The Algorithmics Column

by

Gerhard J Woeginger

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
`gwoegi@win.tue.nl`

K-Best Enumeration

David Eppstein*

Abstract

We survey k-best enumeration problems and the algorithms for solving them, including in particular the problems of finding the k shortest paths, k smallest spanning trees, and k best matchings in weighted graphs.

1 Introduction

Researchers in the design and analysis of algorithms have had much success in finding precise mathematical formulations of optimization problems, developing efficient algorithms for solving those problems, and proving worst-case bounds on the time complexity of these algorithms. A case in point is the problem of finding minimum spanning trees in graphs, studied since the 1920s [22] and finally shown in 1995 to be solvable in (randomized) linear time [114]. Other problems of a similar nature are the graph shortest path problem, famously solved by Dijkstra's algorithm [57], and the problem of finding minimum weight matchings in graphs, first solved in polynomial time by Edmonds [62] and later significantly improved [82, 83].

However, many real-world problems are only approximately modeled by these mathematical formulations. The most desirable solution may depend on quality criteria that are more complicated than a simple sum of edge weights, it may be determined by data that is not yet available or subject to unpredictable changes, or its choice may involve political or social processes that are not well-modeled by mathematical values. In such a situation, it is desirable for an algorithm to output more than one candidate solution. Once such a list of candidates is generated, one can apply more sophisticated quality criteria, wait for data to become available to choose among them, or present them all to human decision-makers. On the other hand, the exponential growth of the solution space for many combinatorial

*Computer Science Department, University of California, Irvine; Irvine, CA, USA. This material is based upon work supported by the National Science Foundation under Grant CCF-1228639 and by the Office of Naval Research under Grant No. N00014-08-1-1015. A significantly shorter version of this material appears in the Springer *Encyclopedia of Algorithms*, 2014.

optimization problems would make it infeasible to list all possible candidate solutions; instead, some filtering is needed.

A common approach to such problems breaks the decision process into two stages. In the first stage, the problem is modeled mathematically in the standard way (e.g. for the problems above, by weighting the edges of a graph and seeking a solution with a small sum of edge weights). However, rather than finding a single optimal solution, an algorithm is used to find the *k best solutions*, for a given parameter value k: that is, the algorithm finds a set of k different solutions that have a better solution quality than any solution not in the set. Then, these candidates are passed on to the second stage of the decision process, where they may be evaluated and compared using more complicated evaluation criteria than sums of weights, used as a set of candidates to switch among dynamically based on updated data, or passed to human decision-makers. A familiar example of this occurs in car navigation systems, which can typically be programmed to present drivers with multiple alternative routes between the current location of the car and the target destination.

In this review we survey algorithms for generating sets of the k best solutions, for several optimization problems, as well as the applications of these algorithms.

2 General Principles

If the single best solution to an optimization problem can be found by an algorithm whose running time is polynomial in the input size, then it is typical for the k best solutions to be found in time polynomial in both the input size and k. There are two general techniques that can be used to achieve this, based on optimal substructures and solution space partitioning. In turn, these methods rely on fast algorithms for the *selection problem* in certain sets of structured values.

2.1 Selection

Underlying many k-best algorithms is the problem of *selection*, finding the k smallest values from a larger set of values [21]. If there are n values in the set, then selection may be solved in time $O(n + k)$, for instance by *quickselect*, an algorithm that chooses a random pivot value from the set, partitions the rest of the set into subsets that are less than, equal to, or greater than the pivot, and then recurses into one of these subsets [131].

However, in k-best problems, we do not wish to generate all solutions before finding the best of them: we wish to avoid the $O(n)$ part of this $O(n+k)$ time bound. Often, the solution values among which we are selecting the k best have some additional structure that allows finding the k best without examining all solutions.

For instance, suppose that the space of solutions can be represented as the vertex set of an (implicitly defined) edge-weighted graph, that the degrees of the vertices in this graph are bounded, and that the quality of any particular solution equals the length of the shortest path in this graph from some designated starting vertex to the solution vertex. In such a case, Dijkstra's algorithm can be used to recover the k smallest of these values in time $O(k \log k)$, independent of the size of the graph, without requiring that the whole graph be explored. Other structures that allow the k best solutions to be found more efficiently than enumeration of the whole solution set include representations of the solution space as the set of sums of pairs of values drawn from two sorted sets [76, 111], or as the elements of an (implicitly defined) matrix whose rows and columns are sorted [77]. Again, for such structures, the k smallest values can be found efficiently without examining the whole matrix or the whole set of pair sums.

A general framework that can be used to describe many of these structured selection problems is the problem of selection in a d-ary heap. A min-heap is a rooted tree, with values associated with its nodes, such that each non-root node has a value that is at least as large as its parent. In order to avoid having to construct all solutions before performing a selection algorithm, and in order to handle situations in which this tree has infinitely many nodes, we will assume that our selection algorithm is given as input an implicit representation of such a heap. That is, its input consists of a binary representation of the root node of the heap together with pointers to two subroutines that take a single node as input: one subroutine to list the children of any node in the heap, and another subroutine to find the value of any node in the heap. Based on this information, the problem is to find the k nodes with the smallest values. For instance, the problem of selection in a sorted array can be represented in this way by a heap in which each node represents an array cell, the root is the upper left corner of the array, and the parent of each node is either the cell above it (if it is not in the first row of the array) or to its left (if it is in the first row). An implicit min-heap represented in this way is a special case of a graph (where the weight of an edge is the difference in values between a node and its parent), so Dijkstra's algorithm can find the k best solution values in such a structure in time $O(k \log k)$. A more sophisticated algorithm of Frederickson for searching an implicit heap improves this time bound to $O(k)$ [74].

2.2 Optimal Substructures

The optimal substructure technique is most applicable to problems whose optimization version is solved by a dynamic programming algorithm. In the dynamic programming technique, one identifies a family of polynomially many subproblems of the same type as the original problem, and a recurrence relation by which the value of any subproblem can be calculated in terms of the values of smaller subproblems. The value of the whole problem can then be found by iterating through all of the subproblems in order by their sizes, using the recurrence to calculate and store the value of each subproblem as a combination of previously-computed values. To apply this algorithmic framework to a k-best problem, one stores the k best solutions to each subproblem (rather than a single best solution), and modifies the recurrence to compute these k best solutions as combinations of the k best solutions of smaller subproblems. The k-best algorithm then iterates through the subproblems in order by size, using this recurrence to compute the k best values of each subproblem.

As an example, the problem of finding a minimum weight triangulation of a point set (NP-hard for arbitrary planar point sets [141]) can be solved in polynomial time for points in convex position, by a dynamic program in which the subproblems are the subsets of the points that can be formed by intersecting them with a half-plane [84,118]. The optimal solution for such a problem may be found by choosing one edge of the convex hull, testing all triangles that could be based on that edge, and for each such triangle adding the weights of the optimal solutions of the two subproblems that the triangle splits the problem into. For n input points, there are $O(n^2)$ subproblems, the solution to each of which involves examining $O(n)$ triangles, so the total time is $O(n^3)$. To modify this dynamic programming algorithm to find the k smallest-weight triangulations, we may store the k smallest weight solutions for each subproblem (in sorted order). To compute the value of a problem, we again choose one convex hull edge and test all triangles that include that edge. However, a single triangle may produce $O(k^2)$ solutions (combinations of the k values stored in each of the two subproblems on either side of the triangle), so implemented naively this method would take $O(k^2 n)$ time per subproblem to sort through all these solutions and select the k best of them, giving an $O(k^2 n^3)$ overall time bound. To speed this up, one can replace the computation for each triangle by an algorithm for finding the k smallest values among the pairwise sums from two sets of k sorted values, which may be solved in $O(k)$ time [76]. This speedup leads to an $O(kn^3)$ time bound overall.

