Second shortest simple paths in directed graphs: a crossing decomposition and a span-adaptive exact algorithm

Antonio Sedeño-Noda [a,*]

[a] *Departamento de Matemáticas, Estadística e Investigación Operativa, Universidad de La Laguna (ULL), 38271, La Laguna, Tenerife, Spain*

Abstract

We study the computation of a second shortest simple s–t path (2-SP) in a directed graph with n nodes, m arcs and nonnegative integer arc costs bounded by C. Working with reduced costs and a depth-first search that gives priority to a fixed shortest path P_{st}, every candidate second path is a prefix of P_{st}, a *detour*, and a suffix of P_{st}, and detours are classified by their re-entry arc as *forward*, *cross* or *backward*. The forward and cross classes are solved exactly in $O(m + n \log n)$ time by a single multi-source Dijkstra computation. For the backward class we introduce a *crossing decomposition*: $O(\ell/T)$ Dijkstra computations evaluate exactly a class containing every backward detour of *span* at least T, where $\ell \leq n$ is the number of nodes of P_{st}. With $T = \lceil \sqrt{\ell} \rceil$ only the *narrow core* — backward detours of span below $\sqrt{\ell}$ — remains, and it is solved exactly by a parametric negative-cycle method with batched branching whose number of rounds adapts to the instance. The complete algorithm is exact, uses $O(m)$ space, and is never asymptotically worse than the classical $O(\ell(m + n \log n))$ one-Dijkstra-per-deviation bound. Finally, under the APSP conjecture, no algorithm solves the narrow core in $O(m\sqrt{n}\,\mathrm{polylog}(nC))$ time for all polynomially bounded integer costs: the decomposition confines the known hardness of 2-SP to a small, explicitly described class of detours.

Keywords: second shortest simple path, K shortest paths, replacement paths, depth-first search, parametric negative cycles, fine-grained

*Corresponding author.
Email address: asedeno@ull.edu.es (Antonio Sedeño-Noda)

1. Introduction

Given a directed graph $G = (V, A)$ with integer arc costs and two distinguished nodes s and t, the *second shortest simple path* problem (2-SP) asks for a simple s–t path of minimum cost among all simple s–t paths different from a fixed shortest one. The problem is the elementary building block of the K shortest simple paths problem (K-SP): the K shortest simple paths can be obtained through at most $2K$ second-shortest-simple-path computations [32], so 2-SP is the primitive whose cost dominates the whole ranking. It is also closely related to the *replacement paths* problem (RP): the cost of the second path equals the minimum, over the arcs e of the shortest path, of the shortest s–t distance in $G - e$.

For undirected graphs, 2-SP is solvable in near-linear time [35]. For directed graphs, in contrast, the best exact combinatorial bounds have remained of Yen type — essentially one shortest-path computation per node of P_{st}, that is, $O(\ell(m + n \log n))$ where ℓ is the number of nodes of the shortest path — for more than fifty years, and there is a fine-grained explanation: Vassilevska Williams and Williams [38] proved that 2-SP in weighted directed graphs is *subcubically equivalent* to All-Pairs Shortest Paths (APSP), so a truly subcubic algorithm whose dependence on the largest cost C is polylogarithmic would refute the APSP conjecture.

Optimistic algorithms [16, 10, 33] match Yen's worst case with excellent typical behaviour, but their fast phase carries no coverage guarantee; this paper provides a certified counterpart. It develops a structural theory of the directed 2-SP problem that (i) isolates, in a precise and algorithmically useful sense, *where* inside the problem this fine-grained hardness lives, and (ii) turns everything outside that hard core into near-linear or $\widetilde{O}(m\sqrt{n})$ deterministic computations.[1] Our contributions are the following.

1. **Detour classification** (Section 4). After a depth-first search that gives priority to the shortest path, every detour has a unique re-entry arc into P_{st}, which is either a *forward* arc (both endpoints on the path),

[1] Throughout, $\widetilde{O}(\cdot)$ suppresses factors polylogarithmic in n and in the largest arc cost C; under the similarity assumption $C = O(n^r)$ these coincide, as then $\log(nC) = O(\log n)$.

a *cross* arc, or a *backward* arc (Figure 1). The classification comes with a key monotonicity invariant: along every arc that does not enter the path, the DFS label $\mathrm{dev}(\cdot)$ never decreases.

2. **The forward and cross classes are easy** (Section 5). The best forward detour is a plain minimum over arcs. The best cross detour is computed by *one* multi-source Dijkstra computation, because the monotonicity invariant makes the validity constraint ("return strictly after you depart") automatic for cross re-entries.
3. **The crossing decomposition** (Section 6). For backward detours the validity constraint is genuinely binding, and this is where all the difficulty of 2-SP concentrates. We prove that for any threshold g, the cheapest backward detour that departs before g and returns at or after g is computable with two multi-source Dijkstra computations (Figure 2). Consequently, $O(\ell/T)$ Dijkstra computations evaluate exactly a captured class of valid detours that contains all backward detours of span at least T (Theorem 6.3). With $T = \lceil \sqrt{\ell}\, \rceil$, the whole problem reduces in $O(\sqrt{\ell}\,(m + n \log n)) \subseteq \widetilde{O}(m\sqrt{n})$ deterministic time to backward detours of span less than $\sqrt{\ell}$: the *narrow core*.
4. **An exact parametric method for the narrow core** (Section 7). We give a two-level auxiliary graph G' (Figure 3) in which valid backward detours correspond exactly to directed cycles using one *marked* arc, an *extraction lemma* showing that every cycle with K marked arcs contains a valid detour of no larger cost, and a parametric negative-cycle search with *batched* branching: in each round the critical ratio λ^* strictly increases and each marked arc is examined at most once, by a reachability test and at most one Dijkstra computation. The complete algorithm is exact, uses $O(m)$ space, and is never asymptotically worse than the classical bound $O(\ell(m + n \log n))$; it is markedly faster when few detour families interleave — in particular it executes a single round, certified without any extra Dijkstra computation, whenever no cycle of the auxiliary graph uses two or more marked arcs (Proposition 7.7).
5. **Localization of the fine-grained barrier** (Section 9). Assuming the APSP conjecture, no $O(m\sqrt{n}\,\mathrm{polylog}(nC))$-time algorithm solves the narrow core for all polynomially bounded integer costs: otherwise, combining with items 2–3 above, 2-SP itself would be truly subcubic with polylogarithmic dependence on C, contradicting [38]. The narrow core is therefore not an artifact of our techniques, but the part of the prob-

lem where the known fine-grained barrier is necessarily concentrated within this framework.

2. Literature review

K shortest paths. When paths are allowed to repeat nodes, the ranking problem is classical [18, 31] and essentially closed: Eppstein's algorithm [9] ranks K (not necessarily simple) paths in $O(m + n \log n + K)$ time. The simple-path (loopless) version is substantially harder. The deviation framework of Yen [41] and Lawler [23] computes each new path by solving $O(\ell)$ shortest-path subproblems, giving $O(Kn(m + n \log n))$ in the worst case; for undirected graphs Katoh, Ibaraki and Mine [20] reduced the effort per path to $O(m + n \log n)$ shortest-path work. On the practical side, implementations and refinements of the deviation framework have been studied extensively [30, 36, 27, 14, 16, 29, 33] and modern practical algorithms achieve large speedups through pruning and node classification [10], sidetrack reasoning [22], time–space trade-offs [3], and the recent reformulation of K-SP as a biobjective path search [26], currently among the fastest in practice while retaining a worst case of the same order as Yen's. Other methods are faster in practice at the price of worst-case bounds exceeding Yen's [34, 11]. The asymptotically best bound for directed K-SP remains $O(K(mn + n^2 \log \log n))$ by Gotthilf and Lewenstein [13]; for nonnegative integer costs the second term is absorbed, since APSP is then solvable in $O(nm)$ time [28], giving $O(Kmn)$. Roditty and Zwick [32] gave a randomized $\widetilde{O}(Km\sqrt{n})$ algorithm for unweighted directed graphs, derandomized by Alon, Chechik and Cohen [2].

