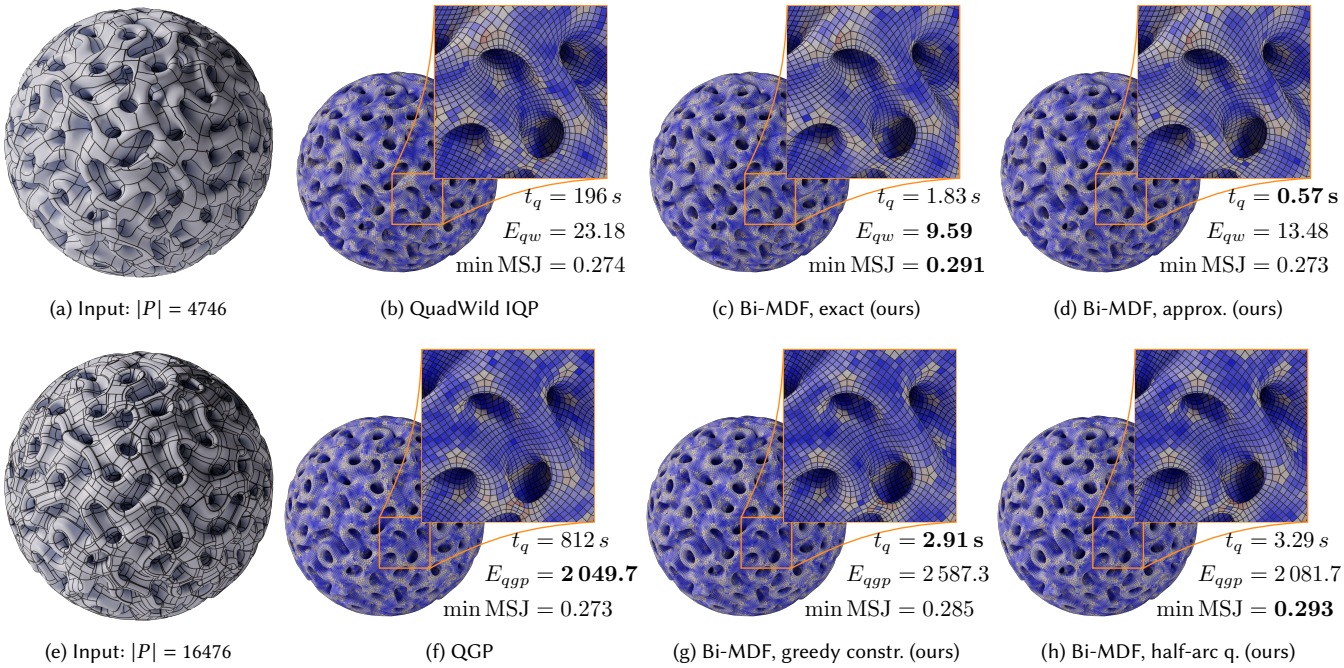
# Min-Deviation-Flow in Bi-directed Graphs for T-Mesh Quantization

MARTIN HEISTERMANN, University of Bern, Switzerland
JETHRO WARNETT, University of Oxford, United Kingdom
DAVID BOMMES, University of Bern, Switzerland




(a) Input: $|P| = 4746$ (b) QuadWild IQP (c) Bi-MDF, exact (ours) (d) Bi-MDF, approx. (ours)

(e) Input: $|P| = 16476$ (f) QGP (g) Bi-MDF, greedy constr. (ours) (h) Bi-MDF, half-arc q. (ours)

Fig. 1. Bi-MDF optimization provides orders-of-magnitude faster performance when quantizing polygonal (a) and quadrangular (e) T-meshes of this genus-548 gyroid model. Colors indicate element quality as measured by the minimum scaled Jacobian (MSJ). For the QuadWild [Pietroni et al. 2021] based results (b-d) we chose $\alpha = 0.005$ and no singularity alignment. (b) uses QuadWild's default settings for clustering ($n = 300$), per-cluster time limit (200 s), and early abort heuristics. With clustering and abort heuristics disabled, QuadWild achieves an energy $E_{qw} = 9.70$ with a duality gap of 23.8% after reaching a 24 h time limit. Figures (f-g) demonstrate performance compared to QGP [Campen et al. 2015]. Bi-MDF with greedily chosen separation constraints (g) is fastest but leads to slightly worse results. Half-arc quantization (h) expands the feasible region of allowed quantizations by quantizing either side of an arc separately.

Subdividing non-conforming T-mesh layouts into conforming quadrangular meshes is a core component of state-of-the-art (re-)meshing methods. Typically, the required constrained assignment of integer lengths to T-Mesh edges is left to generic branch-and-cut solvers, greedy heuristics, or a combination of the two. This either does not scale well with input complexity or delivers suboptimal result quality. We introduce the *Minimum-Deviation-Flow Problem* in bi-directed networks (Bi-MDF) and demonstrate its use in modeling and efficiently solving a variety of T-Mesh quantization problems. We develop a fast approximate solver as well as an iterative refinement algorithm based on matching in graphs that solves Bi-MDF exactly. Compared to the state-of-the-art QuadWild [Pietroni et al. 2021] implementation on the authors' *300* dataset, our exact solver finishes after only 0.49% (total 17.06s) of their runtime (3491s) and achieves 11% lower energy while an approximation is computed after 0.09% (3.19s) of their runtime at the cost of 24% increased energy. A novel half-arc-based T-Mesh quantization formulation extends the feasible solution space to include previously unattainable quad meshes. The Bi-MDF problem is more general than our application in layout quantization, potentially enabling similar speedups for other optimization problems that fit into the scheme, such as quad mesh refinement.

CCS Concepts: • **Theory of computation → Network optimization**; *Integer programming*; • **Computing methodologies → Mesh models**.

Additional Key Words and Phrases: Quad meshing, T-Mesh quantization, Discrete optimization, Flow networks, Bidirected graphs, Binet matrices

**ACM Reference Format:**
Martin Heistermann, Jethro Warnett, and David Bommes. 2023. Min-Deviation-Flow in Bi-directed Graphs for T-Mesh Quantization. *ACM Trans. Graph.* 42, 4 (August 2023), 25 pages. https://doi.org/10.1145/3592437

Authors' addresses: Martin Heistermann, University of Bern, Institute for Computer Science, Bern, Switzerland, martin.heistermann@unibe.ch; Jethro Warnett, University of Oxford, Oxford, United Kingdom; David Bommes, University of Bern, Institute for Computer Science, Bern, Switzerland, david.bommes@unibe.ch.