2.3 Solution-Space Partitions

An alternative general approach to k-best problems involves partitioning the space of solutions into smaller subspaces defined by subproblems of the original problem. For instance, in a problem where the solutions may be described as sets of edges in a graph, a subproblem may be specified by requiring some edges to be included in any solution while removing some other edges from the graph, forcing them to be excluded.

If it is possible to find not just the best solution to a given optimization problem, but also the second best, then this may be used to partition the space of solutions into a collection of subproblems that have the structure of a binary heap. Each node of this heap represents a single subproblem (represented e.g. by its sets of forbidden and required edges) and has a value equal to the second-best solution within that subspace. The root of the heap is the space of all solutions, i.e., a subproblem where we have not yet made any restrictions. For a given node X of this heap, the best and second-best solution must differ from each other, for instance by including an edge e in one of these two solutions and excluding e from the other solution. The two children of node X may then be formed as the subproblems where e is added respectively to the set of forbidden or required edges. One of these two subproblems includes the best but not the second best solution, and the other includes the second best but not the best solution, so both subproblems have second-best solution values that differ from the value for the whole space. This hierarchical partition of the solution space organizes all the solutions except the best one into an implicit binary heap, allowing the selection algorithm of Frederickson [74] to be used to find the k best solutions while examining only $O(k)$ of the subproblems from this structure. This, the time for this procedure is $O(k)$ times the time to find a single second-best solution.

An alternative partitioning technique uses only the best solution in each subproblem, rather than also requiring the second best solution. However, in exchange for this easier computation in each subproblem, it produces a heap-ordered tree of solution values in which the degree of each tree node is higher. Suppose, for instance, that the k-best problem that we are trying to solve has an input in the form of a graph, and that its solutions can be represented by sequences of edges in this graph; for instance, this is true of the k-shortest paths and k-best spanning trees problems. Suppose that the best solution to the problem is represented by the sequence of edges $e_1, e_2, \ldots$. Then, from this solution, we form a collection of subproblems, one for each of these edges, where the ith subproblem consists of the solutions that are forced to include all the edges in the sequence up to e_{i-1} but that exclude e_i. Every solution to the overall problem, other than the best solution, belongs to exactly one of these subproblems: the one defined by the first difference between the given solution and the best solution. Recursively subdividing each of these subproblems into sub-subproblems, etc., produces a heap-ordered tree whose degree equals the maximum number of edges. Again, applying a selection algorithm in a heap allows the k best solutions to be found. However, because of the higher degree of the min-heap in this technique, the number of subproblems that will be examined is $O(ks)$ and the overall time is $O(ks)$ times the time for finding a single best solution, where s is the maximum number of edges in a single solution.

These two partitioning techniques are reviewed in more detail by Hamacher and Queyranne [95]. They attribute the binary heap partition method to a k-best spanning tree algorithm of Gabow [81]. The multiway partition based only on the best solution comes from a k-best matching algorithm by Murty [142] and its generalization by Lawler [124].

3 Shortest Paths

Probably the most important and heavily studied of the k-best optimization problems is the problem of finding k shortest paths [2, 4, 9, 10, 14, 15, 23, 31, 39, 41, 43, 66, 70–72, 80, 85–88, 100, 102, 106, 108, 109, 121–125, 127, 133, 138–140, 152–154, 156, 157, 159, 168–170, 172, 173, 182–185, 193].

The k-shortest paths problem includes as special cases finding the k best solutions to problems such as biological sequence alignment [27, 143, 166, 167, 181] or the (0, 1)-knapsack problem [187], whose dynamic programming solutions can be expressed as shortest path problems in an associated graph. Many problems of hypothesis generation in natural language processing and speech recognition can also be formulated as k-best optimization problems [20, 35, 38, 45–48, 54, 68, 99, 119, 120, 161, 162, 174]. The Viterbi decoding technique for Markov models, commonly used to model these problems, can also be formulated as a search for a path in an associated graph, with a vertex for each pair of a time step and a Markov state, and the k-best beam search technique used for multiple hypothesis generation in these problems can be interpreted as a special case of a k-shortest-path algorithm. This method can also be used to combine different techniques for language and speech recognition, by using one technique to generate hypotheses and the other technique to rescore them [146].

The many other applications of the k shortest paths problem include reconstruction of metabolic pathways [6] and gene regulation networks [171], motion tracking [17], message routing in communications networks [11, 12, 15, 130, 180], listing close genealogical relationships in highly intermarried pedigrees [66], power line placement [44], vehicle and transportation routing [90, 110, 137, 186], building evacuation planning [113], timing analysis of circuits [8, 189], task scheduling [59, 101], and communications and transportation network design [24, 26, 58, 61, 129], as well as in subroutines for other combinatorial optimization problems [19, 33, 51, 53, 79, 105]

In the most basic version of the problem, the input is a weighted directed graph, with two designated source and destination vertices s and t, and a number k. The goal is to find k different walks (paths allowing repeated vertices) from s to t, with the minimum possible weights. An algorithm by Eppstein [66] achieves optimal asymptotic time complexity: $O(m + n \log n + k)$, constant time per path after a preprocessing stage with the running time of Dijkstra's algorithm for a single shortest path. Eppstein's algorithm begins by computing a tree T of shortest paths to t, and (following Hoffman and Pavley [100]) it represents each of its output paths by the sequences of *detours* that these paths make: edges that do not belong to T. The length of the path is then the shortest path distance from s to t, plus the sum of the lengths added by each detour. For each vertex v in the graph, Eppstein's algorithm constructs a collection of the detours whose starting vertex is on the path in T from v to t; this collection is represented as a binary heap, using persistent data structure techniques [60] to allow these collections to share substructures with each other to save preprocessing time and storage. The algorithm uses these collections to partition the space of solutions into a collection of subproblems that themselves have the structure of a bounded-degree heap. Each subproblem consists of the paths that start with a given sequence of detours, and that use at least one more detour from a given binary heap of detours. The optimal solution of such a subproblem is the one whose final detour is at the root of the heap, and it has three subproblems with worse solutions as children: two in which the root detour is not used and instead the path uses at least one detour from a child of the root in the binary heap of detours, and one where the root detour is used but is not the last detour, and the next detour comes from the binary heap associated with the endpoint of the root detour.

In a graph with cycles, the k shortest paths may consist of as few as one simple path, together with one or more repetitions of a short cycle starting and ending at one of the vertices of the path. Loops of this type are generally not desired, and the problem is particularly critical when the input is an undirected graph, as converting it to a directed graph by replacing each undirected edge by two directed edge will create many potential loops. Beginning with Clarke, Krikorian, and Rausen [49] researchers have developed algorithms that instead seek the k shortest simple (or loopless) paths from s to t in a network [18, 30, 40, 98, 104, 132, 151, 158, 178, 188]. Yen's algorithm [188] still remains the one with the best asymptotic time performance. It is based on best-solution partitioning and Dijkstra's algorithm; the number of edges in a single solution is at most $n - 1$, and the time to find a solution using the Fibonacci-heap variant of Dijkstra's algorithm is $O(m + n \log n)$ (in a graph with m edges and n vertices), so following the general form for best-solution partitioning, the time for this method is $O(kn(m+n \log n)$. A more recent algorithm of Hershberger, Maxal, and Suri [98] is often faster, but is based on a heuristic that can sometimes fail, causing it to fall back to Yen's algorithm and achieve the same performance bound. In the case of undirected graphs, it is possible to find the k shortest simple paths faster, in time $O(k(m + n \log n))$ [89, 115, 117].

Minieka [138] and Fox [72] considered an all-pairs variant of the k-shortest-paths problem in which the goal is to find a separate set of k paths for each pair of vertices in the graph. For this problem, Eppstein's algorithm requires only n copies of the preprocessing stage (one for each destination vertex), after which each path takes constant time to find, so the total time is $O(mn + n^2 \log n + kn^2)$. The shortest path tree in a graph is a tree connecting a given source vertex to all other vertices, minimizing the sum of the path costs to the other vertices. The k-best version of this problem, seeking the k best trees according to this quality measure, has also been studied [165].