Second shortest simple path ($K = 2$). For undirected graphs it is solvable in $O(m)$ time once a shortest-path tree is available [35]. For directed graphs, the state of the art follows from the K-SP bounds above: $O(mn+n^2 \log \log n)$ deterministically [13] — or $O(nm)$ for nonnegative integer costs, using [28] — and $\widetilde{O}(m\sqrt{n})$ for unweighted graphs [32, 2]. A related but different problem is the *next-to-shortest path* (shortest among paths strictly longer than the optimum), which behaves differently and has its own literature; we do not treat it here.

Replacement paths. Since $c(\mathrm{2SP}) = \min_{e \in P_{st}} d_{G-e}(s, t)$, every RP upper bound transfers to 2-SP. For undirected graphs RP is solvable in near-linear

time [25, 15]. For directed graphs, Hershberger, Suri and Bhosle [17] proved an $\Omega(m\sqrt{n})$ lower bound in the path-comparison model, and the known upper bounds are: $\widetilde{O}(m\sqrt{n})$ for unweighted graphs (randomized [32], deterministic [2]), with a single-source extension [5]; near-linear time for planar graphs [8, 40]; $(1+\varepsilon)$-approximation in $\widetilde{O}(m/\varepsilon)$ time for general weighted digraphs [4]; and, with integer costs bounded by C, exact algebraic algorithms in $\widetilde{O}(Cn^{\omega})$ time [39, 37], which are truly subcubic for $C = O(n^r)$ with $r < 3 - \omega$. Our crossing decomposition can be read as a deterministic, purely combinatorial counterpart of the "long detours" half of these methods, with the number of path nodes skipped (the span) playing the role that detour length or random sampling plays elsewhere; crucially, it is exact for the whole wide-span class, in weighted graphs, at $O(\ell/T)$ Dijkstra cost.

Fine-grained complexity. Vassilevska Williams and Williams [38] placed 2-SP (and RP) in the subcubic equivalence class of APSP for weighted digraphs: under the APSP conjecture there is no algorithm for either problem running in $O(n^{3-\delta})\,\mathrm{polylog}(C)$ time. Our Theorem 9.1 refines this statement structurally: under the conjecture, already computing the minimum cost over backward detours of span below $\sqrt{\ell}$ is hard, while the complement is covered by an $\widetilde{O}(m\sqrt{n})$-time computation (Sections 5–6).

Parametric machinery. The narrow-core method of Section 7 uses classical tools: parametric shortest paths and minimum ratio cycles [19, 42, 1] and scaling negative-cycle detection [12]. A DFS-based classification in a related spirit was used by Könen, Schmidt and Spisla [21] for the second-best integer flow problem; the classification of the present paper concerns s–t paths, exploits the path-priority DFS order, and drives different algorithms.

3. Preliminaries and the detour reduction

Let $G = (V, A)$ be a directed graph, $|V| = n$, $|A| = m$, with integer arc costs $c_{ij} \geq 0$ for $(i,j) \in A$ and $C = \max_{(i,j)\in A} c_{ij}$.[2] The digraph is assumed simple (no parallel arcs); with parallel arcs, "the arcs of P_{st}" below

[2]Nonnegativity is assumed only for the preprocessing step; with negative costs but no negative cycles, the initial potentials can be obtained with the scaling algorithm of Goldberg [12] in $O(\sqrt{n}\,m\log(nC))$ time, and everything that follows is unchanged since it only uses reduced costs.

refers to the specific arc copies used by P_{st}. Two nodes $s, t \in V$ are given, and every node is assumed reachable from s (unreachable nodes are deleted). Throughout, $\ell \leq n$ denotes the number of nodes of the fixed shortest path, so bounds stated in terms of ℓ are never weaker than their counterparts stated in terms of n. Let π_v be the shortest distance from s to v and fix a shortest s–t path P_{st}. The *reduced cost* of an arc is $\bar{c}_{ij} = c_{ij} + \pi_i - \pi_j \geq 0$, and $\bar{c}_{ij} = 0$ on every shortest path tree arc, in particular on every arc of P_{st}. All costs below are reduced costs. Renumber the nodes so that $P_{st} = \langle 1, 2, \ldots, \ell \rangle$ with $s = 1$, $t = \ell$; the remaining nodes receive numbers larger than ℓ. Nodes $1, \ldots, \ell$ are the *path nodes*; all other nodes are *off-path*. Ties are allowed throughout: the second path is required to differ from the fixed P_{st} as a path, but it may have the same cost (detours of reduced cost zero are valid).

Definition 3.1 (Detour [32])**.** *A* valid detour *is a simple path D_{wv} from a path node w to a path node v with $w < v$, whose internal nodes are off-path and whose arcs are disjoint from the arcs of P_{st}. Its* span *is $v - w$.*

Lemma 3.2 (Single-deviation form)**.** *A second shortest simple path exists if and only if a valid detour exists, and $c(\mathrm{2SP}) = \pi_t + \min\{\bar{c}(D) : D \text{ valid detour}\}$. Moreover, an optimal second path of the form $P_{sw} \circ D_{wv} \circ P_{vt}$ always exists.*

Proof. ($\leq$) For a valid detour D_{wv}, the concatenation $P_{sw} \circ D_{wv} \circ P_{vt}$ is a simple s–t path (the pieces are internally disjoint because $w < v$ and the internal nodes of D are off-path), it differs from P_{st} (its first arc out of w is not the path arc, since detour arcs avoid the arcs of P_{st}), and its original cost is $\pi_t + \bar{c}(D)$.

($\geq$) Let $Q \neq P_{st}$ be a simple s–t path of minimum cost. Follow Q from s while it coincides with P_{st} and let w be the node at which Q first uses an arc different from the path arc; w exists because $Q \neq P_{st}$ and both paths end at t. Let v be the first node of Q strictly after w lying on P_{st} (v exists because $t \in P_{st}$). If $v \leq w$ then v occurs on the common prefix $P_{sw} \subseteq Q$, contradicting simplicity of Q; hence $v > w$, and the segment Q_{wv} is a valid detour: its internal nodes avoid P_{st} by the choice of v, its first arc is not a path arc, and no arc of Q_{wv} is an arc of P_{st} (internal arcs have off-path endpoints; the first and last arcs leave w, respectively enter v, by non-path arcs). Since reduced costs are nonnegative and the common prefix has reduced cost 0, $c(Q) = \pi_t + \bar{c}(Q) \geq \pi_t + \bar{c}(Q_{wv}) \geq \pi_t + \min_D \bar{c}(D)$. □

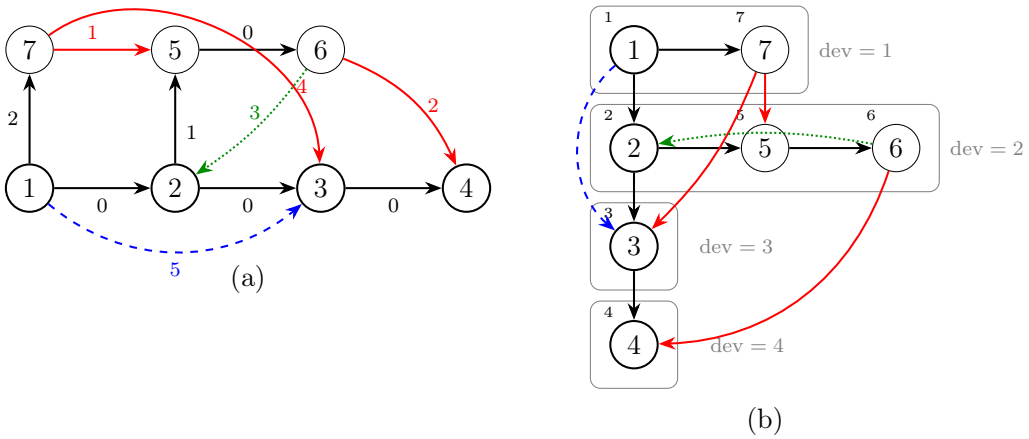

Figure 1: (a) An example with $P_{st} = \langle 1, 2, 3, 4\rangle$ (reduced costs on the arcs; path arcs cost 0). (b) The DFS tree drawn by dev-levels; the small number next to each node is its discovery order. Tree arcs are solid black, the forward arc $(1, 3) \in F$ is blue dashed, the backward arc $(6, 2) \in B$ is green dotted, and cross arcs are red: $(6, 4), (7, 3) \in CR$ enter the path, while $(7, 5)$ joins two off-path nodes and illustrates monotonicity ($\text{dev}(7) = 1 \leq \text{dev}(5) = 2$). The best detour here is $2 \to 5 \to 6 \to 4$, of reduced cost 3, found by the cross-case Dijkstra of Theorem 5.2.

