

A LEAN CERTIFICATE FOR THE SINGLE SOURCE UNSPLITTABLE FLOW COST COUNTEREXAMPLE

ARTHUR FREITAS RAMOS, DAVID BARROS HULAK, AND RUY J. G. B. DE QUEIROZ

ABSTRACT. We explain a standalone Lean 4 and Mathlib verification of the finite counterexample announced by Dmitry Rybin to the cost-preserving single-source unsplittable-flow conjecture. The instance has seven vertices, nine directed arcs, and three terminal demands. An exact rational path flow saturates every arc, has cost 58, and has largest demand 15. Exhaustive graph-path classification reduces every unsplittable routing to two choices per terminal. Three backbone-arc inequalities force any routing with load at most the fractional load plus 15 to use at least two paid routes, each costing 30. Thus every such routing costs at least 60, ruling out cost preservation even with a non-strict congestion bound. We present the numerical certificate, its graph realization, and the boundary between the pinned proved implementation and its statement-only verification interface. The contribution is an exposition of a source-based formal certificate; no new counterexample, first formalization, or general rounding algorithm is claimed.

1. THE COST QUESTION AND THE CERTIFIED RESULT

In single-source unsplittable flow, a terminal's entire demand must be sent along one path from a common source. Dinitz, Garg, and Goemans established a rounding theorem with additive congestion bounded by the largest demand [2]. The cost-enhanced conjecture attributed to Goemans additionally asks that the rounded flow cost no more than the given fractional flow. We use the explicit formulation in Traub, Vargas Koch, and Zenklusen, Conjecture 1.3 [11]. The 1999 paper is background for the congestion theorem; it is not used here as a source for the wording of the cost conjecture.

For the positive-demand subcase sufficient here, take a finite directed graph $G = (V, A)$, source s , distinct terminals t_k different from s , positive demands d_k , a feasible nonnegative fractional flow x , and nonnegative per-unit arc costs c_a , put $D = \max_k d_k$. The existential cost-preserving assertion is that paths P_k from s to t_k can be selected so that, for every arc,

$$(1) \quad y_a := \sum_{k:a \in P_k} d_k \leq x_a + D, \quad \sum_{a \in A} c_a y_a \leq \sum_{a \in A} c_a x_a.$$

Conjecture 1.3 also asserts polynomial-time computation. Refuting its existential part suffices to refute that stronger claim, without formalizing a computational model. The cited formulation permits zero demands; our witness satisfies the stricter positivity condition of the Lean interface. A strict variant replaces the first inequality by $y_a < x_a + D$. We call the two bounds weak and strict only to distinguish $\leq$ from $<$; the cited cost conjecture uses $\leq$.

Rybin announced the counterexample on July 22, 2026, and made the discovery session public [9, 8]. The same finite certificate is formalized in Isabelle/HOL in the Archive of Formal Proofs (AFP) by Ramos, Hulak, and de Queiroz [6]. The Lean repository discussed here is a separate reimplementaion [7], not a translation imported from Isabelle. Other Lean

Date: October 1, 2026.

2020 Mathematics Subject Classification. 68R10, 90C27, 03B35.

Key words and phrases. Single-source unsplittable flow, cost-preserving rounding, finite counterexample, Lean 4, Mathlib, formal verification.

Author ORCID identifiers: Arthur Freitas Ramos, 0009-0003-3568-0325; David Barros Hulak, 0009-0002-8056-1774; Ruy J. G. B. de Queiroz, 0000-0003-1482-0977.

Copyright 2026 the authors. Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0).

verifications by Jason Hickey and by DiscreteAlias are also publicly available [4, 3]. We make no priority claim relative to these developments and no claim to discovery.

All references to the present Lean source mean revision `cc7284cf415fc2a773f00d400319ef39663c277a`. The seven selected declarations are registered in Palomar as PALOMAR-2026-08-29-000003, version 1 [5]. The mathematical core is the following finite statement.

Theorem 1.1. *There is a seven-vertex, nine-arc single-source instance with three demands and a feasible fractional flow x of cost 58, with $D = 15$, such that every unsplittable routing satisfying $y_a \leq x_a + 15$ on every arc has cost at least 60. Consequently neither the weak cost-preserving assertion nor its strict variant holds universally.*

The selected formal theorem proves this over $\mathbb{Q}$. Its graph-path classification ensures that the finite route model covers every source-to-terminal path in the actual graph. The corresponding real-valued counterexample follows from the same integer data and finite inequalities. That interpretation is explained below; a general rational-to-real transfer theorem is not part of this snapshot.