0730-0301/2023/8-ART
https://doi.org/10.1145/3592437

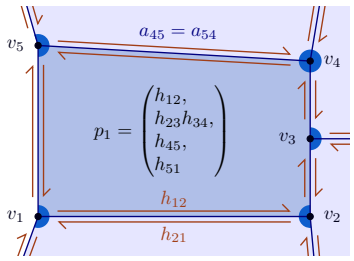



Fig. 2. An example T-Mesh consisting of **v**ertices, **a**rcs, **h**alf-arcs, and **p**atches. Circular segments (blue) indicate patch corners. Refer to Section 4.1.1 for a formal definition.

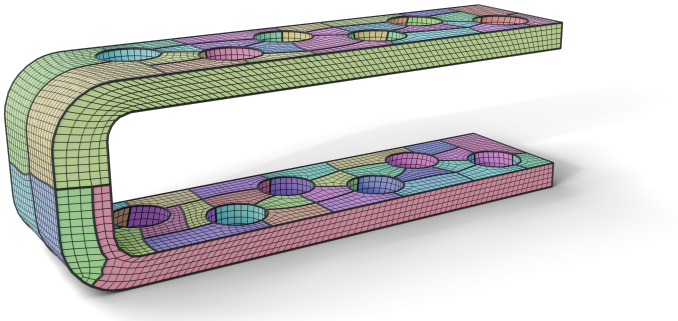


Fig. 3. The polygonal T-mesh quantization without alignment constraints for the Toothbrush-holder[1] model included in the QuadWild-300 dataset presents a seemingly simple quantization problem consisting of 126 patches, but solving the associated IQP problem to optimality using Gurobi is surprisingly expensive. QuadWild's default settings' early abort heuristic stops optimization after 60s with a duality gap of 18.9% and an energy of 0.2928. Running on 128 cores, Gurobi already finds an optimal solution ($E = 0.29186$) after 5 seconds at a duality gap of 20.4%. However, the duality gap is only down to 14.4% when reaching the memory limit of 1 TB after over 5 hours (28 core-days) of computation. Our single-threaded Bi-MDF solver finishes in 26ms, finding the optimum.

## 1 INTRODUCTION

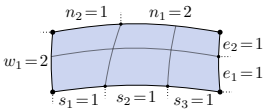
Several state-of-the-art field-based quadrangular re-meshing methods generate non-conforming T-mesh layouts as an intermediate step towards the final quad mesh. This approach was initially conceived as a way to split the expensive mixed-integer problem of generating an integer-grid map into two easier problems [Campen et al. 2015]: A seamless map can be computed for an input surface from a frame field via purely continuous optimization. Its motorcycle graph [Eppstein et al. 2008] yields a T-mesh – a patch layout that may contain T-junctions, i.e., vertices that are corners of some but not all incident patches (cf. Figure 2). The final integer-grid map can then be obtained from this by solving a purely discrete optimization problem, commonly called T-mesh quantization. Later approaches forgo the seamless map generation and operate on the frame field directly to obtain a T-mesh [Pietroni et al. 2021].

In this work, we focus on the T-mesh quantization problem and frame it as an instance of a broader class of combinatorial problems, a generalized minimum-cost flow problem on bi-directed graphs.

### 1.1 The T-mesh quantization problem

Depending on the algorithmic pipeline, the individual layout patches constituting the T-mesh may already all be quadrangular, consist of a limited set of $n$-gons, be simply connected, or might not guarantee any of these useful restrictions. For simplicity, we will always assume that each patch has disk topology.

In its most basic form, a T-mesh quantization assigns a positive natural number to each T-mesh edge (*quantized length*), indicating the number of output quad mesh edges into which to subdivide it. Suitable per-patch *consistency constraints* ensure the existence of a local quad mesh for each patch that conforms to the subdivided patch boundaries, implying a conforming mesh of the entire domain. A simple example of such consistency constraints is to require opposite sides of a quadrangular patch to be quantized to the same number, allowing tessellation with a regular grid, as depicted in the inset figure.




$$n_1 + n_2 = s_1 + s_2 + s_3$$
$$w_1 = e_1 + e_2$$

In addition to the various kinds of T-meshes, two main objectives need to be balanced depending on the application:

- Input fidelity: Quantized patches should be geometrically similar to the original patches to minimize distortion, usually by demanding quantized arc lengths close to their (parametric or geodesic, potentially scaled) length in the T-mesh. Alternatively, deviation of edge directions from prescribed directions (e.g., by frame field or parameterization) should be minimized.
- Topological simplicity: The resulting topological structure should be simple for semi-structured quad meshes, with few well-aligned singular vertices of low degree. Ideally, the base complex should contain a low number of faces.

In practice, quantization problems are often formulated as (Mixed-) Integer Linear (ILP) or Quadratic (IQP) Programs [Lyon et al. 2021b, 2020; Peng et al. 2014; Pietroni et al. 2021] and solved using commercial black-box solvers that usually employ branch-and-cut algorithms [CPLEX 2009; Gurobi Optimizer 2022]. For T-meshes of relatively low complexity, this approach usually finds a solution quickly. However, worst-case exponential runtime motivates modeling the problems in a way that admits efficient solvers that avoid worst-case breakdowns. Such worst-case behavior is not just of theoretical concern but does occur in practice, as demonstrated in Figure 3.

### 1.2 Min-deviation quantization as ILP

We will start by looking into a simplified example problem on a T-mesh consisting only of quadrilateral patches formed by a set of arcs. For each undirected T-mesh arc $a_{ij} = a_{ji} \in \mathcal{A}$, we introduce one integer variable $x_{ij} \in \mathbb{N}_0$ representing its quantized length. For each patch $p \in \mathcal{P}$ with sides $p^0 \dots p^3$, the aforementioned simple

[1] Thingi10k [Zhou and Jacobson 2016] id 54820, original author *sholland91*

consistency constraints amount to a pair of constraints

$$\forall k \in \{0,1\} : \sum_{h_{ij} \in p^k} x_{ij} = \sum_{h_{ij} \in p^{k+2}} x_{ij}. \tag{1}$$

Our objective is to minimize the sum of absolute deviations from real-valued target lengths $\ell_{ij}$,

$$\min_{x \in \mathbb{N}_0^{\mathcal{A}}} \sum_{a_{ij} \in \mathcal{A}} \left| x_{ij} - \ell_{ij} \right|. \tag{2}$$

To bring our problem into ILP form, we apply the standard transformation $x_{ij} = \ell_{ij} + x_{ij}^+ - x_{ij}^-$ with new variables $x_{ij}^+, x_{ij}^- \in \mathbb{R}_{\geq 0}$, and $x_{ij}^- \leq \ell_{ij}$, the latter to ensure non-negativity of $x_{ij}$. This leads to the linear objective function

$$\min_{x} \sum_{a_{ij} \in \mathcal{A}} x_{ij}^+ + x_{ij}^-, \tag{3}$$

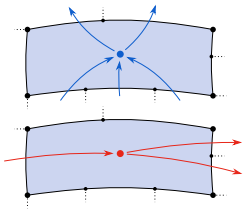
equivalent under minimization, since only one of $x_{ij}^+$ or $x_{ij}^-$ is non-zero in an optimal solution.