By Lemma 3.2 the problem is to find a minimum reduced-cost valid detour.

4. Path-priority DFS, classification, and monotonicity

Perform a depth-first search from $s = 1$ in which the adjacency list of every path node $i < \ell$ is reordered so that the path successor $i+1$ is explored first. Then the first ℓ nodes discovered are $1, 2, \ldots, \ell$ in this order, so $\text{order}_i = i$ for $i \leq \ell$, and the DFS tree $\mathcal{T}$ contains P_{st}. As usual [6], the non-tree arcs are partitioned into *forward* arcs (head is a proper descendant of the tail), *backward* arcs (head is an ancestor), and *cross* arcs (all others; a cross arc (u, v) always satisfies $\text{order}_u > \text{order}_v$).

For every node v let $\text{dev}(v)$ be the last path node on the tree path from s to v; thus $\text{dev}(i) = i$ for path nodes, and an off-path node v *hangs* from the path at node $\text{dev}(v)$. Note that the tree ancestors of a path node i are exactly $1, \ldots, i-1$, all on the path. Figure 1 shows a complete example.

Proposition 4.1 (No cross arcs leave the path)**.** *Every arc leaving a path node is a tree arc, a forward arc, or a backward arc whose head is a path node.*

Proof. When path node i is discovered, the discovered nodes are exactly its tree ancestors $1, \ldots, i-1$. When the search later scans a non-first arc (i, x): if x is undiscovered, the arc becomes a tree arc; if x is discovered and unfinished, x is an ancestor of i, hence a path node, and the arc is backward; if x is finished, then x was discovered after i, so x lies in the already-explored part of the subtree of i, and the arc is forward. □

Definition 4.2 (Arc classes and the off-path graph)**.** *Let F be the set of forward arcs with both endpoints on the path, and let B (respectively CR) be the set of backward (respectively cross) arcs (u, v) with u off-path and v on the path. The* off-path graph G_{off} *consists of all arcs of G whose head is off-path; equivalently, G_{off} is obtained from G by deleting the arcs of P_{st} and every arc entering a path node. Thus G_{off} contains no arc between path nodes and no arc entering a path node, while tails may be path nodes: arcs of G_{off} whose tail is a path node are called* departure arcs*.*

Lemma 4.3 (Re-entry classification)**.** (i) *For $(u, v) \in B$ we have $v \leq \text{dev}(u)$; for $(u, v) \in CR$ we have $\text{dev}(u) < v$.* (ii) *Every arc entering a path node $v \neq s$ from an off-path node belongs to $B \cup CR$.* (iii) *Every valid detour is either a single arc of F, or a walk consisting of one departure arc, internal arcs of G_{off}, and a final arc of $B \cup CR$.*

Proof. (i) If (u, v) is backward, v is a tree ancestor of u, so v lies on the tree path from s to u and $v \leq \text{dev}(u)$. If (u, v) is cross and $v \leq \text{dev}(u)$, then v lies on the tree path from s to $\text{dev}(u)$, hence is an ancestor of u and the arc would be backward, a contradiction. (ii) The tree arc into v is the path arc $(v-1, v)$; a forward arc into v would make its tail an ancestor of v, but all ancestors of v are path nodes. Hence an arc into v from an off-path node is backward or cross. (iii) A single-arc detour (w, v) with $w < v$ on the path is not a path arc, is not backward (backward path-to-path arcs have head smaller than tail) and is not cross (Proposition 4.1); hence it is in F. A longer detour has all internal nodes off-path: its first arc is a departure arc, its internal arcs lie in G_{off}, and its last arc enters the path from an off-path node, so it lies in $B \cup CR$ by (ii). □

Remark 4.4 (Discarded and unusable arcs). Backward arcs with both endpoints on the path belong to none of F, B, CR, G_{off}: Definition 4.2 discards them altogether. This is safe: by part (iii) a valid detour is a single forward arc, or a departure arc and internal arcs of G_{off} (whose heads are off-path) followed by one arc of $B \cup CR$ (whose tail is off-path), and a path-to-path backward arc — path tail, path head, and head preceding tail — fits none of these slots. Similarly, the arcs of B entering s (note that s is an ancestor of every node) can close no valid detour, since every re-entry satisfies $v > w \geq 1 = s$; consistently, the auxiliary graph of Section 7 provides arrival copies only for $v \geq 2$, so these arcs are dropped there as well.

The engine of everything that follows is the monotonicity property below (Lemma 4.6). We first isolate the fact about the path-priority DFS on which its cross-arc case rests.

Lemma 4.5 (Off-path discovery order)**.** *Let $e < f$ be path nodes. Every off-path node with* dev*-value f is discovered before every off-path node with* dev*-value e.*

Proof. Let $d(\cdot)$ and $\varphi(\cdot)$ denote DFS discovery and finishing times, so that for a tree descendant z of a node c the intervals nest: $d(c) < d(z) < \varphi(z) \leq \varphi(c)$ [6]. In the path-priority DFS the tree parent of each path node $i > 1$ is $i-1$, so the path nodes form a chain in $\mathcal{T}$ and every path node $j > e$ is a tree descendant of $e+1$. Let z be off-path with $\text{dev}(z) = f > e$: by the definition of dev, z is a tree descendant of the path node f, hence of $e+1$, and interval nesting gives $d(e+1) < d(z) < \varphi(z) \leq \varphi(e+1)$. Let now y be off-path with $\text{dev}(y) = e$: by the definition of dev, y is a tree descendant of some non-path child c of e; since $e+1$ is the first child of e explored, c is scanned only after the subtree of $e+1$ is finished, so $d(y) \geq d(c) > \varphi(e+1) \geq \varphi(z) > d(z)$. Thus z is discovered, indeed finished, before y is discovered. □

Lemma 4.6 (Monotonicity)**.** *Every arc (x, y) of G_{off} satisfies* $\text{dev}(x) \leq \text{dev}(y)$*. Consequently, every walk in G_{off} starting at a path node w visits nodes of nondecreasing* dev*-value, all at least w; in particular, if the walk ends at u then $w \leq \text{dev}(u)$.*

Proof. Let (x, y) be an arc of G_{off} (y off-path) and consider its DFS class. Tree arc: the tree path to y extends the one to x, and y is off-path, so $\text{dev}(y) \geq \text{dev}(x)$ (with equality unless x is a path node, in which case $\text{dev}(y) = x = \text{dev}(x)$). Forward arc: y is a descendant of x, the tree path to y

passes through x, hence $\mathrm{dev}(y) \geq \mathrm{dev}(x)$. Backward arc: y is an off-path ancestor of x, so x and y lie in the same hanging subtree and $\mathrm{dev}(x) = \mathrm{dev}(y)$. Cross arc: then $\mathrm{order}_x > \mathrm{order}_y$, and x is off-path because no cross arcs leave the path (Proposition 4.1); if we had $\mathrm{dev}(x) > \mathrm{dev}(y)$, Lemma 4.5 would give $\mathrm{order}_x < \mathrm{order}_y$, a contradiction, so $\mathrm{dev}(x) \leq \mathrm{dev}(y)$. The statement about walks follows by induction along the walk, the first arc being a tree or forward departure from w with dev-value at least w. □

5. The forward and cross classes in near-linear time

Proposition 5.1 (Forward detours)**.** *The minimum reduced cost of a valid detour consisting of a single forward arc is* $\lambda_F = \min_{(u,v)\in F} \bar{c}_{uv}$*, computable in* $O(m)$ *time.*

Proof. Forward path-to-path arcs (u, v) have $v > u$, so each is by itself a valid detour, and by Lemma 4.3(iii) these are the only single-arc detours. □

