Polynomial Time Prioritized Multi-Criteria
$k$-Shortest Paths and $k$-Disjoint All-Criteria-Shortest Paths

Yefim Dinitz, Shlomi Dolev, and Manish Kumar
Ben-Gurion University of the Negev, Be’er Sheva, Israel
{dinitz, dolev}@cs.bgu.ac.il, manishk@post.bgu.ac.il

Abstract. The Shortest Path Problem, in real-life applications, has to deal with multiple criteria. Finding all Pareto-optimal solutions for the multi-criteria single-source single-destination shortest path problem with non-negative edge lengths might yield a solution with the exponential number of paths. In the first part of this paper, we study specific settings of the multi-criteria shortest path problem, which are based on prioritized multi-criteria and on $k$-shortest paths. In the second part, we show a polynomial-time algorithm that, given an undirected graph $G$ and a pair of vertices $(s, t)$, finds prioritized multi-criteria 2-disjoint (vertex/edge) shortest paths between $s$ and $t$. In the third part of the paper, we introduce the $k$-disjoint all-criteria-shortest paths problem, which is solved in time $O(\min(k|E|, |E|^{3/2}))$.

Keywords: Multi-Criteria · $k$-Shortest Paths · Disjoint Shortest Paths · Path Selection.

1 Introduction and Related Work

We study a generalization of the shortest path problem in which multiple paths should be computed with consideration to multiple criteria such as cost, delay, and energy consumed edge. The application of these paths is mainly in the transportation networks and communication networks where we need to consider many criteria while computing the shortest path in the transportation network and routing in the communication networks.

Shortest path routing algorithms mainly compute the shortest simple path between two nodes, source $s$ and destination $t$. In our system setting edges can have positive weight only and no loops exist in the shortest paths. In practice, while computing the shortest path routing algorithm, in general, the graph source node always picks the shortest path for routing from source $s$ and destination $t$.

Route latency can be computed by finding multiple (almost) shortest paths in the graph from source $s$ and destination $t$. Finding multiple paths is possible by generalizing the Dijkstra algorithm to find more than one path. In the literature, finding multiple shortest path problems is referred to as $k$-shortest path problem. There are two main variations of the $k − 1$ shortest path routing problem. The first is to not only find the shortest path but also $k − 1$ other paths in non-decreasing shortest length (two shortest paths can have the same length) and paths are allowed to visit the same node more
than once, which allows a loop. In other variations, paths are not allowed to visit an already visited node. Paths are required to be simple and loopless. The $k$-shortest path routing has applications in many areas such as geographical path planning, multiple object tracking, and transportation networks.

While finding multi-criteria $k$-shortest path, some nodes and edges can be traversed by more than one path which may yield accumulated load (and delay) on the shared node/edge. Delay can be avoided by choosing non-overlapping $k$-shortest paths, which is called $k$-disjoint shortest paths. These paths can be categorized further in two types: multi-criteria node-disjoint paths and multi-criteria edge-disjoint paths. These disjoint paths are distinct paths in the graph from source $s$ and destination $t$ which has a wide range of applications other than routing, such as multi-commodity flow.

**Multi-Criteria Shortest Paths**

Many real-life problems can be represented as a network, such as transportation networks, biological networks, and communication networks. In these networks, finding the shortest path resolves many issues such as routing, the distance between two molecules. In general, for finding the shortest path, we consider the criterion (objective) of edge weight (cost), which is called the Shortest Path Problem (SPP) with single criterion. A Multi-Criteria Shortest Path Problem (MCSPP) consists of more than one objective while computing the shortest path between source and destination.

In the literature, many existing results are available on MCSPP. The first result on MCSPP was analyzed by Hansen [14], which was focused on a Bi-criteria Shortest Path Problem. Hansen proved that a family of problems exists, for which, any path between a given pair of nodes is a non-dominated path. Hence, any algorithm for solving MCSPP is exponential in the worst-case analysis. Thus, no polynomial time algorithm can guarantee to determine all non-dominated paths in polynomial time.

Others consider the case of Bi-Criteria, introducing edges that have cost and delay criteria. The goal is to find an $s$-$t$ path that minimizes the cost while having a delay of at most $T$ (threshold). This scenario is known as Restricted or Constrained Shortest Path Problem. Restricted Shortest Path (RSP) is often used in QoS (quality of service) routing, where the goal is to route a package along the cheapest possible path while also satisfying some quality constraint for the user. RSP is known to be NP-Hard [11]. As a result, many fully polynomial-time approximation scheme (FPTAS) [3,12,15,21,25,26] were designed. Heuristic based algorithms discussed in these papers [19,22,24,28].

The Pareto techniques are the only non-heuristics algorithms to address the $k$-shortest multi-criteria, and they are exponential. Our polynomial solutions as the first to improve the exponential Pareto solution, there are several novel approaches here, one is the observation that summing all weights, over all edges in the graph, of particular criteria may add only $O(\log n)$ bits, and therefore will support a polynomial solution where criteria summation along a path does not intervenes with other criteria representation when the combined criteria are designed to have enough bits for each criterion. Other results are novel as well, we were happy to find that the vast number of sophisticated approximation results can be improved in many cases by our polynomial exact solutions.
**k-Shortest Path Problem**

The problem of finding the shortest paths in an edge-weighted graph is an important and well-studied problem in computer science. Dijkstra’s sequential algorithm [5] computes the shortest path to a given destination vertex from every other vertex in \( O(m + n \log n) \) time. The \( k \)-shortest paths (KSP) asks to compute a set of top \( k \)-shortest simple paths from vertex \( s \) to vertex \( t \) in a digraph. In 1971, Yen [27] proposed the first algorithm with the theoretical complexity of \( O(kn(m + n \log n)) \) for a digraph with \( n \) vertices and \( m \) edges.