2. THE FINITE PATH FLOW INTERFACE

The generic structure `PathFlowInstance` has commodity, route, and arc carrier types K, R, A . It records an admissibility predicate $\text{Adm}(k, r)$, demands $d_k \in \mathbb{Q}$, capacities $u_a \in \mathbb{Q}$, costs $c_a \in \mathbb{Q}$, and an incidence predicate $U(k, r, a)$. The generic interface is an abstract incidence model: graph endpoints and path validity are established separately for the concrete witness. With finite K and R , fractional route amounts f_{kr} induce

$$(2) \quad x_a(f) = \sum_{k \in K} \sum_{r \in R} \begin{cases} f_{kr} & U(k, r, a), \\ 0 & \text{otherwise,} \end{cases} \quad C(f) = \sum_{a \in A} c_a x_a(f).$$

The latter sum requires finite A . The maximum demand additionally requires nonempty K and is the maximum of the finite demand image.

Definition 2.1. An instance is well formed if every demand is positive, every commodity has an admissible route, and every capacity and cost is nonnegative. A fractional routing f is feasible if the instance is well formed, $f_{kr} \geq 0$, inadmissible routes have amount zero, $\sum_r f_{kr} = d_k$ for every k , and $x_a(f) \leq u_a$ for every a .

An unsplittable routing is a function $q : K \rightarrow R$ with $\text{Adm}(k, q(k))$ for every k . Its arc load and cost are

$$(3) \quad y_a(q) = \sum_{k \in K} \begin{cases} d_k & U(k, q(k), a), \\ 0 & \text{otherwise,} \end{cases} \quad C(q) = \sum_{a \in A} c_a y_a(q).$$

The definitions `weakDggRounding` and `strictDggRounding` combine admissibility, the corresponding congestion inequality relative to $x(f)$, and $C(q) \leq C(f)$. Their existential versions quantify over q . The reference load is the specified fractional flow, not an arbitrary capacity upper bound. In this witness $x_a(f) = u_a$ on every arc, so the two numerical formulations coincide.

The finite universal negation in the registered theorem uses the specific carrier types with three commodities, two route choices, and nine arcs. It is already enough to contradict an assertion quantified over all graph instances, because the witness is realized by a genuine graph and its routes are exhaustive. The code does not establish a generic equivalence between arbitrary incidence structures and graphs.

3. THE GRAPH AND THE FRACTIONAL WITNESS

Let

$$V = \{s, u, v, w, t_1, t_2, t_3\}, \quad (d_1, d_2, d_3) = (15, 10, 15).$$

The directed arcs and all numerical data are given in Table 1. Figure 1 draws the same endpoint relation. The terminal vertices have no outgoing arcs. The ordering $s, u, v, w, t_1, t_2, t_3$

places every arc forward, so the graph is acyclic. All endpoints are distinct on each arc, and no ordered endpoint pair is repeated. These observations are immediate properties of the data; separate generic graph simplicity or acyclicity theorems are not selected in this formalization.

TABLE 1. The complete arc data. The displayed fractional load equals capacity.

Arc a	Capacity u_a	Fractional load x_a	Unit cost c_a
$s \rightarrow t_1$	10	10	2
$s \rightarrow t_2$	6	6	3
$s \rightarrow u$	24	24	0
$u \rightarrow v$	14	14	0
$u \rightarrow t_3$	10	10	2
$v \rightarrow t_1$	5	5	0
$v \rightarrow w$	9	9	0
$w \rightarrow t_2$	4	4	0
$w \rightarrow t_3$	5	5	0

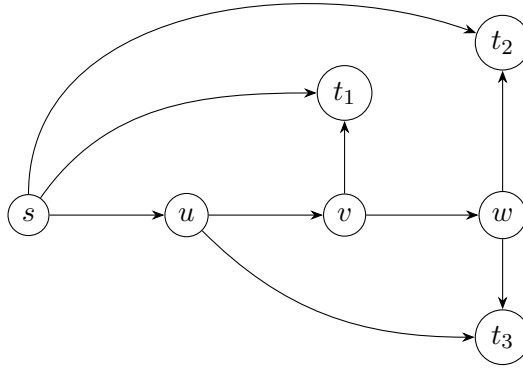


FIGURE 1. The seven vertices and nine arcs. Numerical labels are in Table 1; the drawing asserts only the endpoint relation.