Theorem 5.2 (Cross detours by one Dijkstra)**.** *For every node* x *let* $D(x) = \min_{w\in\{1,\ldots,\ell-1\}} \mathrm{dist}_{G_{\mathrm{off}}}(w \to x)$*, computable by a single multi-source Dijkstra computation with the path nodes* $1, \ldots, \ell - 1$ *as zero-cost sources. Then the minimum reduced cost of a valid detour whose last arc is in* CR *equals* $\lambda_{CR} = \min_{(u,v)\in CR}\big(D(u) + \bar{c}_{uv}\big)$*, computed in* $O(m + n \log n)$ *total time.*

Proof. ($\geq$) A valid detour with last arc $(u, v) \in CR$ departs at some $w < v$ and reaches u through G_{off}; its cost is at least $D(u) + \bar{c}_{uv}$. ($\leq$) Let the minimum $D(u)$ be realized by a walk from a path node w. By Lemma 4.6, $w \leq \mathrm{dev}(u)$, and by Lemma 4.3(i), $\mathrm{dev}(u) < v$; hence $w < v$, and appending (u, v) yields a walk from w to v that never touches the path in between (G_{off} has no arcs entering path nodes). Since the walk realizes the minimum $D(u)$ and reduced costs are nonnegative, every cycle on it has zero reduced cost; removing these zero-cost cycles extracts a simple valid detour of the same cost. □

Remark 5.3. Theorem 5.2 shows that for cross re-entries the constraint $w < v$ is *automatically satisfied* by reachability in G_{off}: any walk of G_{off} reaching u must have departed at some node $w \leq \mathrm{dev}(u)$, while every cross re-entry from u returns at some node $v > \mathrm{dev}(u)$. For backward re-entries the situation is reversed: $(u, v) \in B$ has $v \leq \mathrm{dev}(u)$, so a cheapest walk into u may depart at some $w \geq v$ and be useless. Within this decomposition, all the difficulty of 2-SP is concentrated in the backward class.

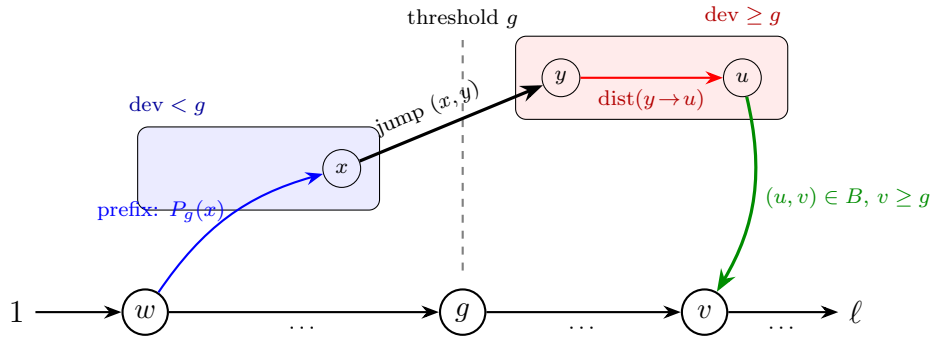

Figure 2: The crossing decomposition (Lemma 6.2). A valid backward detour departing at $w < g$ and returning at $v \geq g$ must cross the threshold g exactly once, at its unique jump arc (x, y) with $\mathrm{dev}(x) < g \leq \mathrm{dev}(y)$. The prefix lives at dev-levels below g (its departure node satisfies $w \leq \mathrm{dev}(x) < g$ automatically, by monotonicity), the suffix at levels $\geq g$. Minimizing the three parts independently is therefore exact: two multi-source Dijkstra computations per threshold. A detour of span $v - w \geq T$ crosses some threshold of the grid $\{T, 2T, 3T, \dots\}$.

6. Backward detours I: the crossing decomposition

The forward and cross classes having been solved exactly in Section 5, the only detours left are those whose last arc lies in B.[3] Below, *crossing* refers to a walk crossing a threshold level g along the path, not to a DFS cross arc.

Definition 6.1 (Jump arcs and crossing arcs)**.** *An arc (x, y) of G_{off} is a* jump arc *if* $\mathrm{dev}(x) < \mathrm{dev}(y)$*. Given a walk W in G_{off} from a path node w to a node u, and a threshold g with $w < g \leq \mathrm{dev}(u)$, the* crossing arc of W at g *is the first arc (x, y) of W with $\mathrm{dev}(x) < g \leq \mathrm{dev}(y)$. It exists and is unique: by Lemma 4.6 the* dev*-values along W are nondecreasing, start below g and end at or above g, and once level g is reached it is never left downwards.*

Lemma 6.2 (Crossing lemma)**.** *Fix a threshold $g \in \{2, \dots, \ell\}$ and compute:*

[3] The two-regime split at a $\sqrt{\cdot}$ threshold is partly inspired by the long/short-detour paradigm of Roditty and Zwick [32]. It differs in four respects: it splits by *span* (index distance along P_{st}) rather than by number of arcs, it is *deterministic* rather than randomized, it applies to *weighted* graphs, and it is driven by the dev-monotonicity of the path-priority DFS (Lemma 4.6).

1. $P_g(x)$ *for every* x *with* $\mathrm{dev}(x) < g$*: multi-source Dijkstra distances in* G_{off} *from the source set* $\{w \in P_{st} : w < g\}$ *(zero-cost sources); only these values are used in item 2;*
2. $Q_g(z)$ *for every* z *with* $\mathrm{dev}(z) \geq g$*: multi-source Dijkstra distances in* G_{off}*, where the sources are the heads* y *of the jump arcs* (x, y) *with* $\mathrm{dev}(x) < g \leq \mathrm{dev}(y)$ *and* $P_g(x) < \infty$*, and each source* y *receives as offset the* minimum *of* $P_g(x) + \bar{c}_{xy}$ *over its incoming jump arcs. By Lemma 4.6 every node reachable from these sources satisfies* $\mathrm{dev}(z) \geq g$*, so this computation may be confined to the subgraph of* G_{off} *induced by* $\{z : \mathrm{dev}(z) \geq g\}$*;*
3. $\mathrm{ans}_g = \min\{Q_g(u) + \bar{c}_{uv} : (u, v) \in B,\ v \geq g\}$.

Then ans_g *equals the minimum reduced cost of a valid backward detour with* $w < g \leq v$*, and the predecessor arrays of the two Dijkstra computations provide a witness detour attaining it. The computation takes* $O(m + n \log n)$ *time per threshold.*

Proof. ($\geq$) Let D be a valid backward detour with departure $w < g$ and return arc $(u, v) \in B$, $v \geq g$. Since $g \leq v \leq \mathrm{dev}(u)$ by Lemma 4.3(i), the walk of D has a crossing arc (x, y) at g. Its prefix from w to x costs at least $P_g(x)$ (as $w < g$, w is a source), and its suffix from y to u costs at least the G_{off}-distance from y to u; hence $\bar{c}(D) \geq P_g(x) + \bar{c}_{xy} + \mathrm{dist}(y \to u) + \bar{c}_{uv} \geq Q_g(u) + \bar{c}_{uv} \geq \mathrm{ans}_g$.

($\leq$) Conversely, ans_g is realized by a walk from some source $w' < g$ to x (with $w' \leq \mathrm{dev}(x) < g$ by Lemma 4.6), the jump arc (x, y), a walk from y to u, and the return arc (u, v) with $v \geq g$. The concatenation departs at $w' < g \leq v$ and satisfies $v \leq \mathrm{dev}(u)$, so it is a valid detour walk; extracting a simple detour cannot increase the cost. Hence ans_g is the exact minimum over the stated class. □

Theorem 6.3 (Grid theorem: wide backward detours)**.** *Let* $T \geq 1$ *and* $\mathcal{G}_T = \{T, 2T, 3T, \dots\} \cap \{2, \dots, \ell\}$*. Then* $\min_{g \in \mathcal{G}_T} \mathrm{ans}_g$ *is the exact minimum reduced cost over all valid backward detours* captured *by some* $g \in \mathcal{G}_T$ *(i.e. with* $w < g \leq v$*), a class containing every valid backward detour of span at least* T*. The total running time is* $O\big((\ell/T)(m + n \log n)\big)$.