The best-known algorithm for the \( k \)-shortest path problem was proposed by Eppstein [7] which allows non-simple paths and runs in \( O(m + n \log n + kn) \) time. In the initialization phase, the algorithm uses a shortest path tree to build a data structure that contains information about all \( s-t \) paths and how they interrelate with each other, in time \( O(m + n) \). The running time for the initialization can be reduced from \( O(m + n \log n) \) to \( O(m + n) \) if the shortest path tree can be computed in time \( O(m + n) \).

In the enumeration phase, a path graph is constructed. The path graph is a min-heap where every path starting from the common root corresponds to a \( s-t \) path in the original graph. If we want the output paths to be sorted by the length in increasing order then the enumeration phase requires \( O(k \log k) \) time. Frederickson’s heap selection algorithm [10] can be used to enumerate the paths after the initialization phase in \( O(k) \) time. Other KSP algorithms discussed in these papers [9, 13, 16–18, 20, 23].

**k-Disjoint Shortest Path Problem**

The \( k \)-disjoint shortest path problem on a graph with \( k \) source-destination pairs \((s_i, t_i)\) looks for \( k \) pairwise node/edge disjoint shortest \( s_i - t_i \) paths. The output is prioritized: the first path should be shortest, the second one should be shortest conditioned by that property of the first path and by disjointness, and so on. The \( k \)-disjoint shortest path problem is known to be NP-complete if \( k \) is part of the input.

This problem disjoint shortest path was first considered by Eilam-Tzoreff [6]. Eilam-Tzoreff provided a polynomial-time algorithm for \( k=2 \), based on a dynamic programming approach for the weighted undirected vertex-disjoint case. This algorithm has running time of \( O(|V|^8) \). Later, Akhmedov [2] improved the algorithm of Eilam-Tzoreff whose running time is \( O(|V|^6) \) for unit-length case of 2-Disjoint Shortest Path and \( O(|V|^7) \) for the weighted case of 2-disjoint shortest path. In both cases Akhmedov [2] considered the undirected vertex disjoint shortest path.

## 2 Finding Prioritized Multi-Criteria k-Shortest Paths in Polynomial Time

### 2.1 Reducing Multi-criteria Weight to Single Weight

We considered multi-criteria in weight(cost)-function in a prioritized manner. In our reduction from multi-criteria to single criterion, we ensemble the weights of the monotonic prioritized criteria into one weight. Next we present a reduction of the prioritized multi-criteria \( k \)-shortest problem to the single criterion \( k \)-shortest path problem. The
idea is to combine the different weights into a single “ensembled” weight, such that the most significant part of the ensembled weight, is the weight of the most important criteria. Say, using the first most important $k_1$ bits, that suffice to accumulate the sum of weights of the most prioritized criteria. The second most important weight resides in the next $k_2$ bits of the edge weight, and so on and so forth. Our algorithms deals with any number of criteria (weights).

Fig. 1. Example for reduction of multi-criteria weight to single weight, multi-criteria weights (3,4,5) on the edge represents weight in the form of $(3=w_1, 4=w_2, 5=w_3)$

Fig. 2. Example for reduction of multi-criteria weight to single weight for binary bits

Fig. 3. Length of the (only and therefore) shortest path from $A$ to $E$, Length of shortest path $= 6478\ (1100101001110)$, each color represent the respective sum of weights for each criterion.

We illustrate the conversion using the following example. In the example we showed the reduction of multi-criteria weight to single weight using an example. In Fig. 1 each edge holds a vector of criteria and sum of weights. In Fig. 2 each edge holds a vector of criteria in binary form and sum of weights in the colored binary form where each color represents the weight of each criterion. In Fig. 3 we computed the only shortest path between node $A$ and $E$ and the total length of path represented in a colored binary number, in that also each color represents the sum of the weight of each individual criteria. The summarized calculations are in Table 1.

2.2 Prioritized Multi-Criteria $k$-Shortest Simple Paths

The multi-criteria shortest path problem has a rich history, several approximation and heuristic-based algorithms have been proposed to solve it. Instead of considering the approximation or heuristic approach, we are interested in problem families for which a polynomial solution exists. For example, (1) if one criterion is that no edge on the
Table 1. Reduction of multi-criteria weight to single weight, where $a_i = \sum w_i$ weights over all edges. $a_1 = 12$, $a_2 = 20$, $a_3 = 14$, $\text{max}_a = 20$, for which value of $\text{max}_a$ is ($2^5 > \text{max}_a \geq 2^4 > 14$), so choose $2^5$ due to 20. In fact, according to $a_3$ only four bits suffice for the description of the $w_3$ to maximal value in a shortest (in fact any simple) path, as $14 < 2^4$. We can use 5 bits for first criterion but in this example 4 bits are enough to represent $w_3$.