There has also been research on finding a given number of paths between two given terminals that are completely disjoint from each other and minimize a sum of weights [32, 144]. This problem can be solved as a special case of the minimum-cost flow problem; it has a significantly different flavor from k-best enumeration problems, as the choice of one path affects the others and the number of paths that can be selected is much smaller.

4 Spanning Trees

A 1977 paper of Gabow [81] introduced both the problem of finding the k minimum-weight spanning trees of an edge-weighted graph, and the technique of finding a binary hierarchical subdivision of the space of solutions, which he used to solve the problem. In any graph, the best and second-best spanning trees differ only by one edge swap (the removal of one edge from a tree and its replacement by a different edge that reconnects the two subtrees formed by the removal), a property that simplifies the search for a second-best tree as needed for Gabow's partitioning technique. For similar reasons, when k is smaller than the numbers n and m of vertices or edges in the input graph, the graph may be simplified by finding a single minimum spanning tree, computing the best swap that each edge of the graph participates in, removing the edges that are not in the tree and do not participate in the k best swaps, and contracting the edges that are in the tree but do not participate in the k best swaps. For this reason, factors of n and m in the running time of any k-best spanning tree algorithm may be replaced by k when this replacement would be an improvement [64].

The fastest known algorithms for the k best spanning trees problem are based on Gabow's partitioning technique, together with dynamic graph data structures that keep track of the best swap in a network as that network undergoes a sequence of edge insertion and deletion operations [67, 73, 75]. To use this technique, one initializes a fully-persistent best-swap data structure (one in which each update creates a new version of the structure without modifying the existing versions, and in which updates may be applied to any version [60]) and associates its initial version with the root of the subproblem tree. Then, whenever an algorithm for selecting the k best nodes of the subproblem tree generates a new node (a subproblem formed by including or excluding an edge from the allowed solutions) the parent node's version of the data structure is updated (by either increasing or decreasing the weight of the edge to force it to be included or excluded in all solutions) and the updated version of the data structure is associated with the child node. In this way, the data structure can be used to quickly find the second-best solution for each of the subproblems explored by the algorithm. Based on this method, the k best spanning trees of a graph with n vertices and m edges can be found (in an implicit representation based on sequences of swaps rather than explicitly listing all edges in each tree) in time $O(\mathrm{MST}(m,n)+k\min(n,k)^{1/2})$ where $\mathrm{MST}(m,n)$ denotes the time for finding a single minimum spanning tree [67]. If randomized algorithms are considered, the minimum spanning tree problem can be solved in linear time [114], so the $\mathrm{MST}(m,n)$ term can be replaced by $m+n$. This technique may also be used to find the k best spanning trees of a set of n points in the Euclidean plane, in time $O(n\log n\log k + k\min(n,k)^{1/2})$ [65, 67]; these trees may include pairs of edges that cross each other, but it is also possible to find the k best non-crossing planar spanning trees efficiently [134].

The kth smallest distinct weight of a spanning tree may be obtained by a sequence of at most $k-1$ swaps from any minimum spanning tree, allowing these weights to be generated in polynomial time when k is constant [112, 136]. However, when k is an input variable, finding the kth smallest distinct spanning tree weight remains NP-hard [136].

The problem of finding the k best spanning trees has been applied to NP-hard multicriterion optimization problems for spanning trees [50, 96, 176], to point process intensity estimation [97], to the analysis of metabolic pathways [7], to image segmentation [177] and classification [63], to the reconstruction of pedigrees from genetic data [52], to the parsing of natural-language text [1], and to the analysis of electronic circuits [190]. This problem is a special case of finding the k best bases of a matroid, which has also been studied [25, 34, 95, 126]. For additional research on the k smallest spanning trees problem see [16, 25, 116, 128, 175, 179].

5 Other Problems

After paths and spanning trees, probably the next most commonly studied k-best enumeration problem concerns matchings. The problem of finding the k minimum-weight perfect matchings in an edge-weighted graph was introduced by Murty [142]. A later algorithm by Chegireddy and Hamacher [36] solves the problem in time $O(kn^3)$ (where n is the number of vertices in the graph) using the technique of building a binary partition of the solution space. The k-best matchings have been used to find matchings with additional side constraints [13] or with multivariate optimization criteria [163]. They have also been applied for message routing in parallel computing systems [42]. For additional work on this problem see [56, 135, 147, 150].

In natural language processing applications, an important generalization of the k shortest paths problem involves finding the k best parse trees of a context-free grammar [35, 103, 107, 148, 149, 192]. Other problems whose k-best solutions have been studied include the Chinese postman problem [160], the traveling salesman problem [155], the k best spanning arborescences in a directed network [28], the matroid intersection problem [29, 191], binary search trees and Huffman coding [5], chess strategies [3], the k best integer flows [92, 93, 164], the k smallest cuts in a network [91, 94, 95], and, in probabilistic reasoning, the k best solutions to a graphical model [55, 69, 78, 145].

For many NP-hard optimization problems, where even finding a single best solution is difficult, an approach that has proven very successful is *parameterized complexity*, in which one finds an integer parameter describing the input instance or its solution that is often much smaller than the input size, and designs algorithms whose running time is a fixed polynomial of the input size multiplied by a non-polynomial function of the parameter value. Chen et al. [37] extend this paradigm to k-best problems, showing that, for instance, many NP-hard k-best problems can be solved in polynomial time per solution for graphs of bounded treewidth.

References

[1] Ž. Agić. K-best spanning tree dependency parsing with verb valency lexicon reranking. *Proc. COLING 2012*, 2012.

[2] H. Aljazzar and S. Leue. K^*: a heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence* 175(18):2129–2154, 2011, doi:10.1016/j.artint.2011.07.003.

[3] I. Althöfer. On the K-best mode in computer chess: measuring the similarity of move proposals. *ICCA J.* 20(3):152–165, September 1997.

[4] K. N. Androutsopoulos and K. G. Zografos. Solving the k-shortest path problem with time windows in a time varying network. *Oper. Res. Lett.* 36(6):692–695, 2008, doi:10.1016/j.orl.2008.07.003.

[5] S. Anily and R. Hassin. Ranking the best binary trees. *SIAM J. Comput.* 18(5):882–892, 1989, doi:10.1137/0218060.

[6] M. Arita. Metabolic reconstruction using shortest paths. *Simulation Practice and Theory* 8(1–2):109–125, 2000, doi:10.1016/S0928-4869(00)00006-9.

[7] M. Arita, K. Asai, and T. Nishioka. Graph modeling of metabolism. *Proc. 4th Int. Conf. Computational Molecular Biology (RECOMB '00)*, 2000.

[8] T. Asano and S. Sato. Long path enumeration algorithms for timing verification on large digital systems. *Graph theory with applications to algorithms and computer science (Kalamazoo, Mich., 1984)*, pp. 25–35. Wiley, New York, Wiley-Intersci. Publ., 1985.

[9] J. A. Azevedo, M. E. O. Santos Costa, J. J. E. R. Silvestre Madeira, and E. Q. Vieira Martins. An algorithm for the ranking of shortest paths. *Eur. J. Oper. Res.* 69(1):97–106, 1993, doi:10.1016/0377-2217(93)90095-5.

[10] J. A. Azevedo, J. J. E. R. Silvestre Madeira, E. Q. Vieira Martins, and F. M. A. Pires. A computational improvement for a shortest paths ranking algorithm. *Eur. J. Oper. Res.* 73(1):188–191, 1994, doi:10.1016/0377-2217(94)90162-7.