Proof. If $v - w \geq T$ then the interval $(w, v]$ contains a multiple of T and the detour is captured at that threshold; Lemma 6.2 evaluates each captured class exactly, and every value ans_g is itself the cost of a valid detour, so the minimum never underestimates. □

Remark 6.4. With $T = 1$ every valid backward detour is captured, and Theorem 6.3 becomes a two-Dijkstras-per-threshold variant of the classical one-shortest-path-per-deviation method, with the same $O(\ell(m + n \log n))$ cost. The grid theorem therefore strictly generalizes the classical approach: the parameter T trades coverage of narrow spans for running time.

Corollary 6.5 (Span-adaptive reduction)**.** *With $T = \lceil \sqrt{\ell} \rceil$, one obtains in $O(\sqrt{\ell}\,(m + n \log n)) \subseteq \widetilde{O}(m\sqrt{n})$ deterministic time a value U_1 equal to the exact minimum reduced cost over a* captured class *of valid detours that contains all forward detours, all cross detours, and every valid backward detour of span at least $\sqrt{\ell}$ (it may also contain narrower backward detours that happen to cross a grid threshold; these are valid as well, so U_1 never underestimates any valid detour it reports). Consequently, writing $N_B = \min\{\bar{c}(D) : D$ valid backward detour$\}$ for the exact backward optimum, $c(\mathrm{2SP}) = \pi_t + \min(U_1, N_B)$: every valid detour is either captured or backward, every backward detour of span at least $\sqrt{\ell}$ is captured — so U_1 already accounts for it — and both minima are attained by valid detours. Hence only backward detours of span below $\sqrt{\ell}$ — the* narrow core *— can improve on U_1.*

7. Backward detours II: an exact parametric procedure for the residual class

The goal of this section is to compute exactly the minimum reduced cost N_B over all valid backward detours; by Corollary 6.5 the algorithm only needs $\min(U_1, N_B)$, and the incumbent U_1 is used to prune the search. The procedure below therefore operates on *all* backward detours, and the pruning ensures that rounds are triggered only by cycles of ratio below U_1. Every *individual* detour cheaper than U_1 is necessarily narrow after the grid phase (Corollary 6.5).

7.1. The two-level auxiliary graph

Definition 7.1 (Auxiliary graph G')**.** *The node set of G' consists of all off-path nodes, a* departure copy *i' for each path node $i \in \{1, \dots, \ell - 1\}$, and an* arrival copy *$i''$ for each $i \in \{2, \dots, \ell\}$. The arcs of G' are (see Figure 3): all arcs of G_{off} between off-path nodes, with their reduced costs; a departure arc (i', x) with cost $\bar{c}_{ix}$ for every arc (i, x) of G with x off-path; a return arc (u, v'') with cost $\bar{c}_{uv}$ for every $(u, v) \in B$; the* rail *arcs $(i'', (i-1)'')$ for $i = 3, \dots, \ell$,*

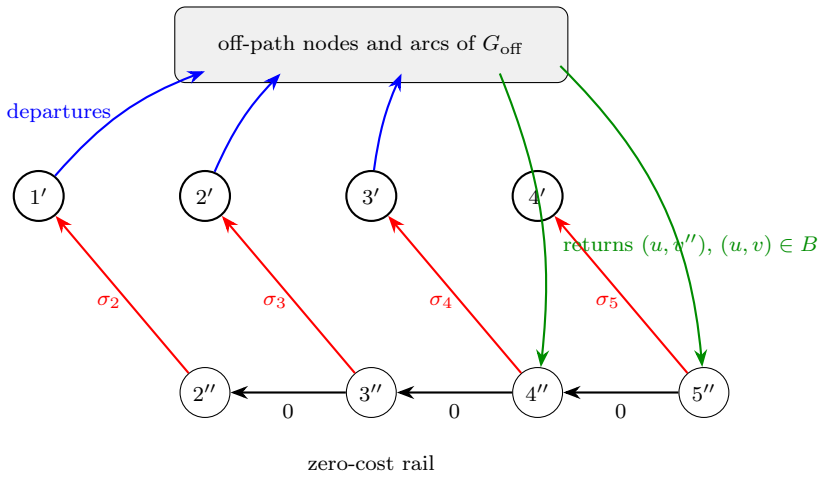

Figure 3: The auxiliary graph G' for $\ell = 5$ (Definition 7.1). A valid backward detour departing at w with return (u, v) corresponds to the cycle: marked arc σ_{w+1}, departure from w', walk through G_{off}, return into v'', rail from v'' down to $(w+1)''$. The rail only descends, so a cycle using the single marked arc σ_i can close only if $v \geq i$ — which is exactly the validity constraint $w < v$ (Lemma 7.2). Cycles using several marked arcs are handled by the extraction lemma and the batched parametric search.

with cost 0*; and the* marked *arcs* $\sigma_i = (i'', (i-1)')$ *for* $i = 2, \ldots, \ell$*, with cost* 0*. Thus* G' *has at most* $n + 2\ell$ *nodes and* $m + 2(\ell - 1)$ *arcs; the only arcs leaving an arrival copy* i'' *are its rail arc and* σ_i*, and the only arcs entering a departure copy are marked arcs.*

Lemma 7.2 (Correspondence)**.** *Directed cycles of* G' *using exactly one marked arc* σ_i *are in cost-preserving correspondence with valid backward detours departing at node* $i - 1$*.*

Proof. Given such a cycle: after σ_i it stands at $(i-1)'$; departure copies have only departure arcs and off-path nodes have only G_{off}-arcs and return arcs, so the cycle proceeds through G_{off} until a return arc puts it at some arrival copy v''; from arrival copies only the rail is available (σ_i being the only marked arc used), and the rail decreases the index by one at each step, so the cycle closes at i'' only if $v \geq i$. The non-rail, non-marked part is then a walk from path node $i-1$ through G_{off} ending with a return $(u, v) \in B$, $v \geq i > i-1$: a valid detour walk of the same cost (rail and marked arcs cost 0); extract a simple detour if necessary. Conversely, a valid detour departing at w with return (u, v) lifts to the cycle through σ_{w+1}, the corresponding

departure, off-path and return arcs, and the rail from v'' down to $(w+1)''$, with the same reduced cost. □

Lemma 7.3 (Extraction)**.** *Every directed cycle W of G' containing $K \geq 1$ marked arcs contains a segment which is a valid backward detour walk of reduced cost at most $\bar{c}(W)$ (costs taken without parametric modification).*

Proof. Traverse W cyclically and let $\sigma_{q_1}, \ldots, \sigma_{q_K}$ be its marked arcs in traversal order, with indices read cyclically ($q_{K+1} = q_1$), so that all three sums below telescope around the cycle. After σ_{q_j} the cycle stands at $(q_j - 1)'$; let b_j'' be the next arrival copy reached (it exists, since the tail q_{j+1}'' of the next marked arc is an arrival copy). As in Lemma 7.2, the segment S_j from $(q_j - 1)'$ to b_j'' consists of a departure, G_{off}-arcs and one return arc, and from b_j'' the cycle follows the rail down to q_{j+1}'', which forces $b_j \geq q_{j+1}$. Summing the index changes around the cycle,

$$\sum_{j=1}^{K} \big(b_j - (q_j - 1)\big) + \sum_{j=1}^{K} \big(q_{j+1} - b_j\big) + \sum_{j=1}^{K} (-1) = 0,$$

so $\sum_j (b_j - q_j + 1) = K + \sum_j (b_j - q_{j+1}) \geq K \geq 1$, and some j has $b_j \geq q_j > q_j - 1$: the segment S_j is a valid detour walk (departure $q_j - 1$, return b_j). Rail and marked arcs cost 0 and all other arcs are nonnegative, so $\bar{c}(S_j) \leq \bar{c}(W)$. □

7.2. The parametric oracle and the exact ratio search

For $\lambda \geq 0$ let c^λ subtract λ on every marked arc and leave all other costs unchanged. Cycles without marked arcs lie in G_{off} (the rail is acyclic) and have nonnegative cost for every λ; therefore

$$G'^{\lambda} \text{ has no negative cycle } \iff \lambda \leq \lambda^* := \min_W \frac{\bar{c}(W)}{K(W)},$$

the minimum ranging over directed cycles with $K(W) \geq 1$ marked arcs. (If no such cycle exists, $N_B = \infty$; this is detected in $O(m)$ time by checking with strongly connected components whether some marked arc lies on a cycle.) Lemma 7.2 gives $\lambda^* \leq N_B$, and Lemma 7.3 converts any cycle of ratio λ into a valid detour of cost at most $K\lambda$.