|       | Edge 1 | Edge 2 | Edge 3 | Edge 4 |
|-------|--------|--------|--------|--------|
| $w_1$ | 3      | 4      | 1      | 4      |
| $w_2$ | 4      | 3      | 6      | 7      |
| $w_3$ | 5      | 2      | 5      | 2      |
| $w'_1 = w_1 \cdot 2^5$ | 1536 | 2048 | 512 | 2048 |
| $w'_2 = w_2 \cdot 2^4$ | 64   | 48    | 96    | 112   |
| $w'_3 = w_3 \cdot 2^0$ | 5    | 2     | 5     | 2     |
| **Total Weight** | 1605 | 2098 | 613 | 2162 |

**Binary Conversion**

|       |       |       |       |       |
|-------|-------|-------|-------|
| $w_1$ | 11    | 100   | 01    | 100   |
| $w_2$ | 100   | 11    | 110   | 111   |
| $w_3$ | 101   | 10    | 101   | 10    |
| $w'_1 = w_1 \cdot 2^5$ | 1000000000 | 100000000000 | 100000000000 | 100000000000 |
| $w'_2 = w_2 \cdot 2^4$ | 1000000 | 110000 | 110000 | 1110000 |
| $w'_3 = w_3 \cdot 2^0$ | 101 | 10 | 101 | 10 |
| **Total Weight** | 1010000101 | 10001100010 | 1011000101 | 100011100010 |

path should weigh more than a given threshold ($T$), then when computing the shortest multi-criteria algorithm, do not consider this edge. (2) Another family of multi-criteria is prioritized multi-criteria where one would like to optimize the first criteria $c_1$, and within all solutions that optimize $c_1$, find the optimal solution for the second criteria $c_2$, and so on. (3) A combination of the two multi-criteria as above.

Thus, as explained above, to ensemble the weights of the monotonic prioritized criteria into one weight, we use the most important part of an edge ensemble weight for the most important criteria, and the least important part of an edge ensemble weight for the least important criteria, and similarly for criteria in between.

To make sure that the portion of edges weight dedicated to criteria does not overlap, we assign each portion a span of bits in the ensemble weight of an edge to suffice for accumulating the criteria weight along the (shortest) path. We can bound the number of bits needed for accumulating the bound on the shortest path, by summing up all weights of the criteria in hand over all the edges in the graph.

Finding the $k$-shortest paths with the ensemble weights results that these are $k$-shortest paths in the most important criteria $c_1$, as all other criteria do not compete with the most important part of the weights when computing the shortest path(s). Thus, the second criterion $c_2$ breaks ties among the paths as above with the same value of the first criterion. In particular, if the weight of the heaviest shortest path according to $c_1$ is $w_1$, the selection from the set of the shortest paths with weight $w_1$ will be according to the second prioritized criterion $c_2$. If the set of shortest paths with
Algorithm 1: Generalized Dijkstra Algorithm for Multi-Criteria Shortest Path

**Input:** A graph $G = (V, E)$, where each edge $e$ holds vector $\bar{w}(e)$, where $\bar{w}(e) = (w_1(e), w_2(e), ..., w_q(e))$ and $w_i(e)$ is the weight of $e$ w.r.t. criterion $c_i$
- $s \in V$: source node
- $t \in V$: destination node
- $T$: Threshold

**Output:** Multi-criteria shortest simple path

Procedure GeneralizedDijkstra(Graph $G$, Source $s$, Destination $t$)
1. Initialize: Source $\{s\}$
2. $dist(s) = 0$
3. $EW = 0$
4. $r_q = 0$
5. for each vertex $v$ except for $s$ do
   6. $dist(v) = \infty$
   7. $S = \phi$
   8. $Q = V$
   9. Compute $W_i = \sum_{e \in E} w_i(e), 1 \leq i \leq q$
   10. Compute $l_i = \lceil \log_2(W_i + 1) \rceil, 1 \leq i \leq q$
   11. Compute $r_i = \sum_{j=i+1}^{q} l_i, 0 \leq i \leq q - 1$
   12. for each edge $e$
      13. $EW(e) = \sum_{i=1}^{q} (2^{r_i} w_i(e))$
      14. if $EW(u, v) \geq T$ then
         15. Delete edge $(u, v)$
   16. while $Q \neq \phi$ do
      17. $u = \text{Extract-Min}(Q)$
      18. $S = S \cup \{u\}$
      19. for each vertex $v \in G.Adj[u]$ do
         20. $\text{Relax}(u, v)$
   21. Procedure Relax($u, v$)
      22. for each neighbor of $u$
         23. if $dist(v) > dist(u) + EW(u, v)$ then
            24. $dist(v) = dist(u) + EW(u, v)$

$w_1$ is chosen according to the second criteria where $w_2$ is the shortest among them, then from the set of paths with weights $w_1$ and $w_2$ paths with the lightest weight according to the third criterion are chosen, and so on and so forth.