For each terminal there is one paid route E_k and one free route Z_k :

$$\begin{aligned}
 E_1 &= (s, t_1), & Z_1 &= (s, u, v, t_1), \\
 E_2 &= (s, t_2), & Z_2 &= (s, u, v, w, t_2), \\
 E_3 &= (s, u, t_3), & Z_3 &= (s, u, v, w, t_3).
 \end{aligned}$$

Here vertex lists abbreviate the consecutive directed arcs. The names paid and free refer to total route cost, not to disjointness: E_3 and Z_3 both use $s \rightarrow u$.

Lemma 3.1. *Every directed source-to- t_k path is exactly E_k or Z_k .*

Proof. From w a nonempty path goes directly to t_2 or t_3 and then stops. From v it goes to t_1 or to w ; from u it goes to t_3 or to v ; from s it goes to t_1 , t_2 , or u . Following these alternatives gives precisely the six displayed terminal paths. There are no additional continuations from any terminal. $\square$

The formal predicate `edgePath` is recursive on a list of arcs. It checks each tail against the current vertex, advances to the head, and accepts the empty list exactly when the current and final vertices agree. It does not assume vertex simplicity. The proof classifies all such lists from each terminal, then from w, v, u, s , and obtains `allSourceTerminalPaths`. Thus no longer walk, omitted splice, or additional source-to-terminal route escapes the two-choice model.

The fractional route amounts are

k	f_{k,E_k}	f_{k,Z_k}	d_k
1	10	5	15
2	6	4	10
3	10	5	15

They are all nonnegative and meet the three demands. Summing their incidences gives Table 1, proving feasibility and saturation. For example the backbone loads are $5 + 4 + 10 + 5 = 24$ on $s \rightarrow u$, $5 + 4 + 5 = 14$ on $u \rightarrow v$, and $4 + 5 = 9$ on $v \rightarrow w$. As an arc-flow check, the source outflow is $10 + 6 + 24 = 40$; conservation at u, v, w is respectively $24 = 14 + 10$, $14 = 5 + 9$, and $9 = 4 + 5$. The terminal inflows are 15, 10, 15. Only three arcs have positive cost, so

$$(4) \quad C(f) = 2 \cdot 10 + 3 \cdot 6 + 2 \cdot 10 = 58, \quad D = \max\{15, 10, 15\} = 15.$$

The corresponding Lean facts are `fractionalLoadEqCapacity`, `splitFlowFeasible`, `fractionalCostValue`, and `counterexampleMaximumDemand`.

4. THE OBSTRUCTION TO COST PRESERVING ROUNDING

Let z_k be 1 if q selects Z_k , and 0 if it selects E_k . Every paid route contributes 30: its cost is respectively $15 \cdot 2$, $10 \cdot 3$, or $15 \cdot 2$. Every free route contributes zero. Hence

$$(5) \quad C(q) = 30(3 - z_1 - z_2 - z_3).$$

The crucial shared-arc loads are

$$(6) \quad y_{su} = 15z_1 + 10z_2 + 15, \quad y_{uv} = 15z_1 + 10z_2 + 15z_3, \quad y_{vw} = 10z_2 + 15z_3.$$

The constant 15 in y_{su} comes from terminal t_3 on either route.

Lemma 4.1. *If $y_a(q) \leq x_a + 15$ on every arc, then at most one free route is selected, and at least two paid routes are selected.*

Proof. If Z_1, Z_2 are selected, then $y_{su} = 40 > 24 + 15 = 39$. If Z_1, Z_3 are selected, then $y_{uv} \geq 30 > 14 + 15 = 29$. If Z_2, Z_3 are selected, then $y_{vw} = 25 > 9 + 15 = 24$. Any selection of at least two free routes contains one of these pairs, so it violates a congestion inequality. $\square$

Proof of Theorem 1.1. The preceding fractional witness is feasible and has cost 58. By Lemma 3.1, every unsplittable graph routing is represented by the three binary choices. Lemma 4.1 and (5) give $C(q) \geq 60 > 58$ for any weakly bounded routing. Thus no routing can satisfy both inequalities in (1). A strict congestion bound implies the weak bound, so the strict cost-preserving assertion fails as well. $\square$

Table 2 is an exhaustive numerical cross-check. Its excess column means $\max_a(y_a - x_a)$, so weak congestion feasibility is exactly excess at most 15. All four feasible assignments actually have excess strictly below 15. This also shows that the obstruction is the cost requirement, rather than a lack of any bounded routing.