Range and granularity. Every potential satisfies $0 \leq \pi_v \leq (n-1)C$, so every reduced cost is at most $\bar{C} := nC$; a simple cycle of G' has fewer

than $n + 2\ell \le 3n$ arcs, so every cycle ratio with $K \ge 1$ lies in $[0, U_2]$ with $U_2 := 3n\bar{C} = 3n^2C$. (The value $\lambda^* = 0$ — a tie — is handled like any other.) Moreover λ^* is a rational p/q with $1 \le q \le \ell - 1$, and two distinct such ratios differ by at least $1/(\ell-1)^2$, since $|p/q - p'/q'| \ge 1/(qq')$ when they differ.

Probes. The binary search maintains an interval $[L, H]$, initialized to $L = \lambda_{\text{prev}}$ ($= 0$ before the first round) and $H = \min(U, U_2)$. A probe at a rational $\lambda = a/b$ with $b \le \ell - 1$ replaces the costs by the integers $b\,\bar{c}_{xy} - a$ on marked arcs and $b\,\bar{c}_{xy}$ elsewhere — magnitudes $O(n^3C)$ — and runs one negative-cycle detection with the scaling algorithm of Goldberg [12], which handles the negative marked arcs, in $O(\sqrt{n}\,m\log(nC))$ time; when feasible, the same computation returns shortest-path distances from an auxiliary source, which serve as the potentials.

Recovery and cost. After $O(\log(U_2(\ell-1)^2)) = O(\log(nC))$ probes the interval is narrower than $1/(\ell-1)^2$ and contains a unique candidate ratio, recovered by the continued-fraction method; a final run at λ^* returns potentials d with $c^{\lambda^*}_{xy} + d_x - d_y \ge 0$ on every arc. This search-and-round scheme is classical [1, 19, 42]. In total, *one exact evaluation of λ^* costs* $O(\sqrt{n}\,m\log^2(nC))$ — the only quantity from this subsection used below.

7.3. Batched branching

Given exact λ^* and potentials d, the *tight subgraph* consists of the arcs with $c^{\lambda^*}_{xy} + d_x - d_y = 0$.

Lemma 7.4. *Every cycle of ratio exactly λ^* lies entirely in the tight subgraph.*

Proof. For such a cycle W, $\sum_{(x,y)\in W}(c^{\lambda^*}_{xy} + d_x - d_y) = \bar{c}(W) - K\lambda^* = 0$, and every summand is nonnegative, hence zero. □

Algorithm NarrowCore(G', U) (incumbent $U = U_1$)

1. While some marked arc lies on a cycle of G' (SCC test, $O(m)$):
   (a) If $U < \infty$ and the probe at $\lambda = U$ is feasible (no negative cycle), **stop**. Otherwise run the exact ratio search of Section 7.2 on the interval $[\lambda_{\text{prev}}, \min(U, U_2)]$, with $\lambda_{\text{prev}} = 0$ initially and $U_2 = 3n^2C$ the a priori bound on every cycle ratio.
   (b) Compute the tight subgraph and its strongly connected components; let $R = \{\sigma_i$ lying in a nontrivial SCC$\}$. ($R \neq \emptyset$: a cycle attaining λ^* is tight by Lemma 7.4 and contains marked arcs.)
   (c) (*certification*) For each $\sigma_i \in R$: one graph search, $O(m)$, testing whether the tight subgraph minus the other marked arcs contains a path from $(i-1)'$ to i'' — that is, whether some cycle attaining λ^* uses σ_i alone. If some test succeeds, set $U \leftarrow \min(U, \lambda^*)$, record the witness (σ_i, d) with d the potentials of this round, and **stop**.
   (d) (*resolution*) Otherwise, for each $\sigma_i \in R$: one Dijkstra computation from $(i-1)'$ in G' minus all marked arcs; the distance to i'' is the exact optimum over detours departing at $i-1$ (Lemma 7.2); if this distance is smaller than U, update $U \leftarrow$ this distance and record the witness (σ_i, tree), the Dijkstra predecessor tree of this computation.
   (e) Delete the arcs of R from G'; set $\lambda_{\text{prev}} \leftarrow \lambda^*$.
2. Return U together with the recorded witness σ_i (the departure position $i-1$ of an optimal backward detour) and the potentials d (equivalently the Dijkstra tree), from which the optimal detour is reconstructed: a zero-reduced-cost $(i-1)' \to i''$ path in the tight subgraph avoiding the other marked arcs.

Theorem 7.5 (Narrow core: correctness and complexity)**.** *NarrowCore returns* $\min(U_1, N_B)$ *exactly, together with a witness identifying an optimal backward detour, in* $O(m)$ *space and*

$$O\big(Q\,\sqrt{n}\,m\log^2(nC)\;+\;\varrho\,(m + n\log n)\big)\ \textit{time},$$

where Q *is the number of rounds executed and* $\varrho \le \ell - 1$ *the number of marked arcs resolved in steps (c)–(d); the critical ratio* λ^* *increases strictly*

from round to round. With the guard of Remark 7.6 this is $O(\ell(m + n \log n))$ unconditionally — matching the classical deviation bound — and it is far smaller when few rounds occur: in the favorable case of Proposition 7.7 a single round suffices, giving $O(\sqrt{n}\, m \log^2(nC) + |R_1|\, m)$.

Proof. Correctness. Two facts hold throughout the execution. First, U starts at U_1, only decreases, and every value assigned to it is the cost of a genuine valid backward detour; hence at all times $U \geq \min(U_1, N_B)$. Second, the *resolution invariant*: for every deleted σ_i, step (d) computed, before the deletion, the exact optimum over detours departing at $i-1$ and folded it into U — the Dijkstra graph excludes marked arcs, so its $(i-1)' \to i''$ paths are exactly the bodies of the one-marked-arc cycles of Lemma 7.2. The procedure terminates in exactly one of three ways:

(T1) *the loop test fails*: no marked arc lies on a cycle; then no surviving departure admits any detour, since a detour departing at $i-1$ closes a cycle through σ_i;

(T2) *the probe at $\lambda = U$ succeeds in step (a)* (only possible when $U < \infty$): every remaining cycle has ratio at least U; in particular every surviving detour — a one-marked-arc cycle, whose cost equals its ratio — costs at least U;

(T3) *a certification succeeds in step (c)*: the test exhibits a tight one-marked-arc cycle through σ_i, i.e. a detour of cost exactly λ^*, and $U = \lambda^*$ after the update (the round ran because $\lambda^* < U$); as in (T2), every surviving detour costs at least its ratio, which is at least $\lambda^* = U$.

In each case, an optimal valid backward detour either departs at a deleted position — where the resolution invariant gives $U \leq N_B$ — or at a surviving one, where its cost is at least the returned U (T2, T3) or no such detour exists (T1). Combined with $U \geq \min(U_1, N_B)$, the returned value is exactly $\min(U_1, N_B)$. In particular, if any valid backward detour exists, its departure position is eventually deleted or the exit is (T2)/(T3), so the returned U is finite.

Adaptivity. A round starts only if the probe at $\lambda = U$ fails, i.e. only while $\lambda^* < U$. And the critical ratio increases strictly: deleting arcs cannot decrease it, and if after deleting R some remaining cycle still had ratio λ^*, it would be tight (Lemma 7.4; the potentials remain valid on the smaller graph), so its marked arcs would have been deleted with R — a contradiction.