The ensemble of the criteria weight into one wight implies finding monotonic multi-criteria $k$-shortest paths. These paths are not necessarily disjoint (as the $k$-shortest simple paths) and also for edge/node disjoint paths. These paths can be computed in polynomial time as long as $k$ is fixed.
Algorithm 2: Multi-Criteria $k$-Shortest Simple Paths

**Input:** A graph $G = (V, E)$
- $s$: source node
- $t$: destination node
- $k$: number of shortest path to find

**Output:** Multi-Criteria $k$-Shortest Simple Paths

1. $P = \text{Empty path set}$
2. $P = \text{Empty stack}$
3. $\text{noMorePaths} = \text{false}$
4. while($|P| < k \land \text{noMorePaths} = \text{false}$) do
   5. $G \leftarrow \text{reduce}(G)$
   6. $P = \text{GeneralizedDijkstra}(G, s, t)$
   7. if $|P| = 0$ then
      8. $\text{noMorePaths} = \text{true}$
   else
      9. $P = P \cup P$
      10. $E \leftarrow E \setminus \{\text{lightestEdge}(P)\}$
11. return $P$

Our approach is based on generalized Dijkstra algorithm for the multi-criteria shortest path. Using the Dijkstra algorithm, it is possible to determine the shortest distance (or the least cost/least delay) between a start node and any other node in a graph. The idea of the algorithm is to continuously apply the original Dijkstra algorithm with the precomputed ensembled weight for each edge, while removing the edges that hold more ensembled weight than Threshold ($T$).

For multiple criteria, to avoid the exponential number of paths, we reduce the set of all criteria as a single value for each edge. We reduce the prioritized multi-criteria by a reduction to a single criterion. Let us define the ensembled edge weights as follows. Let $W_i = \sum_{e \in E} w_i(e)$, $1 \leq i \leq q$. Let $l_i = \lceil \log_2(W_i + 1) \rceil$, $1 \leq i \leq q$, and let $r_q = 0$. $r_i = \sum_{j=i+1}^{q} l_i$, $0 \leq i \leq q - 1$. The ensembled weight of the edge $e \in E$ is defined to be $EW(e) = \sum_{j=1}^{q} (2^{r_j} w_j(e))$. As usual we define the ensembled weight of any path $P$ as $EW(P) = \sum_{e \in P} EW(e)$.

Our approach consists of the following steps: $Q$ is the set of nodes for which the shortest path has not been found. Initialize the source node with distance 0 and all nodes with distance “infinite”. Reduce the multi-criteria into single criterion. At each iteration, the node $v$ that has the minimum distance (sum of weights $EW$) value to the source is added to the $S$, which provides the shortest path from the source node to the destination node.

For computing the multi-criteria $k$-shortest simple paths, we use the Algorithm 1 (generalized Dijkstra algorithm) for computing the shortest path. Consider $P$ an empty path set, which is used for storing final $k$ shortest paths and $P$ an empty stack used for storing tentative shortest path. Execute the while loop for $k$ times, in each step compute a shortest path using $\text{GeneralizedDijkstra}(G, s, t)$ (Algorithm 1), put
it into $P$, and store $P$ in the path set $P$. The $k^{th}$ shortest paths might share edges and sub-paths with $(k-1)^{th}$ shortest path. Such overlapping can be avoided by removing the lightest edge from the last shortest path and recompute the $k^{th}$ on reduced graph. This can be computed using Algorithm 2.

Algorithm 2 as follows: A source $s$ node and destination $t$ node is given in graph $G$ with the $k$ number of shortest path to $t$ found. Initialize $P$ = Empty path set, $P'$ = Empty stack and noMorePaths = false. In each round compute a shortest path using GeneralizedDijkstra$(G, s, t)$ and store in stack $P$. If no more path stored in stack $P$, stop the execution. Otherwise, store the path from stack $P$ in path set $P$, remove the lightest edge from path and reduce the graph. Repeat the same execution on reduced graph until $k$ paths found. Note that when there is a constant number, $c$, of criteria then (any chosen constant) $k'$ paths with each of the possible priorities can be found keeping the computation polynomial.

**Theorem 1.** The prioritized multi-criteria $k$-shortest path problem in a undirected graph can be solved in polynomial time.

**Proof.** The single criterion $k$-shortest path problem is solvable in polynomial time. We polynomially reduced the multi-criteria weights where criteria are used in the prioritized manner to the single criterion weight. So prioritized multi-criteria $k$-shortest path problem is also solvable in polynomial time.

### 3 Prioritized Multi-Criteria 2-Disjoint (Node/Edge) Shortest Paths

In this section, we suggest an algorithm solving the 2-shortest paths edge/node independent problem (see Eilam-Tzoreff [6] and the references therein) for the case of prioritized criteria from a single source $s$ to a single destination $t$ in an undirected graph $G$.

We reduce the prioritized multi-criteria case to the case of a single criterion $EW$, similarly to Section[2.2] Further, for finding 2-disjoint shortest paths from $s$ to $t$, we use a reduction to the case where two sources and two destinations are given (described later). Then, we find the 2-disjoint shortest paths in the resulted graph $G$ by using the algorithm of Akhmedov [2], which computes the 2-disjoint shortest paths for two sources and two destinations in time $O(|V|^7)$.

Let us describe our reduction. For the edge-disjoint case, it is simple. We add to $G$ two nodes $s_1, s_2$ with dummy edges $(s_1, s), (s_2, s)$, two nodes $t_1, t_2$ with dummy edges $(t, t_1), (t, t_2)$, define the weight to be zero for the dummy edges, and declare $s_1, s_2$ be the sources and $t_1, t_2$ the destinations instead of $s$ and $t$. After finding the 2-disjoint shortest paths in the resulting graph $G$, we return them with the dummy edges removed.