The new variables $x_{ij}^+$ and $x_{ij}^-$ represent positive and negative deviations from the ideal length $\ell_i$ that are necessary to obtain per-patch consistency.

### 1.3 Min-deviation quantization as flow problem

A flow $f : E \to \mathbb{N}_0$ in a directed graph $(V, E)$ models the transport of units between nodes along its edges. An edge $e_{ij} = v_i \to v_j$ moves $f_{ij} = f(e_{ij}) \geq 0$ units from its tail $v_i$ to its head $v_j$.

We construct a flow network $(V, E)$ as the *patch dual* of the T-Mesh input. It has a node for each pair of opposite patch sides and an edge for each dual T-mesh arc, connecting the corresponding node of the incident patches, as illustrated in the inset figure. Patch consistency constraints now are precisely *flow conservation* constraints, ensuring that the total incoming and outgoing flows at each node are equal:

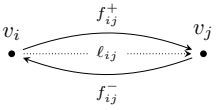


$$\forall v_i \in V : \underbrace{\sum_{e_{ji} \in E} f_{ji}}_{\text{inflow}} - \underbrace{\sum_{e_{ij} \in E} f_{ij}}_{\text{outflow}} = 0. \tag{4}$$

We apply an analog of the transformation used for the ILP to represent the objective as a linear function over edge flow values,

$$f_{ij} = \hat{\ell}_{ij} + f_{ij}^+ - f_{ij}^-, \tag{5}$$

where $\hat{\ell}_{ij} = \text{round}\left(\ell_{ij}\right)$, constrained by $0 \leq f_{ij}^+$ and $0 \leq f_{ij}^- \leq \hat{\ell}_{ij}$. We can understand this as replacing each edge $e_{ij}$ with a virtual edge carrying a constant amount of flow $\hat{\ell}_{ij}$, as well as a pair of antiparallel edges $e_{ij}$ and $e_{ji}$ which allow positive and negative deviations from the ideal value. Minimizing $\left|f_{ij} - \hat{\ell}_{ij}\right|$ now amounts to minimizing $f_{ij}^+ + f_{ij}^-$. Target pseudo-flows $\hat{\ell}_{ij}$ alone will generally violate flow conservation by some $b_i$ at vertices $v_i$. This must be balanced by flows through $e_{ij}$ and $e_{ji}$. Algebraically, after applying the substitution from Eqn. 5, the flow conservation condition (Eqn. 4) amounts to




$$\forall v_i \in V : \sum_{j:e_{ij} \in E} f_{ij}^+ - f_{ij}^- - \sum_{j:e_{ji} \in E} f_{ji}^+ - f_{ji}^- = b_i \tag{6}$$

for per-node *demands*

$$b_i := - \sum_{j:e_{ij} \in E} \hat{\ell}_{ij} + \sum_{j:e_{ji} \in E} \hat{\ell}_{ji}. \tag{7}$$

Minimizing total deviations, we arrive at a *Minimum-Cost-Flow* (MCF) problem [Ford and Fulkerson 1962]:

$$\min_{f} \sum_{ij} f_{ij}^+ + f_{ij}^-, \tag{8}$$

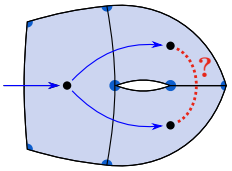
$$\text{s.t. } \forall v_i \in V : \sum_{j:e_{ij} \in E} f_{ij}^+ - f_{ij}^- - \sum_{j:e_{ji} \in E} f_{ji}^+ - f_{ji}^- = b_i \tag{9}$$

$$\forall e_{ij} \in E : 0 \leq f_{ij}^+ \tag{10}$$

$$\forall e_{ij} \in E : 0 \leq f_{ij}^- \leq \hat{\ell}_{ij} \tag{11}$$

The linear constraints (Eqn. 9) are equivalent to $Cf = b$, where $C$ is the node-edge incidence matrix of the flow network. Each edge $i \to j$ corresponds to a column in $C$ with a 1 in row $j$, $-1$ in row $i$, and 0 for all other entries. Due to this special structure, $C$ is always totally unimodular [Ford and Fulkerson 1962; Hoffman and Kruskal 1956]. Total unimodularity implies the existence of integer optimal solutions for the continuous relaxation, as long as all entries of $b$ are integer. The Network Simplex algorithm [Orlin 1997] can efficiently find such solutions, exploiting the graph structure to improve performance compared to the more general classic simplex algorithm [Dantzig 1951, 1963].

So far, we have ignored the problem of combining per-patch flow network pieces into a consistent global network. An implicit patch orientation dictates which patch sides receive inflow or provide outflow. Our flow network extends to the entire T-mesh if we can find an orientation for each patch, such that its outflow edges match up with the inflow edges of adjacent patches and vice versa. Unfortunately, this is not always possible. Take Mitchell's *2-1 radish* [Mitchell 2021], illustrated in the inset, as an example: No matter how we try to orient our network edges, there will always be two patches that demand conflicting orientations for their common arc, i.e., inflow or outflow from both patches.




This observation implies that modeling T-mesh quantization using network flows requires generalizing classical flow problems.

### 1.4 Min-deviation quantization as bi-directed flow problem

Bi-directed networks are a suitable generalization: In addition to regular directed edges, they also admit edges that have two heads or two tails [Edmonds and Johnson 1970; Edmonds 1967; Lawler 1976], which solves the orientation problem outlined above.

The first row of Figure 4 illustrates our choice of graphical notation. Intuitively, applying one unit of flow through a *head-head* edge adds one unit to each incident node, while one unit of flow through a *tail-tail* edge removes one unit from each. The analogies

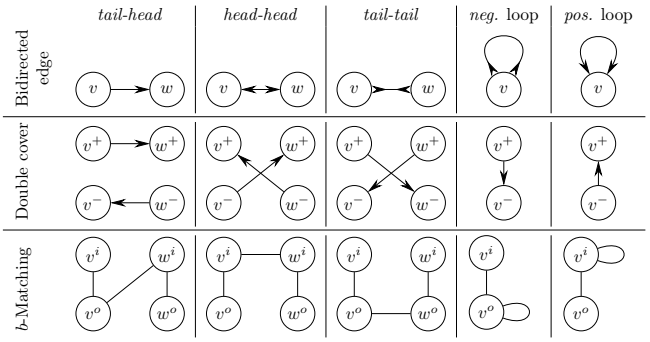



Fig. 4. Types of bi-directed edges and their representation in the double cover MCF as well as the $b$-matching instance

describing flows in simply directed networks, such as fluid in a pipe network, do not directly apply anymore. However, we can imagine a *tail-tail* edge to contain a special sink that always takes the same amount from each incident node and a *head-head* edge to contain a special source that sends flow equally to both nodes.