Complexity. A round costs one ratio search, $O(\sqrt{n}\, m \log^2(nC))$, plus at most $|R|$ reachability tests and $|R|$ Dijkstra computations, $O(m + n \log n)$ each. The batches R are nonempty and disjoint across rounds, so the steps (c)–(d) cost $O(\ell(m + n \log n))$ in total, over all rounds, unconditionally; only the ratio searches can push the total above the classical bound, and the guard of Remark 7.6 caps them at it. This gives the unconditional $O(\ell(m+n \log n))$ bound. The number of rounds is finite — indeed at most $\lceil(\ell-1)/2\rceil+1$, since every non-final round processes $|R| \geq 2$ marked arcs (it finds no one-marked-arc cycle at λ^*, while a simple attaining cycle exists and is thus multi-marked, contributing ≥ 2 marked arcs to R) — but the guard, not this count, is what bounds the running time. The single-round bound is Proposition 7.7. □

Remark 7.6 (Guard and unconditional bound). Only the ratio searches can exceed the classical budget $O(\ell(m + n \log n))$; the work of steps (c)–(d) never does (Theorem 7.5). The implementation therefore caps them: if their cumulative cost exceeds a fixed multiple of $\ell(m + n \log n)$, the parametric phase is aborted and every remaining marked arc is resolved directly by step (d)'s Dijkstra. This preserves exactness and yields the *unconditional* bound $O(\ell(m+n \log n))$ — the classical deviation cost — while retaining the round-adaptive speedup of Theorem 7.5. Section 9 explains why a worst-case polylogarithmic bound on the number of rounds should not be sought.

Proposition 7.7 (Certified early termination)**.** *If in some round the current minimum cycle ratio λ^* is attained by a cycle with exactly one marked arc, that round is the last: the certification step (c) succeeds, no Dijkstra computation runs in that round, and the procedure stops with the exact answer. In particular, if this holds for the* initial *G' — as happens, for instance, whenever G' contains no directed cycle with two or more marked arcs — then* NarrowCore *executes a single round and runs in $O\big(\sqrt{n}\, m \log^2(nC) + |R_1|\, m\big)$ time, where R_1 is the first batch.*

Proof. If G' has no cycle through a marked arc, the initial test stops the procedure at once; if $\lambda^* \geq U$, the probe at $\lambda = U$ is feasible and no round runs. Otherwise let W attain λ^* with a single marked arc σ_i: by Lemma 7.4 W is tight, so $\sigma_i \in R$ and the body of W is a path from $(i-1)'$ to i'' in the tight subgraph avoiding the other marked arcs. The test of step (c) therefore succeeds — at σ_i at the latest — and the procedure stops with $\min(U, \lambda^*)$, which is the exact answer by case (T3) of Theorem 7.5. In the initial-round

form the executed work is one exact ratio search, the SCC computations, and at most $|R_1|$ graph searches of $O(m)$ each. □

Remark 7.8. Proposition 7.7 identifies the *only* source of additional rounds: minimum ratios attained exclusively by multi-marked-arc cycles, that is, by interleaved families of backward detours. The procedure runs precisely until the first round whose critical ratio is attained by a one-marked-arc cycle, the probe at U succeeds, or the marked arcs are exhausted. Two cautions. First, the weaker hypothesis "no backward detour improves on U_1" does *not* suffice for one-probe termination: a composite cycle with K marked arcs and cost just below $K\,U_1$ is negative at $\lambda = U_1$ even if every individual detour costs at least U_1; the hypothesis on the *ratio* is the correct one. Second, an implementation that omits step (c) can recover the same early stop inside step (d) — halt when some Dijkstra returns exactly λ^*; the two triggers are equivalent by Lemma 7.4 — but certification is cheaper: a reachability search instead of a Dijkstra computation, and no Dijkstra at all in the stopping round.

8. The complete algorithm

Algorithm SECONDPATH(G, s, t)

1. Shortest path tree from s, potentials π, reduced costs; fix $P_{st} = \langle 1, \ldots, \ell \rangle$; path-priority DFS; labels order, dev; classes F, CR, B; off-path graph G_{off}. $O(m + n \log n)$
2. $U \leftarrow \lambda_F$ (Proposition 5.1); $U \leftarrow \min(U, \lambda_{CR})$ (Theorem 5.2). $O(m + n \log n)$
3. Grid phase with $T = \lceil \sqrt{\ell} \rceil$: $U \leftarrow \min\big(U, \min_{g \in \mathcal{G}_T} \text{ans}_g\big)$ (Theorem 6.3). $O(\sqrt{\ell}\,(m + n \log n))$
4. $U \leftarrow$ NARROWCORE(G', U) with the guard of Remark 7.6. see Theorem 7.5
5. If $U = \infty$, report that no second path exists; otherwise return the second path $P_{sw} \circ D \circ P_{vt}$ assembled from the argmin detour (all subroutines return predecessor information), of cost $\pi_t + U$.

Theorem 8.1 (Main)**.** *SECONDPATH computes a second shortest simple path exactly, in $O(m)$ space and $O(\ell(m + n \log n))$ time — never asymptotically worse than the classical one-Dijkstra-per-deviation bound. Steps 1–3*

(the forward, cross and wide backward classes) run in $O(\sqrt{\ell}\,(m + n\log n)) \subseteq \widetilde{O}(m\sqrt{n})$ *deterministic time; only the narrow core (step 4) can require more, and it is faster when few rounds occur (Theorem 7.5), costing* $O(\sqrt{n}\,m\log^2(nC) + |R_1|\,m)$ *in the single-round case of Proposition 7.7.*

Proof. Lemma 3.2 reduces 2-SP to the minimum valid detour; Lemma 4.3(iii) partitions valid detours into the classes F, CR, B; Proposition 5.1 and Theorem 5.2 handle F and CR exactly; Theorems 6.3 and 7.5 jointly give the exact minimum over B (the grid phase serves to tighten the incumbent; the returned value equals $\min(\lambda_F, \lambda_{CR}, N_B)$ in all cases). The complexity statement collects the per-phase bounds together with Remark 7.6. □

Remark 8.2 (A span-adaptive reading). If one accepts an answer restricted to detours that are forward, cross, or backward of span at least $\sqrt{\ell}$ — e.g. in applications where near-parallel narrow deviations are irrelevant or are post-processed separately — then steps 1–3 alone constitute a deterministic $\widetilde{O}(m\sqrt{n})$ algorithm: by Theorems 5.2 and 6.3 they return a valid second path whose cost is at most the optimum over that class (possibly better, if a narrower captured detour is cheaper).

9. Localizing the fine-grained barrier

Vassilevska Williams and Williams [38] proved that APSP and 2-SP in weighted digraphs are subcubically equivalent: either both admit truly subcubic algorithms (running time $O(n^{3-\delta})\,\mathrm{polylog}(C)$ for some fixed $\delta > 0$) or neither does. The APSP conjecture asserts that no such algorithm exists; the hard instances produced by the reduction have polynomially bounded integer weights, i.e. they satisfy the similarity assumption $C = O(n^r)$ for a fixed constant r.

Theorem 9.1 (Localization of hardness)**.** *Assume the APSP conjecture, and let* $N = \min\{\bar{c}(D) : D$ *valid backward detour of span* $< \sqrt{\ell}\,\}$ *denote the narrow-core value. Then for every fixed constant* c *there is no algorithm that computes* N *in* $O(\sqrt{n}\,m\log^c(nC))$ *time on all digraphs with polynomially bounded integer costs. (For a* single *restricted weight class the statement may fail: with* $C = O(n^r)$ *and* $r < 3 - \omega$*, subcubic algorithms exist on dense graphs [37, 39]; the theorem excludes a uniform polylogarithmic dependence on* C *over all polynomial weight ranges.)*

Proof. Fix c and suppose such an algorithm existed. Combined with steps 1–3 of SecondPath (Corollary 6.5), it would solve 2-SP exactly in $O(\sqrt{n}\, m \log^c(nC)) \subseteq O(n^{2.5} \log^{c'} n)$ time on all instances with polynomially bounded integer costs, for a constant c' depending only on c and the weight exponent. This is a truly subcubic 2-SP algorithm with polylogarithmic dependence on C, hence yields truly subcubic APSP by [38], contradicting the conjecture. □

Three comments are in order. First, Theorem 9.1 shows that, within this decomposition, the residual narrow backward class cannot be uniformly eliminated in $O(\sqrt{n}\, m \,\mathrm{polylog}(nC))$ time unless the APSP conjecture fails, while everything outside it is covered by such a computation; in this precise sense our reductions confine the hardness of the whole problem to narrow backward detours. The sparse regime tells the same story: under the minimum-weight $(2\ell'+1)$-clique hypothesis, weighted directed 2-SP admits no $O(mn^{1-\varepsilon})$ algorithm [24], and combining this with Corollary 6.5 (whose phase costs $O(\sqrt{\ell}\,(m + n\log n))$), computing the narrow-core value N inherits the same $mn^{1-o(1)}$ barrier; the combination argument is identical to the proof of Theorem 9.1. Second, the theorem does not preclude bounds with *polynomial* dependence on C: an $\widetilde{O}(m\sqrt{nC})$ algorithm for the narrow core, for instance by adapting the sampling technique of [32], would be consistent with the conjecture. Third, for dense graphs and $C = O(n^r)$ with $r < 3 - \omega$, subcubic exact 2-SP already follows from replacement paths in $\widetilde{O}(Cn^\omega)$ [37, 39]; the present algorithm is combinatorial and aimed at sparse graphs.