The reduction for the node-disjoint case is more complicated. We add to $G$ four nodes $s_1, s_2, t_1, t_2$, which will be the sources and destinations instead of $s, t$. If $G$ contains edge $(s, t)$, then we replace it by edge $(s', t')$ of the same weight, and add dummy edges $(s_1, s'), (s_2, s'), (t', t_1), (t', t_2)$. For any other edge $(s, v)$ incident to $s$, we replace it by edge $(s_v, v)$ of the same weight, where $s_v$ is a new node, and add
dummy edges \((s_1, s_2), (s_2, s_1)\). Symmetrically, for any other edge \((v, t)\) incident to \(t\), we replace it by edge \((v, t_v)\) of the same weight, where \(t_v\) is a new node, and add dummy edges \((t_v, t_1), (t_v, t_2)\). The weights of all dummy edges are set be 1. Finally, we remove nodes \(s\) and \(t\) with no incident edges. After finding the 2-disjoint shortest paths in the resulting graph \(\tilde{G}\), we return their abridged variant: with the dummy edges incident to their end-nodes \(s_i\) and \(t_i\) shrunken to \(s\) and \(t\), respectively.

Let us show the correctness of the latter reduction. A necessary condition for using the algorithm of Akhmedov is that the terminals quadruple \((s_1, s_2, t_1, t_2)\) is not rigid, where it is called rigid if \(s_1, t_1 \in L(s_2, t_2)\) and \(s_2, t_2 \in L(s_1, t_1)\), where \(L(s_i, t_i)\) is the set of all nodes belonging to at least one shortest path between \(s_i\) and \(t_i\). Let us prove that \((s_1, s_2, t_1, t_2)\) is not rigid in \(\tilde{G}\). Assume for the contradiction, w.l.o.g., that a shortest path \(P\) from \(s_1\) to \(t_1\) in \(\tilde{G}\) contains two consequent edges \((u, s_2)\) and \((s_2, v)\). Consider path \(P'\) obtained from \(P\) by replacing its prefix from \(s_1\) to \(v\) by edge \((s_1, v)\), with weight 1. Path \(P'\) from \(s_1\) to \(t_1\) is lighter than \(P\), since the weights of three removed edges: the first one of \(P\), \((u, s_2)\), and \((s_2, v)\), are 1 each,—a contradiction.

Let us show the legality and optimality of the returned solution. Since the paths returned by algorithm of Akhmedov are node-disjoint, also their abridged variants are node-disjoint. By the reasons as in the proof of non-rigidity of \((s_1, s_2, t_1, t_2)\), the returned paths do not contain any terminal out of \(s_1, s_2, t_1, t_2\). Therefore, their abridged variants do not contain dummy edges, and thus are legal paths in \(G\). Let \((P_1^*, P_2^*)\) be the optimal pair of 2-disjoint shortest paths from \(s\) to \(t\) in \(G\). The paths corresponding to them in \(\tilde{G}\)—obtained from them by the operations as in the reduction applied to their first and last edges—are node-disjoint and have weights greater by 2 than the weights of \(P_1^*\) and \(P_2^*\). The optimal paths in \(\tilde{G}\) are not worse, and their abridged variants are by 2 lighter. Therefore, the paths pair returned by the reduction is not worse than pair \((P_1^*, P_2^*)\), as required.

The algorithm as above is presented in pseudo-code in Algorithm 3. Its correctness, together with the polynomiality of the algorithm of Akhmedov and of the reduction, implies the following statement.

**Theorem 2.** The prioritized multi-criteria 2-disjoint shortest path problem in an undirected graph can be solved in polynomial time.
Algorithm 3: Prioritized Multi-criteria 2-Disjoint Shortest Paths

Input: A graph $G = (V, E)$, where each edge $e$ holds vector $\bar{w}(e)$, where
$\bar{w}(e) = (w_1(e), w_2(e), ..., w_q(e))$ and $w_i(e)$ is the weight of $e$ w.r.t. criterion $c_i$
- $s \in V$: source node
- $t \in V$: destination node

Output: Prioritized multi-criteria 2-disjoint shortest paths from $s$ to $t$

1. Compute $W_i = \sum_{e \in E} w_i(e), 1 \leq i \leq q$
2. Compute $l_i = \lceil \log_2(W_i + 1) \rceil, 1 \leq i \leq q$
3. Compute $r_i = \sum_{j=i+1}^q l_j, 0 \leq i \leq q - 1$
4. For each edge $e \in E$
   5. $EW(e) = \sum_{j=1}^q (2^{r_j} \cdot w_i(e))$
6. Construct extended graph $\hat{G}$ with two sources $s_1, s_2$ and two destinations $t_1, t_2$ as described in the text
7. $(\hat{P}_1, \hat{P}_2) \leftarrow 2DSP(\hat{G}, EW, s_1, t_1, s_2, t_2)$
8. Transform $\hat{P}_1, \hat{P}_2$ to their abridged variants $P_1, P_2$ (see the text)
9. Return $(P_1, P_2)$
10. Procedure 2DSP($G', w, s_1, t_1, s_2, t_2$)
11. Execute the algorithm of Akhmedov [4] for computing 2-disjoint shortest paths in graph $G'$ with edges weights $w$

4 $k$-Disjoint All-Criteria-Shortest Paths

This section studies the following $k$-disjoint all-criteria-shortest paths problem. The input is a directed graph $G = (V, E)$, $q$ weight functions $w_i$ on edge set $E$, $1 \leq i \leq q$, source node $s$ and destination node $t$, $s, t \in V$, and integer $k$. We say that a path $P^*$ from $x$ to $y$ is the shortest w.r.t. criterion $c_i$, if $c_i(P^*) = \sum_{e \in P^*} w_i(e)$ is minimal among all $c_i(P) = \sum_{e \in P} w_i(e)$ over all paths $P$ from $x$ to $y$. A set of $k$ (edge-)disjoint paths from $s$ to $t$ such that each one of them is shortest regarding each one of the $q$ criteria is sought for, if exists. A theoretical analysis is provided, and a polynomial algorithm solving the problem is presented and analyzed. We first reduce the problem to its single criterion version, then reduce the latter problem to finding $k$ disjoint paths from $s$ to $t$ in a certain sub-graph of $G$, if exist, and finally present an algorithm for finding them, if exist, using known techniques: max-flow finding and flow decomposition.