With these additional edge types, we can model the constraints of our example problem as a bi-directed MCF (Bi-MCF) problem. However, the resulting constraint matrix – now the node-edge incidence matrix of a bi-directed graph – generally is not totally unimodular anymore.

Luckily, weaker properties that still hold for these matrices, which belong to the class of *binet*[2] matrices: If all entries of both the right-hand side $b$ and the variable bounds $u$, $l$ are integer, the vertices of the linear program's feasible region polytope – its basic feasible solutions (BFS) – are always at half-integer coordinates, i.e., each entry is in $\frac{1}{2}\mathbb{Z}$ [Bolker and Zaslavsky 2006]. If $b$, $u$, and $l$ consist only of even integers, all BFS are fully integer [Appa and Kotnyek 2000; Appa et al. 2007; Hochbaum 2004; Kotnyek 2002].

The (half-)integrality of BFS implies the existence of (half-)integral optimal solutions because every optimal value can be attained at a BFS.

### 1.5 Overview

After an overview of the related work on both T-mesh quantization and solving relevant optimization problems in Section 2, we define the *Bidirected Minimum Deviation Flow* (Bi-MDF) problem in Section 3, which generalizes the MCF framework to per-edge convex cost functions, and discuss various methods to solve Bi-MDF problems both approximately and exactly. Section 4 covers Bi-MDF modeling of various T-mesh quantization problems, including arc- and half-arc quantization of quadrangular T-meshes with support for patch collapses limited by separation constraints, as well as polygonal T-mesh quantization with multiple objectives. An empirical evaluation in Section 5 demonstrates the practical efficacy of our methods.

[2]not related to the mathematician Jacques Binet

## 2 RELATED WORK

### 2.1 Patch layout quantization for quad meshing

*2.1.1 CAD meshing.* The earliest works known to us that we would, in retrospect, categorize as T-mesh quantization aim at meshing CAD models that are non-conforming assemblies of topologically polygonal patches.

Tam and Armstrong formulate a quantization problem to produce quad meshes from polygonal input meshes [Tam and Armstrong 1993]. They minimize a weighted sum of absolute deviations from target lengths under per-patch linear constraints that ensure meshability. The resulting ILP is solved using Gomory's cutting plane algorithm.

Scott Mitchell has been developing and publishing a series of algorithms for the *Interval Assignment* problem, a generalization of the quantization problem that also encompasses assigning edge lengths to polyhedral complexes to facilitate hexahedral meshing of volumes. Notably, he utilizes multi-objective optimization in the form of lexicographic minimization as opposed to a single objective consisting of a sum of per-edge energies. This line of research includes *Branch & Bound IIA* [Mitchell 2000], which utilizes multiple rounds of optimization to minimize a linear objective, and *Nonlinear Interval Assignment* [Mitchell 2013], which uses a more conventional sum-of-cubes objective, augmented by a secondary optimization with a piecewise linear objective in local neighborhoods. The most recent work, *Incremental Interval Assignment* [Mitchell 2021], applies integer linear algebra first to find a feasible solution for the linear constraint system, which is subsequently improved by adding integer vectors of the system's null space obtained from a reduced row echelon form of the constraint matrix. Optimality and feasibility with respect to variable bounds are not guaranteed, but feasible solutions of sufficient quality are obtained quickly in many practical cases, according to the author.

A number of works by Müller-Hannemann, Möhring, and Weihe have the closest relation to our approach for modeling quantization problems: They construct a flow problem on a bi-directed graph from per-patch templates where flows on dual mesh edges correspond to the number of subdivisions of the primal edge. Templates are selected based on patch geometry and potentially replaced by more general *emergency* templates to treat infeasibility in an iterative process [Möhring et al. 1995; Möhring et al. 1997]. Alternatively, more complex general templates can be employed from the start, which do not require the iterative process of choosing appropriate templates [Müller–Hannemann 2000]. The latter paper utilizes piecewise-linear convex cost functions to penalize deviations from target lengths, albeit with fixed upper bounds. In contrast to our work, their approach always quantizes arcs to at least one. It is thus not directly applicable to quantization schemes that rely on the structural simplifications facilitated by this extension. The weighted $b$-matching problems resulting from this process are solved either using the commercial ILP solver CPLEX, custom heuristics, or, in later works, a custom exact solver.

*2.1.2 Quantized Global Parameterization and extensions.* In recent years, much focus has been on frame-field-based methods to create quad meshes from unstructured input data, such as triangle meshes.

Campen et al. developed the *Quantized Global Parameterization* (QGP) algorithm, an automatic quad meshing method that extracts a quadrangular T-mesh from a "seamless" parameterization by means of the motorcycle graph [Eppstein et al. 2008], which is then quantized using a custom heuristic to finally produce a conforming quad mesh [Campen et al. 2015]. Notably, their methods allow for structural simplification by collapsing T-mesh patches when some of their sides are quantized to zero length. This is especially helpful in eliminating thin quad strips that arise as artifacts from T-mesh generation.

Allowing quantizing to zero, however, necessitates *separation* constraints that enforce strictly positive distances in parametric space between singularities, boundaries, and features to prevent degenerate results. In the first stage, QGP iteratively adds integer vectors from the constraint matrix null space (corresponding to a subset of circuits in a bi-flow network representation) to an initial all-zero solution until each edge is assigned a strictly positive length, trivially achieving feasibility w.r.t. separation constraints. Subsequently, further integer null space vectors are added and subtracted that improve the quadratic objective function while only allowing steps that do not violate separation constraints, thus preserving feasibility. For fine target mesh resolutions, this process takes a long time, however, this could at least partially be ameliorated by adding multiples of the null space vectors.

Lyon et al. extend QGP to compute coarse quad layouts by first computing an extended version of the motorcycle complex, on which they then formulate quantization as an ILP [Lyon et al. 2021a]. Their method aims to find the coarsest layout that fulfills hard constraints which imply upper bounds on the angular deviation between mesh edge directions and the gradient of the input seamless map. A single parameter $\alpha$, specifying the tolerated angular deviation, allows a user to balance layout complexity and geometric fidelity.

In a later work, they managed to relax separation constraints to allow limited merging of singularities to simplify the resulting layout [Lyon et al. 2021b]. This relaxation also allows quantizing T-meshes obtained from tracing the motorcycle graph directly in a frame field without having to explicitly modify the T-mesh to ensure the existence of a feasible quantization solution as proposed by Myles et al. [Myles et al. 2014].

The authors' formulation of separation constraints proves very useful for our new modeling approaches: A simple non-zero path constraint (i.e., T-mesh paths that may not be quantized to a sum of zero) for each motorcycle trace is sufficient (albeit not necessary) to prevent undesired collapses. This greatly simplifies the handling of separation constraints.