[11] K. Bala, T. E. Stern, and K. Bala. Algorithms for routing in a linear lightwave network. *Proc. 10th Joint. Conf. IEEE Computer & Communications Socs. (INFOCOM 1991)*, vol. 1, pp. 1–9, 1991, doi:10.1109/INFCOM.1991.147477.

[12] K. Bala, T. E. Stern, D. Simchi-Levi, and K. Bala. Routing in a linear lightwave network. *ACM/IEEE Trans. on Networking* 3(4):459–469, 1995, doi:10.1109/90.413220.

[13] M. O. Ball, U. Derigs, C. Hilbrand, and A. Metz. Matching problems with generalized upper bound side constraints. *Networks* 20(6):703–721, 1990, doi:10.1002/net.3230200602.

[14] H. Beilner. Ein Algorithmus zur Ermittlung *k*-bester Kantenfolgen zwischen allen Ecken eines Graphen [An algorithm for determining *k*-best sequences of edges between all vertices of a graph]. *Computing* 10(3):205–220, 1972, doi:10.1007/BF02316908.

[15] R. Bellman and R. Kalaba. On *k*th best policies. *J. SIAM* 8(4):582–588, 1960, doi:10.1137/0108044.

[16] P. C. Berbert, L. J. R. Freitas Filho, T. A. Almeida, M. B. Carvalho, and A. Yamakami. Artificial immune system to find a set of *k*-spanning trees with low costs and distinct topologies. *Artificial Immune Systems: 6th International Conference, ICARIS 2007, Santos, Brazil, August 26-29, 2007, Proceedings*, pp. 395–406. Springer, Lecture Notes in Computer Science 4628, 2007, doi:10.1007/978-3-540-73922-7_34.

[17] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using *K*-shortest paths optimization. *IEEE Trans. Pattern Analysis and Machine Intelligence* 33(9):1806–1819, 2011, doi:10.1109/TPAMI.2011.21.

[18] A. Bernstein. A nearly optimal algorithm for approximating replacement paths and *K* shortest simple paths in general graphs. *Proc. 21st ACM-SIAM Symp. Discrete Algorithms (SODA '10)*, pp. 742–755. Society for Industrial and Applied Mathematics, 2010.

[19] S. Bespamyatnikh and A. Kelarev. Algorithms for shortest paths and *d*-cycle problems. *J. Discrete Algorithms* 1(1):1–9, 2003, doi:10.1016/S1570-8667(03)00002-9.

[20] M. Betz and H. Hild. Language models for a spelled letter recognizer. *Proc. Internat. Conf. Acoustics, Speech, and Signal Processing*, vol. 1, pp. 856–859. IEEE, 1995, doi:10.1109/ICASSP.1995.479829.

[21] M. Blum, V. Pratt, R. E. Tarjan, R. W. Floyd, and R. L. Rivest. Time bounds for selection. *J. Comput. System Sci.* 7:448–461, 1973, doi:10.1016/S0022-0000(73)80033-9.

[22] O. Borůvka. O jistém problému minimálním (About a certain minimal problem). *Práce mor. přírodověd. spol. v Brně III* 3:37–58, 1926.

[23] A. W. Brander and M. C. Sinclair. A comparative study of k-shortest path algorithms. *Performance Engineering of Computer and Telecommunications Systems: Proc. UKPEW'95, Liverpool John Moores University, UK, 5–6 September 1995*, pp. 370–379, 1996, doi:10.1007/978-1-4471-1007-1_25.

[24] G. Bruno, G. Ghiani, and G. Improta. A multi-modal approach to the location of a rapid transit line. *Eur. J. Oper. Res.* 104(2):321–332, 1998, doi:10.1016/S0377-2217(97)00187-2.

[25] R. N. Burns and C. E. Haff. A ranking problem in graphs. *Proc. 5th Southeastern Conference on Combinatorics, Graph Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1974)*, pp. 461–470. Utilitas Math., Congressus Numerantium 10, 1974.

[26] M. T. Busche, C. M. Lockhart, and C. Olszewski. Dynamic K-shortest path (DKSP) facility restoration algorithm. *GLOBECOM, Communications: The Global Bridge*, vol. 1, pp. 536–542. IEEE, 1994, doi:10.1109/GLOCOM.1994.513577.

[27] T. H. Byers and M. S. Waterman. Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming. *Oper. Res.* 32(6):1381–1384, 1984, doi:10.1287/opre.32.6.1381.

[28] P. M. Camerini, L. Fratta, and F. Maffioli. The K best spanning arborescences of a network. *Networks* 10(2):91–110, 1980, doi:10.1002/net.3230100202.

[29] P. M. Camerini and H. W. Hamacher. Intersection of two matroids: (condensed) border graphs and ranking. *SIAM J. Discrete Math.* 2(1):16–27, 1989, doi:10.1137/0402002.

[30] W. M. Carlyle and R. K. Wood. Near-shortest and K-shortest simple paths. *Networks* 46(2):98–109, 2005, doi:10.1002/net.20077.

[31] P. Carraresi and C. Sodini. A binary enumeration tree to find K shortest paths. *VII. symposium on operations research, Sections 1–3 (St. Gallen, 1982)*, pp. 177–188. Athenäum/Hain/Hanstein, Methods Oper. Res. 45, 1983.

[32] D. Castanon. Efficient algorithms for finding the K best paths through a trellis. *IEEE Transactions on Aerospace and Electronic Systems* 26(2):405–410, 1990, doi:10.1109/7.53448.

[33] H.-J. Chang and U.-T. Lai. Empirical comparison between two k-shortest path methods for the generalized assignment problem. *J. Inform. Optim. Sci.* 19(2):153–171, 1998, doi:10.1080/02522667.1998.10699369.

[34] B. Chaourar. On the Kth best base of a matroid. *Oper. Res. Lett.* 36(2):239–242, 2008, doi:10.1016/j.orl.2007.05.007.

[35] E. Charniak and M. Johnson. Coarse-to-fine N-best parsing and MaxEnt discriminative reranking. *Proc. 43rd Meeting of Association for Computational Linguistics (ACL '05)*, pp. 173–180, 2005, doi:10.3115/1219840.1219862.

[36] C. R. Chegireddy and H. W. Hamacher. Algorithms for finding K-best perfect matchings. *Discrete Appl. Math.* 18(2):155–165, 1987, doi:10.1016/0166-218X(87)90017-5.

[37] J. Chen, I. A. Kanj, J. Meng, G. Xia, and F. Zhang. Parameterized top-K algorithms. *Theoret. Comput. Sci.* 470:105–119, 2013, doi:10.1016/j.tcs.2012.10.052.

[38] J.-K. Chen and F. K. Soong. An N-best candidates-based discriminative training for speech recognition applications. *Trans. Speech & Audio Processing* 2(1, part 2):206–216, 1994, doi:10.1109/89.260363.

[39] Y. L. Chen. An algorithm for finding the k quickest paths in a network. *Comput. Oper. Res.* 20(1):59–65, 1993, doi:10.1016/0305-0548(93)90096-2.

[40] Y. L. Chen. Finding the k quickest simple paths in a network. *Inform. Process. Lett.* 50(2):89–92, 1994, doi:10.1016/0020-0190(94)00008-5.

[41] Y.-L. Chen and H.-H. Yang. Finding the first K shortest paths in a time-window network. *Comput. Oper. Res.* 31(4):499–513, 2004, doi:10.1016/S0305-0548(02)00230-7.

[42] W.-K. Chiang and R.-J. Chen. Block-switch networks: a cost-effective class of interconnection networks. *Computer Systems Science and Engineering* 12(3):175–185, 1997, `http://ir.lib.nctu.edu.tw/handle/987654321/39344`.

[43] E. I. Chong, S. Maddila, and S. Morley. On finding single-source single-destination k shortest paths. *J. Comput. Inf.* 1(2):40–47, 1995.