For example (E_1, E_2, Z_3) has loads, in the arc order of Table 1,

$$(15, 10, 15, 15, 0, 0, 15, 0, 15).$$

Every load is strictly below $x_a + 15$, and its cost is 60. Thus 60 is the exact minimum under either congestion reading. This attainment check and the full table are elementary consequences of the displayed data; the pinned selected Lean result asserts the lower bound, not a separately packaged optimum-attainment theorem. The all-free routing has cost zero and maximum excess 26, illustrating that cost alone is also satisfiable. The example makes no claim against rounding guarantees allowing larger additive error, such as $2D$.

TABLE 2. All eight routing assignments, computed from the arc incidences.

t_1	t_2	t_3	Cost	Maximum excess	Bound ≤ 15
E_1	E_2	E_3	90	5	Yes
E_1	E_2	Z_3	60	10	Yes
E_1	Z_2	E_3	60	6	Yes
E_1	Z_2	Z_3	30	16	No
Z_1	E_2	E_3	60	10	Yes
Z_1	E_2	Z_3	30	16	No
Z_1	Z_2	E_3	30	16	No
Z_1	Z_2	Z_3	0	26	No

5. THE LEAN PROOF AND ITS VERIFICATION BOUNDARY

The development uses Lean 4 [1] and Mathlib [10]. The graph, route lists, rational amounts, loads, and predicates are in `Definitions.lean`. The substantive proofs are in `Proof.lean`. Finite carrier instances explicitly list the three commodities, two route choices, and nine arcs. This permits finite sums to simplify to exact rational expressions.

Path completeness is proved structurally, not stipulated by making all binary choices admissible. In the concrete instance both choices are declared admissible, but `routeEdgesValid` proves their graph validity and `allSourceTerminalPaths` proves completeness for arbitrary arc lists. This separation is essential: an incidence-table counterexample would otherwise leave open whether the real graph admits an unlisted cheap routing.

The arithmetic proof first identifies the finite universes. It uses `simp` and `norm_num` for well-formedness, demand sums, capacity saturation, maximum demand, and fractional cost. The three backbone-load formulas and `unsplittableCostEqPaidCount` are proved by cases on the three route choices. In `boundedRoutingHasTwoPaid`, the hypotheses for $s \rightarrow u, u \rightarrow v, v \rightarrow w$ are extracted; each of the eight cases reduces to rational arithmetic, and `linarith` closes the impossible ones. Coercing the paid-count bound to $\mathbb{Q}$ yields `boundedUnsplittableCost`. The remaining witness and universal negations instantiate the property at the concrete instance and combine $60 \leq C(q) \leq 58$.

The statement surface `Challenge.lean` repeats the definitions and states seven targets with deliberate `sorry` bodies. It is an interface for statement comparison, not the completed proof. `Solution.lean` imports `DinitzGargGoemans.Proof`, whose import chain goes through `Definitions` rather than `Challenge`. The inspected implementation files contain no `sorry`, `admit`, `native_decide`, or custom axiom declaration. A textual inspection by itself does not establish the transitive axiom footprint.

The pinned comparator configuration selects the seven theorems named in the registry and ten load, cost, feasibility, rounding, and witness definitions. It permits only `propext`, `Quot.sound`, and `Classical.choice`, and enables NanoDa checking. The registry records successful historical mechanical verification on August 28, 2026, under Lean v4.33.0; the Mathlib manifest resolves to `db584cd6d46c92f209a44c0f1c829460d327499d`. Its linked workflow is run 33131019615 [5]. These are historical verification records, not a new kernel or comparator run performed during manuscript preparation.

The boundary of the formal result is therefore explicit. It is a finite rational counterexample with exhaustive graph-path coverage and exact cost separation. Interpretation as an ordinary real-valued graph-flow counterexample uses the same integer incidence and arithmetic, and does not require choosing any irrational route amounts: the routing domain is still the same eight assignments. Source alignment, the conjecture’s historical attribution, and prose claims about other work are scholarly judgments outside the Lean kernel. The development does not

formalize the general Dinitz–Garg–Goemans theorem, algorithmic complexity, parameterized counterexample families, or every published variant of the cost conjecture.