This approach to quad layout generation is also helpful for coarsening existing quad meshes: Couplet et al. apply it to T-meshes generated from unstructured quad meshes to obtain coarse quad layouts for higher-order meshes [Couplet et al. 2021].

*2.1.3 Quantization of polygonal T-meshes.* QuadMixer [Nuvoli et al. 2019] performs polygonal T-mesh quantization for localized quad meshing to complete a partial quad mesh obtained from Boolean-like blending operations between input quad meshes. The problem is modeled as IQP to assign strictly positive integer lengths to each T-mesh arc before tessellating each T-mesh patch according to these assigned lengths. Apart from fixed values on boundaries that ensure conformity with existing quad patches, the only other hard constraint is an even boundary length for each patch to ensure the existence of a tessellation. The objective is a weighted sum of an isometry term that penalizes deviation from target lengths and a regularity term for quadrilateral T-mesh patches that favors regular tessellations. An optional fallback to a simpler ILP formulation improves runtime in complex cases.

The recent quad-meshing tool QuadWild [Pietroni et al. 2021] decomposes the input mesh into a field-aligned polygonal quad layout, which is quantized using an improved version of the QuadMixer IQP. The authors extend the regularity term to discourage the formation of more than one singular vertex in non-quad patches and add a more global singularity alignment term to minimize the resulting layout complexity. Additionally, violated soft constraints are removed in a second round of optimization to improve result quality. To achieve acceptable runtimes for large and complex inputs, they partition the input surface into contiguous subsets that are solved individually after fixing their shared boundaries. An early-abort heuristic further prevents excessive solver runtimes by accepting sub-optimal solutions based on a function of runtime and optimality gap.

Note that a T-mesh quantization does not uniquely define a quadrangulation: additional singular vertices allow for an infinite number of patch tessellations. Multiple methods have been proposed to find suitable patch tessellations using topological and geometric objectives, e.g., using ILP solvers [Peng et al. 2014; Takayama et al. 2014] or a data-driven approach selecting tessellations from a database [Marcias et al. 2015], allowing for fine-grained user control.

*2.1.4 Hexahedral meshing.* Brückler et al. extend T-mesh quantization to the volumetric case, quantizing 3D motorcycle complexes created from seamless maps [Brückler et al. 2022b] to robustly obtain hexahedral meshes using IQP. Their method allows greater flexibility by permitting negative quantizations as long as these do not cause inversions. This extension could also be applied to the 2-D case and parallels the half-arc quantization approach we introduce in Section 4.3.

*2.1.5 Flows and matching for quad meshing.* Huang et al. developed *QuadriFlow* [Huang et al. 2018], an extension to the *Instant Field-Aligned Meshes* method [Jakob et al. 2015], aiming to minimize the number of singular vertices. They set up an ILP that modifies the *position field* to eliminate singularities while minimizing length deviations. Not mentioned by the authors, their ILP corresponds to a Bi-MCF problem. They approximate it as an MCF problem by *balancing* as many edges as possible (corresponding to partially orienting the bi-directed network), then replacing the variables corresponding to non-orientable edges with constant values. For large inputs, they further approximate an MCF solution by instead solving a maximum flow problem. Our problem formulations and algorithms could be used to solve their ILP exactly.

Razafindrazaka et al. utilize perfect matching problems with additional *conflict pair constraints* to derive coarse quad layouts from frame fields, parameterizations [Razafindrazaka et al. 2015], or an existing quad mesh [Razafindrazaka and Polthier 2017]. In contrast

to our method, the matching problems they construct directly target singularity connectivity.

### 2.2 Solving Bi-MCF and $b$-matching problems

A usual first step in solving Bi-directed flow problems, or equivalently, $b$-matching problems [Edmonds and Johnson 1970; Edmonds 1967], is the computation of a feasible approximation from the solution of a half-integer relaxation of the problem, often referred to as *fractional jump-start*.

Anstee introduced a method to obtain such a half-integer optimal solution[3] to a non-perfect weighted $b$-matching problem that involves the construction of an MCF problem in a bipartite directed graph, as well as a rounding procedure that can be used to transform any such half-integer solution into an all-integer feasible solution that can then be used as a good starting point to solve to optimality [Anstee 1987].

To obtain half-integer super-optimal solutions to certain classes of linear programs that include Bi-MCF problems, Hochbaum proposes a *monotonization* approach that constructs an MCF problem on a directed double cover graph of the bi-directed network [Hochbaum 2004] without an intermediate reduction to $b$-matching. This results in a smaller MCF problem compared to Anstee's method. If all bounds and $b$ entries are even, her method can obtain the integer optimum. We will refer to this approach as *symmetric double cover*. Medvedev and Brudno apply her approach in a genome assembly algorithm and report satisfactory results [Medvedev and Brudno 2009].

Finally, Müller-Hannemann and Schwartz describe a flexible software design for a $b$-matching algorithm using a fractional jump-start followed by Pulleyblank's algorithm [Müller-Hannemann and Schwartz 2000; Pulleyblank 1973] and provide empirical results on tests of multiple design choices [Müller-Hannemann and Schwartz 2001], however, unfortunately, did not publish their code.

For this reason, we cannot directly compare the performance of their solver implementation with ours. However, our less direct approach has the considerable advantage of not only producing a feasible approximation quickly, but also providing strictly improving solutions iteratively, making it suitable even for large-scale problems whose exact solution is not practical anymore. Additionally, our iterative approach can solve problems with unbounded edges, which cannot be reduced to a finite $b$-matching instance in one step.

### 2.3 Contributions

(1) We define the *Bi-directed Minimum-Deviation Flow* (Bi-MDF) problem – a generalization of Bi-MCF – as a convenient problem representation well-suited for our needs and demonstrate multiple ways of using it to express various quantization problems;
(2) A novel fast approximation method (*asymmetric double cover*) for Bi-MDF problems delivers feasible integer approximations as an extension to Hochbaum's monotonization procedure, which can be used directly or serve as initial solutions for exact solvers;
(3) A fast and straightforward rounding algorithm to obtain Bi-MCF problems with even right-hand sides and bounds from Bi-MDF problems that are suitable for computing integer solutions with double-cover-based solvers;
(4) A novel, easy-to-implement iterative refinement algorithm that solves Bi-MDF exactly, which relies on reductions to the Weighted Perfect Matching (WPM) problem, for which (open source) implementations are readily available;
(5) An open-source C++ library, *libSatsuma*[4], implements both our approximate and exact algorithms for general Bi-MDFs;
(6) Half-arc quantization as an expansion of the feasible region of quantization problems that is also especially well suited to handle separation constraints in Bi-MDF formulations.