[44] F. Choobineh and T. Burgman. Transmission line route selection: An application of K-shortest paths and goal programming. *Trans. Power Apparatus and Systems* PAS-103(11):3253–3259, 1984, doi:10.1109/TPAS.1984.318562.

[45] W. Chou, C.-H. Lee, and B.-H. Juang. Minimum error rate training based on N-best string models. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93)*, vol. 2, pp. 652–655, 1993, doi:10.1109/ICASSP.1993.319394.

[46] W. Chou, C.-H. Lee, B.-H. Juang, and F. K. Soong. A minimum error rate pattern recognition approach to speech recognition. *Int. J. Pattern Recognition & Artificial Intelligence* 8(1):5–31, 1994, doi:10.1142/S0218001494000024.

[47] W. Chou, T. Matsuoka, B.-H. Juang, and C.-H. Lee. An algorithm of high resolution and efficient multiple string hypothesization for continuous speech recognition using inter-word models. *Proc. Internat. Conf. Acoustics, Speech, and Signal Processing*, vol. 2, pp. II/153–156. IEEE, 1994.

[48] Y.-L. Chow and R. Schwartz. The N-best algorithm: an efficient search procedure for finding top N sentence hypotheses. *Proc. DARPA Speech & Natural Language Worksh.*, pp. 199–202, 1989.

[49] S. Clarke, A. Krikorian, and J. Rausen. Computing the N best loopless paths in a network. *J. SIAM* 11(4):1096–1102, 1963, doi:10.1137/0111081.

[50] J. C. N. Clímaco, M. E. Captivo, and M. M. B. Pascoal. On the bicriterion—minimal cost/minimal label—spanning tree problem. *Eur. J. Oper. Res.* 204(2):199–205, 2010, doi:10.1016/j.ejor.2009.10.013.

[51] J. M. Coutinho-Rodrigues, J. C. N. Clímaco, and J. R. Current. An interactive bi-objective shortest path approach: searching for unsupported nondominated solutions. *Comput. Oper. Res.* 26(8):789–798, 1999, doi:10.1016/S0305-0548(98)00094-X.

[52] R. G. Cowell. A simple greedy algorithm for reconstructing pedigrees. *Theor. Population Biol.* 83:55–63, 2013, doi:10.1016/j.tpb.2012.11.002.

[53] J. R. Current, C. S. ReVelle, and J. L. Cohon. The median shortest path problem: a multiobjective approach to analyze cost vs. accessibility in the design of transportation networks. *Transportation Science* 21(3):188–197, 1987, doi:10.1287/trsc.21.3.188.

[54] J.-C. Dai and H.-J. Lee. Parsing with tag information in a probabilistic generalized LR parser. *Proc. Internat. Conf. Chinese Computing*, pp. 33–39. Nat. Univ. Singapore, 1994.

[55] R. Dechter, N. Flerova, and R. Marinescu. Search algorithms for m best solutions for graphical models. *Proc. 26th AAAI Conf. Artificial Intelligence*, pp. 1895–1901, 2012, `http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/viewFile/5167/5349`.

[56] U. Derigs and A. Metz. On the construction of the set of K-best matchings and their use in solving constrained matching problems. *Combinatorial Optimization (Ankara, 1990)*, pp. 209–223. Springer, NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci. 82, 1992, doi:10.1007/978-3-642-77489-8_11.

[57] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271, 1959, doi:10.1007/BF01386390.

[58] J. M. B. Diogo, J. C. N. Clímaco, P. M. N. Nordeste, and J. M. F. Craveirinha. Dynamic planning model for urban telephone networks and its applications. *IEE Proceedings I (Communications, Speech and Vision)* 136(4):283–290, 1989.

[59] B. M. Dodin. Determining the K most critical paths in PERT networks. *Oper. Res.* 32(4):859–877, 1984, doi:10.1287/opre.32.4.859.

[60] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making data structures persistent. *J. Comput. System Sci.* 38(1):86–124, 1989, doi:10.1016/0022-0000(89)90034-2.

[61] D. A. Dunn, W. D. Grover, and M. H. MacGregor. Comparison of k-shortest paths and maximum flow routing for network facility restoration. *IEEE J. Selected Areas in Communications* 12(1):88–99, 1994, doi:10.1109/49.265708.

[62] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *J. Res. Nat. Bur. Standards Sect. B* 69B:125–130, 1965.

[63] S. El Fkihi, M. Daoudi, and D. Aboutajdine. The mixture of K-optimal-spanning-trees based probability approximation: Application to skin detection. *Image and Vision Computing* 26(12):1574–1590, 2008, doi:10.1016/j.imavis.2008.02.003.

[64] D. Eppstein. Finding the k smallest spanning trees. *BIT* 32(2):237–248, 1992, doi:10.1007/BF01994879.

[65] D. Eppstein. Tree-weighted neighbors and geometric k smallest spanning trees. *Internat. J. Comput. Geom. Appl.* 4(2):229–238, 1994, doi:10.1142/S0218195994000136.

[66] D. Eppstein. Finding the k shortest paths. *SIAM J. Comput.* 28(2):652–673, 1998, doi:10.1137/S0097539795290477.

[67] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification—a technique for speeding up dynamic graph algorithms. *J. ACM* 44(5):669–696, 1997, doi:10.1145/265910.265914.

[68] K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs. *Proc. 23rd Internat. Conf. Computational Linguistics (COLING '10)*, pp. 322–330. Association for Computational Linguistics, 2010.

[69] N. Flerova, E. Rollon, and R. Dechter. Bucket and mini-bucket schemes for m best solutions over graphical models. *Proc. 2nd Internat. Conf. Graph Structures for Knowledge Representation and Reasoning (GKR'11)*, pp. 91–118. Springer, Lecture Notes in Computer Science 7205, 2012, doi:10.1007/978-3-642-29449-5_4.

[70] B. L. Fox. Calculating kth shortest paths. *INFOR–Canad. J. Op. Res. & Inf. Proc.* 11:66–70, 1973.

[71] B. L. Fox. k-th shortest paths and applications to the probabilistic networks. *Bull. Operations Research Soc. of America* 23:B263, 1975.

[72] B. L. Fox. More on kth shortest paths. *Commun. ACM* 18(5):279, 1975, doi:10.1145/360762.360803.

[73] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.* 14(4):781–798, 1985, doi:10.1137/0214055.

[74] G. N. Frederickson. An optimal algorithm for selection in a min-heap. *Inform. and Comput.* 104(2):197–214, 1993, doi:10.1006/inco.1993.1030.

[75] G. N. Frederickson. Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees. *SIAM J. Comput.* 26(2):484–538, 1997, doi:10.1137/S0097539792226825.

[76] G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Comput. System Sci.* 24(2):197–208, 1982, doi:10.1016/0022-0000(82)90048-4.

[77] G. N. Frederickson and D. B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM J. Comput.* 13(1):14–30, 1984, doi:10.1137/0213002.

[78] A. Fromer, M. and Globerson. An LP view of the M-best MAP problem. *Advances in Neural Information Processing Systems* 22:567–575, 2009, `http://papers.nips.cc/paper/3745-an-lp-view-of-the-m-best-map-problem`.

[79] L. Fu and L. R. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Res. B* 32(7):499–516, 1998, doi:10.1016/S0191-2615(98)00016-2.

[80] A. Fuchs. 1–n Verfahren zur Bestimmung von k-kürzesten Wegen [1–n method for determining k-shortest paths]. *Operations Research Proceedings 1984 (St. Gallen, 1984)*, pp. 421–428. Springer, 1985, doi:10.1007/978-3-642-70457-4_108.

[81] H. N. Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM J. Comput.* 6(1):139–150, 1977, doi:10.1137/0206011.

[82] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM* 38(4):815–853, 1991, doi:10.1145/115234.115366.

[83] Z. Galil, S. Micali, and H. N. Gabow. An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput.* 15(1):120–130, 1986, doi:10.1137/0215009.