We assume that each node is reachable from $s$ and that $t$ is reachable from each node in $G$; otherwise, the extra nodes could be removed from $G$, leaving the problem equivalent. Let distances $d_s(x, x)$ and $d_t(y, t)$, $x, y \in V$, denote the lengths of shortest paths from $s$ to $x$ and from $y$ to $t$, respectively, w.r.t. criterion $c_i$ in $G$. We assume that there is no negative cycle in $G$ w.r.t. any weight function $w_i$, in order that shortest paths will exist. (Here and in what follows, see, e.g., [4] for the basic information on graph algorithms.)

Let us define the auxiliary aggregated weight $w(e) = \sum_i w_i(e)$ for each edge $e \in E$, and the auxiliary aggregated criterion $c(P) = \sum_{e \in P} w(e)$ for each path $P$ in $G$. Note that there is no negative cycle in $G$ w.r.t. weight function $w$, by our assumption;
hence, shortest paths w.r.t. $c$ exist, and thus distances $d(s, x)$ and $d(y, t)$, $x, y \in V$, w.r.t. $w$ are well defined. Observe that $d(s, t) = \min_P c(P) = \min_P \sum_i c_i(P) \geq \sum_i \min_P c_i(P) = \sum_i d_i(s, t)$, where each minimum is taken over all paths $P$ from $s$ to $t$ in $G$. Moreover, the equality $d(s, t) = \sum_i d_i(s, t)$ holds, if and only if, there exist paths from $s$ to $t$ shortest w.r.t. each criterion $c_i$; in this case, these paths and only these are shortest w.r.t. criterion $c$.

As a consequence, we obtain the reduction from our problem to the auxiliary single criterion disjoint shortest paths problem, as follows. If $d(s, t) > \sum_i d_i(s, t)$, then no paths from $s$ to $t$ shortest w.r.t. each criterion $c_i$ exist. Checking this could be made via $q + 1$ executions of algorithm Dijkstra on $G$, w.r.t. each criterion $c_i$ and w.r.t. criterion $c$. Otherwise, the $k$ disjoint paths, as required, are the $k$-disjoint shortest paths w.r.t. criterion $c$, if exist. In what follows, we present the solution— an analysis and an algorithm—to the single criterion disjoint shortest $k$ paths problem. Let us begin with the problem analysis.

**Lemma 1.**
1. Node $u$ belongs to at least one shortest path from $s$ to $t$ if and only if $d(s, u) + d(u, t) = d(s, t)$.
2. Edge $(u, v)$ belongs to at least one shortest path from $s$ to $t$ if and only if $d(s, u) + w(u, v) + d(v, t) = d(s, t)$.

*Proof.* (1) If a path $P$ from $s$ to $t$ going via $u$ is shortest, then it is known that its parts from $s$ to $u$ and from $u$ to $t$ are also shortest. In other words, their lengths are $d(s, u)$ and $d(u, t)$, respectively. The equation as required follows. If $d(s, u) + d(u, t) = d(s, t)$, then the concatenation of the shortest paths from $s$ to $u$ and from $u$ to $t$ is a path from $s$ to $t$ of length $d(s, t)$, as required. The proof of item (2) is similar.

Let us define $\tilde{V}$ as the subset of nodes as in Lemma 1(1) and $\tilde{E}$ as the subset of edges as in Lemma 1(2). We denote by $G$ the (sub-)graph $(\tilde{V}, \tilde{E})$.

**Lemma 2.**
1. Each shortest path from $s$ to $t$ is contained in $G$.
2. Each path from $s$ to $t$ contained in $G$ is shortest.

*Proof.* The proof of item (1) is a straightforward corollary from Lemma 1.

(2) Let $P$ be any path from $s$ to $t$ in $G$. Assume to the contrary that $c(P) > d(s, t)$. Let us denote by $P_v$ the prefix of $P$ ending at $v, v \in P$. Note that the (degenerate) path $P_s$ from $s$ to itself is shortest: $c(P_s) = 0 = d(s, s)$. Let $v$ be the first node on $P$ such that $c(P_v) > d(s, v)$; let $P'$ be a shortest path from $s$ to $v, c(P') < c(P_s)$. Denote by $P''$ some shortest path from $v$ to $t$. Let $(u, v) \in G$ be the edge on $P$ entering $v$. By definition of $v, c(P_u) = d(s, u)$. By Lemma 1(2), $d(s, u) + w(u, v) + d(v, t) = d(s, t)$. Let us concatenate $P'$ and $P''$.

$$c(P' \cdot P'') < c(P_u \cdot P''') = c(P_u \cdot (u, v) \cdot P''') = d(s, u) + w(u, v) + d(v, t) = d(s, t).$$

Thus, $c(P' \cdot P'') < d(s, t), a contradiction to the definition of $d(s, t)$.