10. Concluding remarks

After a path-priority DFS, the directed 2-SP problem decomposes cleanly: forward and cross detours in near-linear time, wide backward detours in $\widetilde{O}(m\sqrt{n})$ by the crossing decomposition, and a narrow core — backward detours of span below $\sqrt{\ell}$ — that provably concentrates the entire fine-grained hardness of the problem and is attacked exactly by a parametric negative-cycle method whose round count adapts to the instance. An experimental study of the round count Q on real networks is beyond the scope of this theoretical paper.

The main open problem is to bound the number of parametric rounds Q for structured graph classes whose shortest paths are long — most notably

road networks and graphs of bounded highway dimension. (The planar case is already near-linear via replacement paths [8, 40].)

Data availability

No data was used for the research described in the article.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the author used Claude (Anthropic), under the author's direction, to assist with parts of the algorithm design and analysis and with the preparation of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] N. Alon, S. Chechik, S. Cohen, Deterministic combinatorial replacement paths and distance sensitivity oracles, in: Proc. 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), LIPIcs 132, 2019, 12:1–12:14.

[3] A. Al Zoobi, D. Coudert, N. Nisse, Finding the k shortest simple paths: time and space trade-offs, ACM Journal of Experimental Algorithmics 28 (2023) 1.11.

[4] A. Bernstein, A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs, in: Proc. 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), 2010, pp. 742–755.

[5] S. Chechik, O. Magen, Near optimal algorithm for the directed single source replacement paths problem, in: Proc. 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020), LIPIcs 168, 2020, 81:1–81:17.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, third ed., MIT Press, Cambridge, MA, 2009.

[7] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.

[8] Y. Emek, D. Peleg, L. Roditty, A near-linear-time algorithm for computing replacement paths in planar directed graphs, ACM Transactions on Algorithms 6 (4) (2010) 64.

[9] D. Eppstein, Finding the k shortest paths, SIAM Journal on Computing 28 (2) (1998) 652–673.

[10] G. Feng, Finding k shortest simple paths in directed graphs: a node classification algorithm, Networks 64 (1) (2014) 6–17.

[11] G. Feng, Improving space efficiency with path length prediction for finding k shortest simple paths, IEEE Transactions on Computers 63 (10) (2014) 2459–2472.

[12] A. V. Goldberg, Scaling algorithms for the shortest paths problem, SIAM Journal on Computing 24 (3) (1995) 494–504.

[13] Z. Gotthilf, M. Lewenstein, Improved algorithms for the k simple shortest paths and the replacement paths problems, Information Processing Letters 109 (7) (2009) 352–355.

[14] E. Hadjiconstantinou, N. Christofides, An efficient implementation of an algorithm for finding K shortest simple paths, Networks 34 (2) (1999) 88–101.

[15] J. Hershberger, S. Suri, Vickrey prices and shortest paths: what is an edge worth?, in: Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS 2001), 2001, pp. 252–259.

[16] J. Hershberger, M. Maxel, S. Suri, Finding the k shortest simple paths: a new algorithm and its implementation, ACM Transactions on Algorithms 3 (4) (2007) 45.

[17] J. Hershberger, S. Suri, A. Bhosle, On the difficulty of some shortest path problems, ACM Transactions on Algorithms 3 (1) (2007) 5.

[18] W. Hoffman, R. Pavley, A method for the solution of the Nth best path problem, Journal of the ACM 6 (1959) 506–514.

[19] R. M. Karp, J. B. Orlin, Parametric shortest path algorithms with an application to cyclic staffing, Discrete Applied Mathematics 3 (1981) 37–45.

[20] N. Katoh, T. Ibaraki, H. Mine, An efficient algorithm for K shortest simple paths, Networks 12 (1982) 411–427.

[21] D. Könen, D. R. Schmidt, C. Spisla, Finding all minimum cost flows and a faster algorithm for the K best flow problem, Discrete Applied Mathematics 321 (2022) 333–349.

[22] D. Kurz, P. Mutzel, A sidetrack-based algorithm for finding the k shortest simple paths in a directed graph, in: Proc. 27th International Symposium on Algorithms and Computation (ISAAC 2016), LIPIcs 64, 2016, 49:1–49:13.

[23] E. L. Lawler, A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem, Management Science 18 (1972) 401–405.

[24] A. Lincoln, V. Vassilevska Williams, R. R. Williams, Tight hardness for shortest cycles and paths in sparse graphs, in: Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA 2018), 2018, pp. 1236–1252.

[25] K. Malik, A. K. Mittal, S. K. Gupta, The k most vital arcs in the shortest path problem, Operations Research Letters 8 (4) (1989) 223–227.

[26] P. Maristany de las Casas, A. Sedeño-Noda, R. Borndörfer, M. Huneshagen, K-shortest simple paths using biobjective path search, Mathematical Programming Computation 17 (2) (2025) 349–384.

[27] E. Q. V. Martins, M. M. B. Pascoal, J. L. E. Santos, Deviation algorithms for ranking shortest paths, International Journal of Foundations of Computer Science 10 (3) (1999) 247–261.

[28] J. B. Orlin, L. A. Végh, Directed shortest paths via approximate cost balancing, Journal of the ACM 70 (1) (2022) 3.

[29] M. M. B. Pascoal, A. Sedeño-Noda, Enumerating K best paths in length order in DAGs, European Journal of Operational Research 221 (2) (2012) 308–316.

[30] A. Perko, Implementation of algorithms for K shortest loopless paths, Networks 16 (1986) 149–160.

[31] M. Pollack, The kth best route through a network, Operations Research 9 (1961) 578–580.

[32] L. Roditty, U. Zwick, Replacement paths and k simple shortest paths in unweighted directed graphs, ACM Transactions on Algorithms 8 (4) (2012) 33.

[33] A. Sedeño-Noda, An efficient time and space K point-to-point shortest simple paths algorithm, Applied Mathematics and Computation 218 (20) (2012) 10244–10257.

[34] A. Sedeño-Noda, Ranking one million simple paths in road networks, Asia-Pacific Journal of Operational Research 33 (5) (2016) 1650042.

[35] A. Sedeño-Noda, C. González-Martín, On the second point-to-point undirected shortest simple path problem, Optimization Letters 7 (8) (2013) 1875–1881.

[36] C. C. Skiscim, B. L. Golden, Solving k-shortest and constrained shortest path problems efficiently, Annals of Operations Research 20 (1989) 249–282.

[37] V. Vassilevska Williams, Faster replacement paths, in: Proc. 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2011), 2011, pp. 1337–1346.

[38] V. Vassilevska Williams, R. Williams, Subcubic equivalences between path, matrix, and triangle problems, Journal of the ACM 65 (5) (2018) 27.

[39] O. Weimann, R. Yuster, Replacement paths and distance sensitivity oracles via fast matrix multiplication, ACM Transactions on Algorithms 9 (2) (2013) 14.

[40] C. Wulff-Nilsen, Solving the replacement paths problem for planar directed graphs in $O(n \log n)$ time, in: Proc. 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), 2010, pp. 756–765.

[41] J. Y. Yen, Finding the K shortest loopless paths in a network, Management Science 17 (1971) 712–716.

[42] N. E. Young, R. E. Tarjan, J. B. Orlin, Faster parametric shortest path and minimum-balance algorithms, Networks 21 (2) (1991) 205–221.