[84] P. N. Gilbert. New results in planar triangulations. Master's thesis, Coordinated Science Lab., Univ. of Illinois, Urbana, 1979.

[85] N. Gravin and N. Chen. A note on k-shortest paths problem. *J. Graph Theory* 67(1):34–37, 2011, doi:10.1002/jgt.20510.

[86] F. Guerriero and R. Musmanno. Parallel asynchronous algorithms for the K shortest paths problem. *J. Optim. Theory Appl.* 104(1):91–108, 2000, doi:10.1023/A:1004676705907.

[87] F. Guerriero, R. Musmanno, V. Lacagnina, and A. Pecorella. A class of label-correcting methods for the K shortest paths problem. *Oper. Res.* 49(3):423–429, 2001, doi:10.1287/opre.49.3.423.11217.

[88] M. Günther, J. Schuster, and M. Siegle. Symbolic calculation of K-shortest paths and related measures with the stochastic process algebra tool CASPA. *Proc. 1st Worksh. Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (DYADEM-FTS '10)*, pp. 13–18. ACM, 2010, doi:10.1145/1772630.1772635.

[89] E. Hadjiconstantinou and N. Christofides. An efficient implementation of an algorithm for finding K shortest simple paths. *Networks* 34(2):88–101, 1999, doi:10.1002/(SICI)1097-0037(199909)34:2<88::AID-NET2>3.3.CO;2-T.

[90] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Ann. Oper. Res.* 61:21–43, 1995, doi:10.1007/BF02098280.

[91] H. W. Hamacher. An $(K \cdot n^4)$ algorithm for finding the k best cuts in a network. *Oper. Res. Lett.* 1(5):186–189, 1982, doi:10.1016/0167-6377(82)90037-2.

[92] H. W. Hamacher. A note on K best network flows. *Ann. Oper. Res.* 57:65–72, 1995, doi:10.1007/BF02099691.

[93] H. W. Hamacher and C. Hüsselmann. Ranking approach to max-ordering combinatorial optimization and network flows. *Beiträge zur angewandten Analysis und Informatik*, pp. 97–111. Shaker, 1994.

[94] H. W. Hamacher, J.-C. Picard, and M. Queyranne. On finding the K best cuts in a network. *Oper. Res. Lett.* 2(6):303–305, 1984, doi:10.1016/0167-6377(84)90083-X.

[95] H. W. Hamacher and M. Queyranne. K best solutions to combinatorial optimization problems. *Ann. Oper. Res.* 4(1–4):123–143, 1985, doi:10.1007/BF02022039.

[96] H. W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Ann. Oper. Res.* 52:209–230, 1994, doi:10.1007/BF02032304.

[97] A. O. Hero and O. Michel. Robust estimation of point process intensity features using k-minimal spanning trees. *Proc. Internat. Symp. Information Theory*, p. 74. IEEE, 1997, doi:10.1109/ISIT.1997.612989.

[98] J. Hershberger, M. Maxel, and S. Suri. Finding the k shortest simple paths: a new algorithm and its implementation. *ACM Trans. Algorithms* 3(4):A45, 2007, doi:10.1145/1290672.1290682.

[99] T. Hisamitsu and Y. Nitta. A generalized algorithm for Japanese morphological analysis and a comparative evaluation of some heuristics. *Systems & Computers in Japan* 26(1):73–87, 1995, doi:10.1002/scj.4690260107.

[100] W. Hoffman and R. Pavley. A method for the solution of the Nth best path problem. *J. ACM* 6(4):506–514, 1959, doi:10.1145/320998.321004.

[101] G. J. Horne. Finding the K least cost paths in an acyclic activity network. *J. Oper. Res. Soc.* 31(5):443–448, 1980, `http://www.jstor.org/stable/2581195`.

[102] T.-Y. Hu, H. S. Mahmassani, and A. K. Ziliaskopoulos. Implementation and testing of a K-shortest path algorithm in a vector and parallel processing environment. *Proc. Conf. Computer Science & Operations Research: New Developments & their Interfaces*, 1992.

[103] L. Huang and D. Chiang. Better k-best parsing. *Proc. 9th Internat. Worksh. Parsing Technology (IWPT '05)*, pp. 53–64. Association for Computational Linguistics, 2005, `http://www.aclweb.org/anthology/W05-1506.pdf`.

[104] H. Ishii. A new method finding the Kth best path in a graph. *J. Oper. Res. Soc. Japan* 21(4):469–476, 1978.

[105] Z. Jia and P. Varaiya. Heuristic methods for delay constrained least cost routing using k-shortest-paths. *IEEE Trans. Automat. Control* 51(4):707–712, 2006, doi:10.1109/TAC.2006.872827.

[106] V. M. Jiménez and A. Marzal. Computing the K shortest paths: A new algorithm and an experimental comparison. *Algorithm Engineering: 3rd International Workshop, WAE'99 London, UK, July 19–21, 1999, Proceedings*, pp. 15–29. Springer, Lecture Notes in Computer Science 1668, 1999, doi:10.1007/3-540-48318-7_4.

[107] V. M. Jiménez and A. Marzal. Computation of the N best parse trees for weighted and stochastic context-free grammars. *Proc. Joint IAPR Internat. Workshops on Advances in Pattern Recognition*, pp. 183–192. Springer, Lecture Notes in Computer Science 1876, 2000, doi:10.1007/3-540-44522-6_19.

[108] V. M. Jiménez and A. Marzal. A lazy version of Eppstein's K shortest paths algorithm. *Experimental and Efficient Algorithms: Second International Workshop, WEA 2003, Ascona, Switzerland, May 26–28, 2003, Proceedings*, pp. 179–190. Springer, Lecture Notes in Computer Science 2647, 2003, doi:10.1007/3-540-44867-5_14.

[109] L.-M. Jin and S.-P. Chan. An electrical method for finding suboptimal routes. *Int. Symp. Circuits and Systems*, vol. 2, pp. 935–938. IEEE, 1989, doi:10.1109/ISCAS.1989.100505.

[110] W. Jin, S. Chen, and H. Jiang. Finding the K shortest paths in a time-schedule network with constraints on arcs. *Comput. Oper. Res.* 40(12):2975–2982, 2013, doi:10.1016/j.cor.2013.07.005.

[111] D. B. Johnson and S. D. Kashdan. Lower bounds for selection in $X + Y$ and other multisets. *J. ACM* 25(4):556–570, 1978, doi:10.1145/322092.322097.

[112] M. Kano. Maximum and kth maximal spanning trees of a weighted graph. *Combinatorica* 7(2):205–214, 1987, doi:10.1007/BF02579450.

[113] C. J. Karbowicz and J. M. Smith. A K-shortest paths routing heuristic for stochastic network evacuation models. *Engineering Optimization* 7(4):253–280, 1984, doi:10.1080/03052158408960642.

[114] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM* 42(2):321–328, 1995, doi:10.1145/201019.201022.

[115] N. Katoh, T. Ibaraki, and H. Mine. An $O(Kn^2)$ algorithm for K shortest simple paths in an undirected graph with nonnegative arc length. *Electron. Comm. Japan* 61(12):1–8 (1980), 1978.

[116] N. Katoh, T. Ibaraki, and H. Mine. An algorithm for finding K minimum spanning trees. *SIAM J. Comput.* 10(2):247–255, 1981, doi:10.1137/0210017.

[117] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for K shortest simple paths. *Networks* 12(4):411–427, 1982, doi:10.1002/net.3230120406.

[118] G. T. Klincsek. Minimal triangulations of polygonal domains. *Combinatorics 79*, pp. 121–123. Elsevier, Ann. Discrete Math. 9, 1980, doi:10.1016/S0167-5060(08)70044-X.

[119] K. Knight and J. Graehl. Machine transliteration. *Comput. Linguist.* 24(4):599–612, 1998.