Our contributions obviate the need for either commercial ILP/IQP solvers or the intricate implementation of a direct $b$-matching solver to solve T-mesh quantization problems while delivering vastly better runtime performance than more general-purpose ILP/IQP solvers achieve for the same inputs.

We demonstrate the use of Bi-MDF in meshing pipelines based on quadrangular T-mesh quantization [Campen et al. 2015; Lyon et al. 2021a] using either a greedy selection of separation constraints or our half-arc quantization extension. Furthermore, we publish an open-source extension to the *QuadWild* [Pietroni et al. 2021] implementation, which solves Bi-MDF instead of IQP problems, significantly improving its runtime. As an additional advantage, the now optional dependency on the Gurobi solver paves the way for packaging the software in open-source software packages such as Blender.

## 3 FLOWS IN BI-DIRECTED NETWORKS

In Section 1, we modeled a simple quantization problem as a Bi-MCF problem. To simplify modeling for more complex real-world problems, we will define the *Bi-directed Minimum Deviation Flow* (*Bi-MDF*) problem, a generalization of Bi-MCF problems more amenable to expressing quantization objectives.

We solve such Bi-MDF problems by first constructing feasible approximate solutions using reductions to a standard MCF problem via an intermediate Bi-MCF problem. An iterative refinement algorithm subsequently solves to optimality by utilizing a series of reductions towards a *Weighted Perfect Matching* (WPM) problem in each step.

A series of illustrations of the individual algorithmic steps for solving a simple Bi-MDF instance derived from a quantization problem (Mitchells' 2-1 radish, Figure 6) exemplify the reduction pipelines.

### 3.1 Bi-directed networks

As mentioned in the introduction, bi-directed networks generalize directed networks (graphs) by allowing edges with two heads or two tails in addition to usual edges with one head and one tail. The top row of Figure 4 illustrates the allowed edge types. Note the omission of uni-directed self-loops $v \longrightarrow v$ – they have no effect, as any flow through such edges moves an equal amount out of and into $v$.

[3] For all-even $b$, this can also be used to find the integer optimum

[4] Available at https://www.algohex.eu/publications/bimdf-quantization

*Definition 3.1.* A bi-directed network $(V, E, \sigma)$ consists of a vertex set $V = \{v_1, \ldots, v_n\}$, an edge set $E = \{e_1, \ldots, e_m\}$, and an incidence function $\sigma : V \times E \to \{-2, -1, 0, 1, 2\}$.

We identify the function $\sigma$ with a matrix $\sigma \in \{-2, -1, 0, 1, 2\}^{n \times m}$ for notational convenience. Viewed as an undirected (multi-)graph, $(V, E)$ may contain self-loops and multiple distinct edges joining the same pair of vertices. $\sigma(v, e)$ determines the incidence relationship of edge $e$ at vertex $v$: If $e$ and $v$ are not incident, $\sigma(v, e) = 0$. For a non-loop $e$, $\sigma(v, e) = 1$ if $e$ has a head, or conversely, $\sigma(v, e) = -1$ if $e$ has a tail at $v$. In case $e$ is a loop of $v$, a value of 2 denotes a positive loop with two heads, and a value of $-2$ corresponds to a negative loop with two tails.

These options are formalized succinctly as

$$\forall e \in E : \; 0 < \sum_{v \in V} |\sigma(v, e)| \leq 2, \tag{12}$$

i.e., each edge only has one or two ends. Note that this allows for non-loop edges that are incident to only one vertex with $\sigma(v, e) = 1$ – these we denote outer-edges [Chen 2021].

The matrix $\sigma$ belongs to the class of binet matrices [Appa and Kotnyek 2000; Appa et al. 2007; Kotnyek 2002].

Graphically, we represent an edge $e$ as

$v \longrightarrow w$, iff $\sigma(v, e) = -1$ and $\sigma(w, e) = 1$,
$v \rightarrow\!\leftarrow w$, iff $\sigma(v, e) = -1$ and $\sigma(w, e) = -1$,
$v \longleftrightarrow w$, iff $\sigma(v, e) = 1$ and $\sigma(w, e) = 1$,
$v \longrightarrow$ , iff $\sigma(v, e) = -1$ and $\forall v' \neq v : \sigma(v', e) = 0$, or
$v \longleftarrow$ , iff $\sigma(v, e) = 1$ and ——— " ———.

The latter two edge types correspond to the outer-edges mentioned earlier.

## 3.2 Flows in bi-directed networks

*Definition 3.2.* A *circulation* of a bi-directed network $(V, E, \sigma)$ is an assignment $f \in \mathbb{Z}^{|E|}$ of integer flow values to the network edges, such that the incoming and outgoing flows at each node cancel each other out:

$$\forall v_i \in V : \; \sum_{e_j \in E} \sigma(v_i, e_j) \cdot f_j = 0. \tag{13}$$

For a given circulation $f$, the amount of flow entering a node $n$ is called the *throughflow* $t(n)$ of the node:

$$t(n) := \sum_{\substack{e_j \in E, \\ \sigma(v_n, e_j) > 0}} \sigma(v_n, e_j) \cdot f_j. \tag{14}$$

A circulation is *conformally decomposable* if it is the sum of non-zero circulations which share the same sign on all common non-zero edges. Otherwise, we call it a *circuit flow*. As an example, Figure 9a shows a flow $g$ that is conformally decomposable into circuit flows $\frac{g}{2} + \frac{g}{2}$. Other decompositions include $g = g + 0$ and $g = 2g + (-g)$, but neither decomposition is conformal. It can be shown that the maximum node throughflow of a circuit flow never exceeds two [Chen et al. 2017; Kotnyek 2002]. This fact will be helpful in developing and analyzing our exact iterative solver.

*Definition 3.3.* A *pseudo-flow* of a bi-directed network $(V, E, \sigma)$ with lower bounds $l \in \mathbb{N}_0^{|E|}$ and upper bounds $u \in (\mathbb{N}_0 \cup \{\infty\})^{|E|}$ for the flow on each edge is an assignment $f \in \mathbb{R}^{|E|}$ that satisfies these bounds, i.e., $l \leq f \leq u$ holds component-wise.

*Definition 3.4.* A pseudo-flow $f$ is *feasible* (or just a *flow*) if the flow conservation condition

$$\forall v_i \in V : \; \sum_{e_j \in E} \sigma(v_i, e_j) \cdot f_j = b_i \tag{15}$$

holds for per-vertex demands $b \in \mathbb{Z}^{|V|}$, i.e., the net inflow for each node corresponds to its demand.