By Lemma 2 we have a reduction from the single criterion disjoint shortest $k$ paths problem to finding $k$ disjoint paths from $s$ to $t$ in $G$, if exist. Finding such paths, if exist, may be done by known max-flow techniques. Let $N$ be the flow network $(\tilde{G}, s, t)$ with unit capacities of all its edges. In what follows, we omit detailed proofs, since the material is basic.
Algorithm 4: $k$-Disjoint All-Criteria-Shortest Paths Finding

**Input:** A directed graph $G = (V, E)$, weight functions $w_i$, $1 \leq i \leq q$, on edge set $E$, source node $s$, destination node $t$, and integer $k$

**Output:** $k$-Disjoint All-Criteria-Shortest Paths, if exist

1. Compute the aggregated weight function $w(e) = \sum_{i=1}^{q} w_i(e)$, for all edges $e \in E$
2. Run $q + 1$ times algorithm Dijkstra for finding distance functions $d$ and $d_i$ w.r.t. weights $w$ and $w_i$, respectively, $1 \leq i \leq q$
3. if $d(s, t) > \sum_{i=1}^{q} d_i(s, t)$ then
   - return "No path from $s$ to $t$ shortest w.r.t. each criterion $c_i$ exist"
4. else
   5. set sets $\tilde{V}$ and $\tilde{E}$ be empty
   6. for each node $u$ in $V$
      - if $d(s, u) + d(u, t) = d(s, t)$ then
        - add node $u$ to $\tilde{V}$
   7. for each edge $(u, v)$ in $E$
      - if $d(s, u) + w(u, v) + d(v, t) = d(s, t)$ then
        - add edge $(u, v)$ to $\tilde{E}$
   8. construct flow network $N = (\tilde{G} = (\tilde{V}, \tilde{E}), s, t)$ with unit capacities on all edges
   9. run the max-flow algorithm on $N$, finding flow $f^*$
10. if the value of $f^*$ is less than $k$ then
    - return "There exist no $k$ paths from $s$ to $t$ shortest w.r.t. each criterion $c_i$"  
   11. else
      12. initialize path set $P$ be empty
      13. repeat $k$ times
      14. Phases
      15. Phase 1
         16. set stack $S$ be empty and set $v$ be $t$, and mark it
         17. repeat
            18. choose an edge $(u, v)$ with $f(u, v) = 1$ and push it into $S$
            19. set $v$ be $u$, and mark it
            20. If $v = s$, break the repeat loop and go to Phase 3
            21. If $v$ is marked, suspend the repeat loop and go to Phase 2
         22. Phase 2
            23. set $z$ be $v$
            24. repeat
               25. pop edge $(u, v)$ from $S$, and set $f(u, v) = 0$
               26. set $v$ be $u$, and unmark it
               27. If $v = z$, mark $v$ and resume the repeat loop of Phase 1
            28. Phase 3
               29. set edge list $P$ be empty and unmark $v = s$
               30. repeat while $v \neq t$
                  31. pop edge $(v, u)$ from $S$
                  32. set $f(u, v) = 0$, and add $(v, u)$ to $P$
                  33. set $v$ be $u$, and unmark it
                  34. add $P$ to $P$
               35. return $P$

Proposition 1. A set of $k$ disjoint paths from $s$ to $t$ in $\tilde{G}$ exists if and only if the value of maximal $f_{\text{low}}$ in $N$ is at least $k$.

Proof. **direction only if** Assume that $\mathcal{P}$ is a set of $k$ disjoint paths from $s$ to $t$. Let us define flow $f$ by setting it to 1 on all edges belonging to the paths in $\mathcal{P}$ and to 0 on all other edges. It is easy to see that $f$ is a flow of value $k$ from $s$ to $t$ in $N$. Hence, the value of a max-flow in $N$ is at least $k$.

**direction if** Let $f$ be an integer (that is 0/1) flow in $N$ of value at least $k$. Let $E_f$ be the set of edges with flow 1 in them. Executing the following triple-phased path-finding routine $k$ times, beginning from an empty path set $\mathcal{P}$, finds a set $\mathcal{P}$ of $k$ disjoint paths from $s$ to $t$ in $\tilde{G}$.

**Phase 1** Set stack $S$ be empty. Set $v$ be $t$ and mark it. Choose an edge $(u, v)$ in $E_f$ and push it into $S$. Set $v$ be $u$, mark it, and continue in the same way. If we arrived at $v = s$, go to Phase 3. If we arrived at a marked node $v$, suspend Phase 1 and go to Phase 2.

**Phase 2** Set $z$ be $v$. Pop edge $(u, v)$ from $S$. Unmark $u$, set $f(u, v) = 0$, set $v$ be $u$, and continue in the same way. When arrived at $v = z$, mark $v$ and resume Phase 1.

**Phase 3** Set list $P$ be empty. Unmark $v = s$. Pop edge $(u, v)$ from $S$. Unmark $u$, add $(u, v)$ to $P$, set $f(u, v) = 0$, set $v$ be $u$, and continue in the same way. Upon arrival at $v = t$, add $P$ to $\mathcal{P}$.

Let us explain briefly the correctness. After each execution of Phase 2 (removal of a flow cycle), $f$ becomes a correct flow with the same value. After each execution of Phase 3 (removal of a flow path), $f$ becomes a correct flow with a value smaller by 1. As far as the flow value is non-zero, there exist at least one edge in $E_f$ entering $t$. For a correct flow, at each node $v$, the numbers of edges in $E_f$ incoming and outgoing $v$ are equal; hence, if there is an edge outgoing $v$ in $E_f$, then also an edge incoming $v$ exists in $E_f$.