[120] K. Knight and V. Hatzivassiloglou. Two-level, many-paths generation. *Proc. Conf. Assoc. for Computational Linguistics*, pp. 252–260, 1995.

[121] S. Kundu. An incremental algorithm for identification of longest (shortest) paths. *Integration* 17(1):25–31, 1994, doi:10.1016/0167-9260(94)90018-3.

[122] P. I. Lalov and T. G. Vasil'eva. Transformation of an oriented graph for finding the k shortest paths. *Godishnik Vissh. Uchebn. Zaved. Prilozhna Mat.* 23(4):189–194, 1987.

[123] A. G. Law and A. Rezazadeh. Computing the K-shortest paths, under nonnegative weighting. *Twenty-second Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1992)*, pp. 277–280. Utilitas Math., Congressus Numerantium 92, 1993.

[124] E. L. Lawler. A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem. *Management Sci.* 18(7):401–405, 1972, doi:10.1287/mnsc.18.7.401.

[125] E. L. Lawler. Comment on computing the k shortest paths in a graph. *Commun. ACM* 20(8):603–604, 1977, doi:10.1145/359763.359804.

[126] M. Leclerc and F. Rendl. k-best constrained bases of a matroid. *Z. Oper. Res.* 34(2):79–89, 1990, doi:10.1007/BF01415971.

[127] G. Liu and K. G. Ramakrishnan. A*Prune: an algorithm for finding K shortest paths subject to multiple constraints. *Proc. 20th Joint Conf. IEEE Computer & Communications Socs. (INFOCOM 2001)*, vol. 2, pp. 743–749, 2001, doi:10.1109/INFCOM.2001.916263.

[128] J. Ma, K. Iwama, and Q.-P. Gu. A parallel algorithm for k-minimum spanning trees. *Proc. 2nd AIZU Int. Symp. Parallel Algorithms/Architecture Synthesis*, pp. 384–388. IEEE, 1997, doi:10.1109/AISPAS.1997.581705.

[129] M. H. MacGregor and W. D. Grover. Optimized k-shortest-paths algorithm for facility restoration. *Software: Practice and Experience* 24(9):823–834, 1994, doi:10.1002/spe.4380240904.

[130] S. W. Mao, H. G. Zhang, C. C. Huang, and W. Q. Wu. A new fault-tolerance mechanism in communications based on the K shortest path algorithm. *J. Wuhan Univ. Natur. Sci. Ed.* 59(6):534–538, 2013.

[131] C. Martínez and S. Roura. Optimal sampling strategies in quicksort and quickselect. *SIAM J. Comput.* 31(3):683–705, 2001, doi:10.1137/S0097539700382108.

[132] E. Q. V. Martins and M. M. B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR* 1(2):121–133, 2003, doi:10.1007/s10288-002-0010-2.

[133] H. Maruyama. A new kth-shortest path algorithm. *IEICE Trans. Information & Systems* E76-D(3):388–389, 1993.

[134] A. Marzetta and J. Nievergelt. Enumerating the k best plane spanning trees. *Comput. Geom. Theory and Appl.* 18(1):55–64, 2001, doi:10.1016/S0925-7721(00)00029-8.

[135] T. Matsui, A. Tamura, and Y. Ikebe. Algorithms for finding a Kth best valued assignment. *Discrete Appl. Math.* 50(3):283–296, 1994, doi:10.1016/0166-218X(92)00175-L.

[136] E. W. Mayr and C. G. Plaxton. On the spanning trees of weighted graphs. *Combinatorica* 12(4):433–447, 1992, doi:10.1007/BF01305236.

[137] S.-P. Miaou and S.-M. Chin. Computing k-shortest path for nuclear spent fuel highway transportation. *Eur. J. Oper. Res.* 53(1):64–80, 1991, doi:10.1016/0377-2217(91)90093-B.

[138] E. Minieka. On computing sets of shortest paths in a graph. *Commun. ACM* 17(6):351–353, 1974, doi:10.1145/355616.364037.

[139] E. Minieka. The K-th shortest path problem. *Bull. Operations Research Soc. of America* 23:B/116, 1975.

[140] E. Minieka and D. R. Shier. A note on an algebra for the k best routes in a network. *IMA J. Appl. Math.* 11(2):145–149, 1973, doi:10.1093/imamat/11.2.145.

[141] W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM* 55(2):A11, 2008, doi:10.1145/1346330.1346336, arXiv:cs.CG/0601002.

[142] K. G. Murty. Letter to the editor—An algorithm for ranking all the assignments in order of increasing cost. *Oper. Res.* 16(3):682–687, 1968, doi:10.1287/opre.16.3.682.

[143] D. Naor and D. Brutlag. On near-optimal alignments of biological sequences. *J. Computational Biology* 1(4):349–366, 1994, doi:10.1089/cmb.1994.1.349.

[144] S. Nikolopoulos, A. Pitsillides, and D. Tipper. Addressing network survivability issues by finding the K-best paths through a trellis graph. *Proc. 16th Joint Conf. IEEE Computer & Communications Socs. (INFOCOM 1997)*, vol. 1, pp. 370–377, 1997, doi:10.1109/INFCOM.1997.635161.

[145] D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing* 8(2):159–173, 1998, doi:10.1023/A:1008990218483.

[146] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek. Integration of diverse recognition methodologies through reevaluation of N-best sentence hypotheses. *Proc. Worksh. Speech and Natural Language (HLT '91)*, pp. 83–87. Association for Computational Linguistics, 1991, doi:10.3115/112405.112416.

[147] M. M. B. Pascoal, M. E. Captivo, and J. C. N. Clímaco. A note on a new variant of Murty's ranking assignments algorithm. *4OR* 1(3):243–255, 2003, doi:10.1007/s10288-003-0021-7.

[148] A. Pauls and D. Klein. K-best A* parsing. *Proc. Joint Conf. 47th Annual Meeting of the ACL & 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL '09)*, vol. 2, pp. 958–966. Association for Computational Linguistics, 2009.

[149] A. Pauls, D. Klein, and C. Quirk. Top-down k-best A* Parsing. *Proc. ACL 2010 Conference Short Papers (ACLShort '10)*, pp. 200–204. Association for Computational Linguistics, 2010.

[150] C. R. Pedersen, L. R. Nielsen, and K. A. Andersen. An algorithm for ranking assignments using reoptimization. *Comput. Oper. Res.* 35(11):3714–3726, 2008, doi:10.1016/j.cor.2007.04.008.

[151] A. Perko. Implementation of algorithms for k shortest loopless paths. *Networks* 16(2):149–160, 1986, doi:10.1002/net.3230160204.

[152] M. Pollack. Letter to the Editor—The kth best route through a network. *Oper. Res.* 9(4):578–580, 1961, doi:10.1287/opre.9.4.578.

[153] M. Pollack. Solutions of the kth best route through a network—a review. *J. Math. Anal. and Appl.* 3(3):547–559, 1961, doi:10.1016/0022-247X(61)90076-2.

[154] M. Pollack. Shortest path solutions of the kth best route problems. *Bull. Operations Research Soc. of America*, 1969.

[155] E. S. van der Poort, M. Libura, G. Sierksma, and J. A. A. van der Veen. Solving the k-best traveling salesman problem. *Comput. Oper. Res.* 26(4):409–425, 1999, doi:10.1016/S0305-0548(98)00070-7.

[156] E. de Queirós Vieira Martins, M. M. B. Pascoal, and J. L. E. Dos Santos. Deviation algorithms for ranking shortest paths. *Internat. J. Found. Comput. Sci.* 10(3):247–261, 1999, doi:10.1142/S0129054199000186.

[157] K. A. Rink, E. Y. Rodin, and V. Sundarapandian. A simplification of the double-sweep algorithm to solve the k-shortest path problem. *Appl. Math. Lett.* 13(8):77–85, 2000, doi:10.1016/S0893-9659(00)00099-9.