Note that the difference between two integer flows (i.e., flows in $\mathbb{N}_0^{|E|}$) is always a circuit. The equivalence to $\sigma \cdot f = b$ by treating $\sigma$ as a matrix shows the relation between flow problems and linear problems. In the theory of linear programming, vertices of the polytope forming the feasible region, determined by per-edge bounds and Eqn. 15 are called *basic feasible solutions*. In MCF problems on directed graphs, these basic feasible solutions are always integer due to the total unimodularity of the node-edge incidence matrix. In Bi-MCF problems, our matrix is not totally unimodular, but binet – and thus every basic feasible solution is half-integer [Appa and Kotnyek 2000; Appa et al. 2007; Bolker and Zaslavsky 2006].

*3.2.1 Elimination of outer-edges.* Bi-flow networks can be assumed to contain no outer-edges. This is not a proper restriction, as we can always obtain an equivalent problem without outer-edges using the following procedure:

Add an additional node $\partial$ with zero demand and an unbounded zero-cost tail-tail self-loop. Add $\partial$ to each outer-edge $e$ as second incident node with $\sigma(\partial, e) = 1$. With this construction, $\partial$ can only sink even amounts of flow. The parity sum of flows into $\partial$ always corresponds to the parity of the sum of all node demands. Therefore, if that parity is odd, we set the demand of $\partial$ to one instead.

## 3.3 Bi-directed minimum-cost flow

The *Bi-directed Minimum Cost Flow* problem (*Bi-MCF*) [Appa et al. 2007; Edmonds and Johnson 1970; Edmonds 1967; Gabow 1983; Lawler 1976] is to find a feasible flow $f$ that minimizes

$$E_{\text{mcf}}(f) = \sum_{e_i \in E} \omega_i \cdot f_i = \omega^T f. \tag{16}$$

for a weight function $\omega : E \to \mathbb{R}$. Further constraining the flow to be integer on every edge yields the *Integral Bi-MCF* problem.

## 3.4 Bi-directed minimum-deviation flow

Extending linear costs per edge to per-edge separable convex functions is a natural generalization of Bi-MDF problems. Some authors refer to this as a convex min-cost bi-flow problem [Medvedev and Brudno 2009]. However, that term neither conveys that the cost function is separable into per-edge convex functions nor that the problem actually is non-convex due to the implied integer constraints.

We will consider a slightly weaker extension, where the convex cost function $c_e$ of each edge $e$ must attain a (not necessarily unique) minimum value at a parameter we denote the *target* of this edge. This excludes functions such as $x \mapsto 1/x$ for unbounded edges. We can then understand $c_e$ to measure the cost of deviations from the

target $e$. Therefore we will succinctly refer to such problems as the *Bi-directed Minimum Deviation Flow* problem (*Bi-MDF*).

Formally, a Bi-MDF problem is a bi-directed flow network equipped with convex per-edge cost functions $c_e : \mathbb{R} \to \mathbb{R}$, with the goal of finding a feasible flow that minimizes

$$E_{\text{mdf}}(f) = \sum_{e \in E} c_e(f_i). \tag{17}$$

For $c_i(x) = \omega_i x$ we obtain Bi-MCF problems as special cases.

### 3.5 Reducing Bi-MDF to Bi-MCF

The difference between the Bi-MDF and Bi-MCF problems lies solely in the more limited cost function of Bi-MCF. While Bi-MDF allows for arbitrary convex per-edge costs, Bi-MCF costs are always linear.

Given a Bi-MDF instance $M$, we construct a Bi-MCF problem $M'$ which exactly represents the feasible region but whose cost function is equivalent merely in a region $g - U \le f \le g + U$ centered around a given pseudo-flow $g \in \mathbb{N}_0^{|E|}$ for a maximal deviation $U \in \mathbb{N}$. We introduce the deviation limit to keep the size of the resulting Bi-MCF problem finite and linear in the size of the Bi-MDF instance and $U$. Cost equivalence is only required to hold under minimization: Any optimal solution to the Bi-MCF problem that deviates from $g$ by not more than $U$ units on every edge corresponds to a Bi-MDF flow with identical cost.

We first create a modified *split* Bi-MDF problem centered around $g$ that is exactly equivalent to the original problem. In the second step, we transform it into Bi-MCF form.

We split up a flow $f$ into components $f = g + f^+ - f^-$, where $f^+, f^- \ge 0$ model positive and negative deviation from $g$. The bounds $l \le f \le u$ are reflected in bounds $f^- \le g - l$ and $f^+ \le u - g$.

Algebraically, $Cf = b$ becomes $Cf^+ - Cf^- = b - Cg$. Graphically, each original edge $e$ gets replaced by an identical edge $e^+$ (forming $C$) and a reversed edge $e^-$ (forming $-C$). We denote $b - Cg$ as flow demand vector $b'$: Applying $f^+$ and $f^-$ shall result in a net flow per vertex that counteracts the imbalance caused by applying $g$ on top of the existing right-hand side $b$. The total cost of flow through both edges is given by

$$c'_e(f^+, f^-) = c_{e^+}(f_e^+) + c_{e^-}(f_e^-) + c_e(g_e), \tag{18}$$

with the per-edge cost functions

$$c_{e^+}(f_e^+) = c_e(g_e + f_e^+) - c_e(g_e) \tag{19}$$
$$c_{e^-}(f_e^-) = c_e(g_e - f_e^-) - c_e(g_e). \tag{20}$$

Theorem 3.5. *The split Bi-MDF problem $M'$ is equivalent to $M$ under optimization.*

Proof. For $f_e^+ = 0$ or $f_e^- = 0$, $c'_e(f_e^+, f_e^-) = c_e(g_e + f_e^+ - f_e^-)$ holds, i.e., costs are equivalent when only one arc carries flow. It is left to show that when simultaneously sending flow in both directions, $c'_e(f^+, f^-) \ge c_e(g + f^+ - f^-)$ still holds, i.e., we cannot underestimate the objective function.

The slope of a secant to the left of a point on the graph of a scalar convex function cannot exceed the slope of a secant to its right side. Therefore, with $\delta \ge 0$, we have

$$\begin{aligned} & \frac{c_e(g_e + f^+ + \delta) - c_e(g_e + f^+)}{\delta} && \ge \frac{c_e(g_e) - c_e(g_e - \delta)}{\delta} \\ \iff \quad & c_e(g_e + f^+ + \delta) + c_e(g_e - \delta) && \ge c_e(g_e) + c_e(g_e + f^+) \\ \iff \quad & c'_e(f^+ + \delta, \delta) && \ge c'_e(f^+, 0) \end{aligned}$$

Due to symmetry, this suffices to show that $c_e$ is not under-estimated. □

We now have a Bi-MDF problem that computes the cost of deviations from $g$, in which every lower bound is zero, and $c_{e^\pm}(0) = 0$. This more restricted set of cost functions in the intermediate Bi-MDF now simplifies their representation in the Bi-MCF objective function.

