NPO problems. A problem is additionally called a P-optimization (PO) problem, if there exists an algorithm which finds optimal solution in polynomial time. Often, when dealing with class NPO, one is interested in optimization problems for which decision versions are NP-complete. Note: hardness relations are always w.r.t. some reduction. Due to connection between approximation algorithms & computational optimization problems, reductions which preserve approximation in some respect are for this subject preferred than usual Turing & Karp reductions. An example of such a reduction would be L-reduction. For this reason, optimization problems with NP-complete decision versions are not necessarily called NPO-complete.

– Bài toán tối ưu hóa NP (NPO) là một bài toán tối ưu hóa tổ hợp với các điều kiện bổ sung sau. Lưu ý: các đa thức được đề cập dưới đây là các hàm phụ thuộc vào kích thước của các đầu vào tương ứng, chứ không phải kích thước của một tập hợp các thể hiện đầu vào ngầm định.

- Kích thước của mọi nghiệm khả thi $y \in f(x)$ bị chặn đa thức theo kích thước của thể hiện x đã cho,
- Các ngôn ngữ $\{x; x \in I\}$ & $\{(x, y); y \in f(x)\}$ có thể được nhận dạng trong thời gian đa thức, &
- m có thể tính toán được trong thời gian đa thức.

Điều này ngụ ý: bài toán quyết định tương ứng nằm trong NP. Trong khoa học máy tính, các bài toán tối ưu hóa thú vị thường có các tính chất trên & do đó là các bài toán NPO. 1 bài toán còn được gọi là bài toán tối ưu hóa P (PO) nếu tồn tại một thuật toán tìm ra nghiệm tối ưu trong thời gian đa thức. Thông thường, khi xử lý lớp NPO, người ta quan tâm đến các bài toán tối ưu hóa mà các phiên bản quyết định là NP-đầy đủ. Lưu ý: các quan hệ độ cứng luôn liên quan đến một số phép rút gọn. Do mối liên hệ giữa các thuật toán xấp xỉ và các bài toán tối ưu hóa tính toán, các phép rút gọn bảo toàn xấp xỉ ở một số khía cạnh được ưu tiên hơn so với các phép rút gọn Turing và Karp thông thường. 1 ví dụ về phép rút gọn như vậy là phép rút gọn L. Vì lý do này, các bài toán tối ưu hóa với các phiên bản quyết định NP-đầy đủ không nhất thiết được gọi là NPO-đầy đủ.

NPO is divided into following subclasses according to their approximability:

- NPO(I): Equals FPTAS. Contains Knapsack problem.
- NPO(II): Equals PTAS. Contains Makespan scheduling problem.
- NPO(III): Class of NPO problems that have polynomial-time algorithms which computes solution with a cost at most c times optimal cost (for minimization problems) or a cost at least $\frac{1}{c}$ of optimal cost (for maximization problems). In Hromkovič's book, excluded from this class are all NPO(II)-problems save if P = NP. Without exclusion, equals APX. Contains MAX-SAT & metric TSP.
- NPO(IV): Class of NPO problems with polynomial-time algorithms approximating optimal solution by a ratio that is polynomial in a logarithm of size of input. In Hromkovič's book, all NPO(III)-problems are excluded from this class unless P = NP. Contains set cover problem.
- NPO(V): Class of NPO problems with polynomial-time algorithms approximating optimal solution by a ratio bounded by some function on n. In Hromkovic's book, all NPO(IV)-problems are excluded from this class unless P = NP. Contains TSP & clique problem.

– NPO được chia thành các lớp con sau theo tính xấp xỉ của chúng:

- NPO(I): Bằng FPTAS. Chứa bài toán Knapsack.
- NPO(II): Bằng PTAS. Chứa bài toán lập lịch Makespan.
- NPO(III): Lớp các bài toán NPO có thuật toán thời gian đa thức tính toán nghiệm với chi phí tối đa là c lần chi phí tối ưu (đối với bài toán cực tiểu hóa) hoặc chi phí tối thiểu là $\frac{1}{c}$ chi phí tối ưu (đối với bài toán cực đại hóa). Trong sách của Hromkovič, tất cả các bài toán NPO(II) trừ khi P = NP đều bị loại trừ khỏi lớp này. Nếu không loại trừ, bằng APX. Chứa MAX-SAT & metric TSP.
- NPO(IV): Lớp các bài toán NPO với thuật toán thời gian đa thức tính toán nghiệm tối ưu theo tỷ lệ đa thức trong logarit kích thước đầu vào. Trong sách của Hromkovič, tất cả các bài toán NPO(III) đều bị loại khỏi lớp này trừ khi P = NP. Chứa bài toán phủ tập hợp.
- NPO(V): Lớp các bài toán NPO với thuật toán thời gian đa thức xấp xỉ nghiệm tối ưu theo tỷ lệ bị chặn bởi một hàm số nào đó trên n. Trong sách của Hromkovic, tất cả các bài toán NPO(IV) đều bị loại khỏi lớp này trừ khi P = NP. Chứa bài toán TSP & clique.

An NPO problem is called polynomially bounded (PB) if, for every instance x & for every solution $y \in f(x)$, measure $m(x, y)$ is bounded by a polynomial function of size of x. Class NPOPB is class NPO problems that are polynomially-bounded.

– 1 bài toán NPO được gọi là bị chặn đa thức (PB) nếu, với mọi trường hợp x & với mọi nghiệm $y \in f(x)$, độ đo $m(x, y)$ bị chặn bởi một hàm đa thức có độ lớn x. Lớp NPOPB là lớp các bài toán NPO bị chặn đa thức.

2.1.4 Specific problems

Assignment problem, bin packing problem, Chinese postman problem, closure problem, constraint satisfaction problem, cutting stock problem, dominating set problem, integer programming, job shop scheduling, knapsack problem, metric k-center/vertex k-center problem, minimum relevant variables in linear system, minimum spanning tree, nurse scheduling problem, ring star problem, set cover problem, talent scheduling, traveling salesman problem, vehicle rescheduling problem, vehicle routing problem, weapon target assignment problem.

– Bài toán gán, bài toán đóng thùng, bài toán người đưa thư Trung Quốc, bài toán đóng, bài toán thỏa mãn ràng buộc, bài toán cắt kho, bài toán tập hợp trội, lập trình số nguyên, lập lịch xưởng gia công, bài toán ba lô, bài toán tâm đỉnh k, biến liên quan tối thiểu trong hệ thống tuyến tính, cây bao trùm tối thiểu, bài toán lập lịch y tá, bài toán ngôi sao vòng, bài toán che phủ tập hợp, lập lịch tài năng, bài toán nhân viên bán hàng du lịch, bài toán lập lịch lại phương tiện, bài toán định tuyến phương tiện, bài toán chỉ định mục tiêu vũ khí.

3 Miscellaneous

Tài liệu

[KV18] Bernhard Korte and Jens Vygen. *Combinatorial optimization*. Vol. 21. Algorithms and Combinatorics. Theory and algorithms, Sixth edition of [MR1764207]. Springer, Berlin, 2018, pp. xxi+698. ISBN: 978-3-662-56038-9; 978-3-662-56039-6. DOI: 10.1007/978-3-662-56039-6. URL: https://doi.org/10.1007/978-3-662-56039-6.