We conclude that for solving the problem, it is sufficient to find a max-flow in $N$, and if the flow value is at least $k$, to execute the path-finding routine $k$ times. The pseudo-code of the described solution scheme is presented in Algorithm 4. Consider its running time bound. A flow either of value $k$, if exists, or a max-flow, otherwise, in a network with unit edge capacities can be found in time $O(\min(k|E|, |E|^{3/2}))$ (see [1][8]). All executions of the path-finding routine together take $O(|E|)$ time, since each edge is processed in time $O(1)$ in total. Summarizing, the running time bound of Algorithm 4 is $O(\min(k|E|, |E|^{3/2}))$.

5 Conclusion

We presented polynomial time multi-criteria paths algorithms, which include prioritized multi-criteria $k$-shortest paths algorithm, multi-criteria 2-disjoint (vertex/edge) shortest paths algorithm for undirected graphs, and $k$-disjoint all-criteria-shortest paths algorithm.
References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice hall (1993)
2. Akhmedov, M.: Faster 2-disjoint-shortest-paths algorithm. In: Computer Science - Theory and Applications - 15th International Computer Science Symposium in Russia, CSR 2020, Yekaterinburg, Russia, June 29 - July 3, 2020, Proceedings. pp. 103–116 (2020)
3. Bernstein, A.: Near linear time $(1 + \varepsilon)$-approximation for restricted shortest paths in undirected graphs. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012. pp. 189–201 (2012)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition. The MIT Press, 3rd edn. (2009)
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. 1(1), 269–271 (Dec 1959)
6. Eilam-Tzoreff, T.: The disjoint shortest paths problem. Discret. Appl. Math. 85(2), 113–138 (1998)
7. Eppstein, D.: Finding the k shortest paths. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 154–165 (1994)
8. Even, S.: Graph Algorithms. Cambridge University Press, 2 edn. (2011)
9. Feng, G.: Finding $k$ shortest simple paths in directed graphs: A node classification algorithm. Networks 64(1), 6–17 (2014)
10. Frederickson, G.N.: An optimal algorithm for selection in a min-heap. Inf. Comput. 104(2), 197–214 (1993)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., USA (1990)
12. Goel, A., Ramakrishnan, K.G., Kataria, D., Logothetis, D.: Efficient computation of delay-sensitive routes from one source to all destinations. In: Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, Anchorage, Alaska, USA, April 22-26, 2001, pp. 854–858 (2001)
13. Gotthilf, Z., Lewenstein, M.: Improved algorithms for the $k$ simple shortest paths and the replacement paths problems. Inf. Process. Lett. 109(7), 352–355 (2009)
14. Hansen, P.: Bicriterion path problems. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Application. pp. 109–127. Springer Berlin Heidelberg, Berlin, Heidelberg (1980)
15. Hassin, R.: Approximation schemes for the restricted shortest path problem. Math. Oper. Res. 17(1), 36–42 (1992)
16. Hershberger, J., Maxel, M., Suri, S.: Finding the $k$ shortest simple paths: A new algorithm and its implementation. ACM Trans. Algorithms 3(4), 45 (2007)
17. Hershberger, J., Suri, S.: Erratum to "vickrey pricing and shortest paths: What is an edge worth?". In: 43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings. p. 809 (2002)
18. Katoh, N., Ibaraki, T., Mine, H.: An efficient algorithm for K shortest simple paths. Networks 12(4), 411–427 (1982)
19. Korkmaz, T., Krunz, M.: Multi-constrained optimal path selection. In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213). vol. 2, pp. 834–843 vol.2 (2001)
20. Kurz, D., Mutzel, P.: A sidetrack-based algorithm for finding the $k$ shortest simple paths in a directed graph. In: 27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia. pp. 49:1–49:13 (2016)
21. Lorenz, D.H., Raz, D.: A simple efficient approximation scheme for the restricted shortest path problem. Oper. Res. Lett. 28(5), 213–219 (2001)
22. Misra, S., Xue, G., Yang, D.: Polynomial time approximations for multi-path routing with bandwidth and delay constraints. In: INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil. pp. 558–566 (2009)
23. Roditty, L., Zwick, U.: Replacement paths and \(k\) simple shortest paths in unweighted directed graphs. In: Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings. pp. 249–260 (2005)
24. Shigang Chen, Nahrstedt, K.: On finding multi-constrained paths. In: ICC '98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No.98CH36220). vol. 2, pp. 874–879 vol.2 (1998)
25. Warburton, A.: Approximation of pareto optima in multiple-objective, shortest-path problems. Oper. Res. 35(1), 70–79 (1987)
26. Xue, G., Zhang, W., Tang, J., Thulasiraman, K.: Polynomial time approximation algorithms for multi-constrained qos routing. IEEE/ACM Trans. Netw. 16(3), 656–669 (2008)
27. Yen, J.Y.: Finding the \(k\) shortest loopless paths in a network. Management Science 17(11), 712–716 (1971)
28. Yuan, X., Liu, X.: Heuristic algorithms for multi-constrained quality of service routing. In: Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, Anchorage, Alaska, USA, April 22-26, 2001. pp. 844–853 (2001)