6. PROVENANCE AND REPRODUCIBILITY

This manuscript is a new exposition of the pinned Lean certificate. The counterexample is credited to Rybin and the GPT-5.6 Pro session he published; the prior AFP proof document remains a separate Isabelle artifact [8, 6]. No prior proof-document text or figure is reproduced here. The data tables and diagram were prepared from the inspected Lean definitions, with an independent exact-arithmetic enumeration used as a numerical cross-check. A matching enumeration is not a substitute for formal theorem verification.

The repository metadata records GPT-5 agent assistance for the historical Lean proof engineering. The present manuscript was generated primarily with OpenAI GPT-6.1 assistance for exposition, source comparison, bibliographic research, and typesetting. Arthur Freitas Ramos directed the submission and reports understanding some parts of the manuscript. No comprehensive personal line-by-line review by Arthur, or complete understanding by every listed author, is claimed. Model-assisted manuscript checks should not be read as independent human peer review.

The referenced Lean repository is licensed Apache 2.0, and the AFP entry has a BSD license. Those artifact licenses remain distinct from this manuscript’s CC BY 4.0 license. The source archive for this article contains only its L^AT_EX source and bibliography; it does not bundle the Lean project or the prior Isabelle document. The immutable source and registry references identify the formal artifact needed to inspect or independently reproduce the mathematical verification.

REFERENCES

1. Leonardo de Moura and Sebastian Ullrich, *The Lean 4 theorem prover and programming language*, Automated Deduction – CADE 28, Lecture Notes in Computer Science, vol. 12699, Springer, 2021, https://doi.org/10.1007/978-3-030-79876-5_37, pp. 625–635.
2. Yefim Dinitz, Naveen Garg, and Michel X. Goemans, *On the single-source unsplittable flow problem*, *Combinatorica* **19** (1999), no. 1, 17–41, <https://doi.org/10.1007/s004930050043>.
3. DiscreteAlias, *unsplittable-flow*, Lean verification repository, 2026, Separate verification and catalogue of cost-conjecture formulations. Repository consulted October 1, 2026. <https://github.com/DiscreteAlias/unsplittable-flow>.
4. Jason Hickey, *dinitz-verify*, Lean verification repository, 2026, With Claude assistance; separate verification of the same finite witness. Repository consulted October 1, 2026. <https://github.com/jyh/dinitz-verify>.
5. Palomar Registry, *PALOMAR-2026-08-29-000003, version 1*, Immutable formalization registry record, 2026, Registered August 29, 2026. <https://palomar-registry.org/entry?id=PALOMAR-2026-08-29-000003&version=1>.
6. Arthur Freitas Ramos, David Barros Hulak, and Ruy Jose Guerra Barretto de Queiroz, *A formal counterexample to the cost-preserving single-source unsplittable flow conjecture*, Archive of Formal Proofs, 2026, Entry dated July 22, 2026. Isabelle/HOL formalization. https://isa-afp.org/entries/Dinitz_Garg_Goemans_Counterexample.html.
7. ———, *Lean formalization of the Dinitz–Garg–Goemans cost counterexample*, Version-pinned source repository, 2026, Revision cc7284cf415fc2a773f00d400319ef39663c277a. <https://github.com/Arthur742Ramos/dinitz-garg-goemans-counterexample-lean/tree/cc7284cf415fc2a773f00d400319ef39663c277a>.
8. Dmitry Rybin, *Counterexample to Dinitz conjecture*, Public shared ChatGPT session, 2026, Source discovery transcript, including the complete finite certificate. <https://chatgpt.com/share/6a60b2eb-0b64-83ee-9c76-7931ca1de063>.
9. ———, *Counterexample to the Dinitz–Garg–Goemans cost conjecture*, Public announcement on X, July 22, 2026, <https://x.com/DmitryRybin1/status/2079904005652893709>.
10. The mathlib Community, *The Lean mathematical library*, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, ACM, 2020, <https://doi.org/10.1145/3372885.3373824>, pp. 367–381.
11. Vera Traub, Laura Vargas Koch, and Rico Zenklusen, *Single-source unsplittable flows in planar and bounded-genus graphs*, Mathematical Programming, online first, 2026, Published July 22, 2026. Definition 1.1 and Conjecture 1.3. <https://doi.org/10.1007/s10107-026-02365-x>.