[158] L. Roditty. On the k shortest simple paths problem in weighted directed graphs. *SIAM J. Comput.* 39(6):2363–2376, 2010, doi:10.1137/080730950.

[159] E. Ruppert. Finding the k shortest paths in parallel. *Algorithmica* 28(2):242–254, 2000, doi:10.1007/s004530010038.

[160] Y. Saruwatari and T. Matsui. A note on K-best solutions to the Chinese postman problem. *SIAM J. Optim.* 3(4):726–733, 1993, doi:10.1137/0803037.

[161] R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91)*, pp. 701–704, 1991, doi:10.1109/ICASSP.1991.150436.

[162] R. Schwartz and Y.-L. Chow. The N-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-90)*, pp. 81–84, 1990, doi:10.1109/ICASSP.1990.115542.

[163] D. Schweigert. Vector-weighted matchings. *Combinatorics Advances (Tehran, 1994)*, pp. 267–276. Kluwer Acad. Publ., Mathematics and Its Applications 329, 1995, doi:10.1007/978-1-4613-3554-2_19.

[164] A. Sedeño-Noda and J. J. Espino-Martín. On the K best integer network flows. *Comput. Oper. Res.* 40(2):616–626, 2013, doi:10.1016/j.cor.2012.08.014.

[165] A. Sedeño-Noda and C. González-Martín. On the K-shortest path trees problem. *Eur. J. Oper. Res.* 202(3):628–635, 2010, doi:10.1016/j.ejor.2009.06.017.

[166] T. Shibuya and H. Imai. Enumerating suboptimal alignments of multiple biological sequences efficiently. *Proc. 2nd Pacific Symp. Biocomputing*, pp. 409–420, January 1997.

[167] T. Shibuya and H. Imai. New flexible approaches for multiple sequence alignment. *J. Computational Biology* 4(3):385–413, 1997, doi:10.1089/cmb.1997.4.385.

[168] D. R. Shier. Computational experience with an algorithm for finding the k shortest paths in a network. *J. Res. Nat. Bur. Standards Sect. B* 78B:139–165, 1974.

[169] D. R. Shier. Iterative methods for determining the k shortest paths in a network. *Networks* 6(3):205–229, 1976, doi:10.1002/net.3230060303.

[170] D. R. Shier. On algorithms for finding the k shortest paths in a network. *Networks* 9(3):195–214, 1979, doi:10.1002/net.3230090303.

[171] Y.-K. Shih and S. Parthasarathy. A single source K-shortest paths algorithm to infer regulatory pathways in a gene network. *Bioinformatics* 28(12):i49–i58, 2012, doi:10.1093/bioinformatics/bts212.

[172] C. C. Skicism and B. L. Golden. Solving k-shortest and constrained shortest path problems efficiently. *Network Optimization and Applications*, pp. 249–282, Ann. Oper. Res. 20, 1989, doi:10.1007/BF02216932.

[173] C. C. Skiscim and B. L. Golden. Computing k-shortest path lengths in Euclidean networks. *Networks* 17(3):341–352, 1987, doi:10.1002/net.3230170308.

[174] F. K. Soong and E.-F. Huang. A tree-trellis based fast search for finding the N-best sentence hypotheses in continuous speech recognition. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-91)*, pp. 705–708, 1991, doi:10.1109/ICASSP.1991.150437.

[175] K. Sörensen and G. K. Janssens. An algorithm to generate all spanning trees of a graph in order of increasing cost. *Pesqui. Oper.* 25(2):219–229, 2005, doi:10.1590/S0101-74382005000200004.

[176] S. Steiner and T. Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Comput. Oper. Res.* 35(1):198–211, 2008, doi:10.1016/j.cor.2006.02.023.

[177] C. Straehle, S. Peter, U. Köthe, and F. A. Hamprecht. K-smallest spanning tree segmentations. *Pattern Recognition: 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013, Proceedings*, pp. 375–384. Springer, Lecture Notes in Computer Science 8142, 2013, doi:10.1007/978-3-642-40602-7_40.

[178] K. Sugimoto and N. Katoh. An algorithm for finding k shortest loopless paths in a directed network. *Trans. Information Processing Soc. Japan* 26:356–364, 1985.

[179] C. S. Tang and W. F. Liang. A parallel algorithm for finding K minimum spanning trees. *J. China Univ. Sci. Tech.* 20(4):464–471, 1990.

[180] D. M. Topkis. A k shortest path algorithm for adaptive routing in communications networks. *IEEE Trans. Comm.* 36(7):855–859, 1988, doi:10.1109/26.2815.

[181] M. S. Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proc. Natl. Acad. Sci. U.S.A.* 80(10):3123–3124, 1983, doi:10.1073/pnas.80.10.3123.

[182] M. M. Weigand. Algorithmus 20: Ein leistungsfähiger Algorithmus zur Bestimmung von K-kürzesten Wegen in einem Graphen [Algorithm 20: An efficient algorithm for the determination of k-shortest paths in a graph]. *Computing* 12(3):273–283, 1974, doi:10.1007/BF02293111.

[183] M. M. Weigand. Algorithmus 27: Ein neuer Algorithmus zur Bestimmung von k-kürzesten Wegen in einem Graphen [Algorithm 27: A new algorithm for the determination of k-shortest paths in a graph]. *Computing* 16(1–2):139–151, 1976, doi:10.1007/BF02241985.

[184] A. Weintraub. The shortest and the K-shortest routes as assignment problems. *Networks* 3(1):61–73, 1973, doi:10.1002/net.3230030105.

[185] A. Wongseelashote. An algebra for determining all path-values in a network with application to K-shortest-paths problems. *Networks* 6(4):307–334, 1976, doi:10.1002/net.3230060403.

[186] W. Xu, S. He, R. Song, and S. S. Chaudhry. Finding the K shortest paths in a schedule-based transit network. *Comput. Oper. Res.* 39(8):1812–1826, 2012, doi:10.1016/j.cor.2010.02.005.

[187] H. H. Yanasse, N. Y. Soma, and N. Maculan. An algorithm for determining the K-best solutions of the one-dimensional knapsack problem. *Pesqui. Oper.* 20(1):117–134, 2000, doi:10.1590/S0101-74382000000100011.

[188] J. Y. Yen. Finding the K shortest loopless paths in a network. *Management Sci.* 17:712–716, 1971, `http://www.jstor.org/stable/2629312`.

[189] S. H. Yen, D. H. Du, and S. Ghanta. Efficient algorithms for extracting the K most critical paths in timing analysis. *Proc. 26th ACM/IEEE Design Automation Conf. (DAC '89)*, pp. 649–654. ACM, 1989, doi:10.1145/74382.74497.

[190] Q. Yu and C. Sechen. Generation of colour-constrained spanning trees with application in symbolic circuit analysis. *Int. J. Circuit Theory and Appl.* 24(5):597–603, 1996, doi:10.1002/(SICI)1097-007X(199609/10)24:5<597::AID-CTA938>3.0.CO;2-Q.

[191] Q. Yu and C. Sechen. Efficient approximation of symbolic network functions using matroid intersection algorithms. *IEEE Trans. Computer-Aided Design of Integrated Circuits & Systems* 16(10):1073–1081, 1997, doi:10.1109/43.662671.

[192] Y. Zhang, S. Oepen, and J. Carroll. Efficiency in unification-based *n*-best parsing. *Proc. 10th Internat. Conf. Parsing Technologie (IWPT '07)*, pp. 48–59. Association for Computational Linguistics, 2007, `http://www.aclweb.org/anthology/W07-2207`.

[193] N. J. van der Zijpp and S. Fiorenzo Catalano. Path enumeration by finding the constrained *K*-shortest paths. *Transportation Res. B* 39(6):545–563, 2005, doi:10.1016/j.trb.2004.07.004.