As Bi-MCF per-edge costs are linear functions of the flow, assigning a positive cost value to each edge already yields a Bi-MCF that penalizes deviation from $g$. To represent the full range of convex cost functions, we need another transformation, where multiple parallel copies with a capacity of $u_e = 1$ and individual costs replace each edge $e$. This transformation procedure is well-established for MCF problems on directed graphs [Ahuja et al. 1984, 1993; Minoux 1986], and the constructions directly extend to bi-directed graphs.

Consider a convex function $c(x) : \mathbb{R}^{\ge 0} \to \mathbb{R}$, $c(0) = 0$ in the intermediate Bi-MDF. We define $\hat{c} : [0, U] \to \mathbb{R}$ as a piecewise linear approximation (cf. Figure 5):

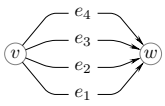
$$\hat{c}(x) = \min_{\mathbf{f}} \sum_{i=1}^{U} \overbrace{(c(i) - c(i-1))}^{\omega_i} f_i \tag{21}$$

$$\text{s.t. } x = \sum_{i=1}^{U} f_i, \tag{22}$$

$$0 \le f_i \le 1 \tag{23}$$

Due to the convexity of $c$, $\omega_i \le \omega_{i+1}$ holds. Therefore for a given $x$ in the domain of $\hat{c}$, the minimum value can be attained by setting $f_1 \dots f_{\lfloor x \rfloor} = 1$, $f_{\lceil x \rceil} = x - \lfloor x \rfloor$ and the rest to 0. This corresponds to an allocation of $x$ units among the variables $x_i$ that starts at the lowest-cost variable, using much capacity as possible before moving on to the next more expensive variable until $x$ is fully distributed.

For integer $x$, with this assignment, the sum in (21) simplifies to $\sum_{i=1}^{x} \omega_i = c(x) - c(0) = c(x)$, thus $\hat{c}$ is equal to $c$ for all integer arguments in its domain.




As we are only interested in integer solutions to the MCF, we can represent the objective function exactly in the range $[0, U]$, using up to $U$ duplicates $e_i$ of an edge, each one with a capacity of 1 and a cost of $\omega_i$, carrying $f_i$ units of flow. We pick $U_i$ as the minimum of a global $U$ parameter for the reduction and the per-edge capacity $u(e_i)$. If several $\omega_i$ are identical, we combine their corresponding arcs, summing up the capacities.

We can increase the capacity of the last edge $e_{U_i}$ by the remaining (possibly infinite) capacity of $u - U$ to exactly preserve the feasible region at the cost of not modeling costs correctly for flow values above $U$.

Thus for a Bi-MDF problem, a pseudo-flow $g$ and an upper bound $U$, we can construct an equivalent Bi-MCF (with the same number

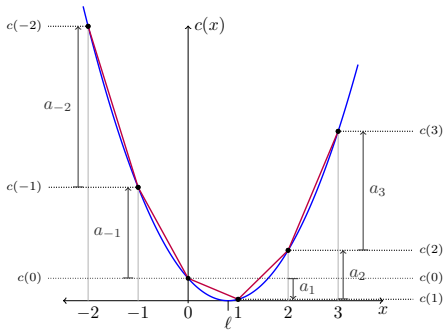



Fig. 5. Representing a convex function with flow costs, $g = 1$.

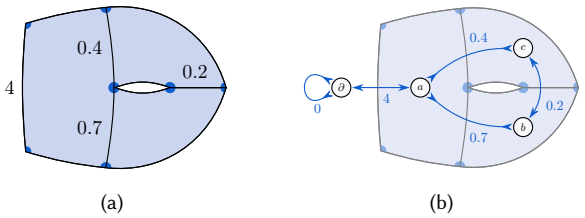



Fig. 6. Modeling example: (a) Mitchell's *2-1 radish* T-mesh [Mitchell 2021] with target lengths $\ell$ for some of the arcs. (b) One connected component of the corresponding Bi-MDF problem, obtained using the *simple* template. Lower bound $u$ is 1 for $\partial \longleftrightarrow a$, otherwise zero.

of nodes and up to $2U$ as many arcs, fewer for linear or absolute cost functions) that represents the same feasible set, preserves costs under minimization for flows $f$ with $g - U \le f \le g + U$, and approximates the objective function with constant slope beyond $\pm U$.

## 3.6 Bi-MCF and Bi-MDF approximation via double cover

We obtain approximate solutions for Bi-MDF problems using a double cover approach. These approximations can either serve as a feasible starting point for an exact solver (Section 3.7) or be used directly if speed is more important than accuracy.

The basic idea is to create a copy $v^-$ of each node $v$ with negated $\sigma$; this allows us to connect them using only directed edges by appropriately choosing negated and non-negated endpoints.

Hochbaum describes lifting a Bi-MCF problem $M = (b, l, u, \omega)$ on a bi-directed graph $G = (V, E, \sigma)$ to a MCF problem $M' = (b, l', u', \omega')$ on a directed graph $G' = (V', E')$ [Hochbaum 2004]. $G'$ consists of twice the amount of nodes and edges of $G$, and the solutions of $M'$ directly relate to the solutions of $M$. We refer to $M'$ as a *double cover* of $M$. Figure 7 illustrates the result of this construction.

For each node $v$ in $V$, $V'$ contains vertices $v^+$ and $v^-$. For each bi-directed edge $e_i \in E$, $E'$ has two edges as follows (cf. Figure 4):

- $e_i = v \longrightarrow w$: Add $e_i^0 = v^+ \to w^+$ and $e_i^1 = w^- \to v^-$
- $e_i = v \longleftrightarrow w$: Add $e_i^0 = v^- \to w^+$ and $e_i^1 = w^- \to v^+$
- $e_i = v \to\!\!\leftarrow w$: Add $e_i^0 = v^+ \to w^-$ and $e_i^1 = w^+ \to v^-$

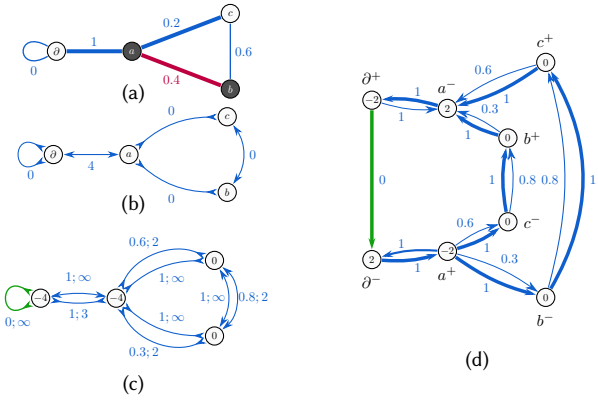



Fig. 7. Double cover construction: (a) T-join problem for finding a minimal $g$ adjustment for even right-hand sides. Grey nodes form the set $T$; edge labels indicate adjustment cost. The bold edges are a minimum spanning tree; the red edge is the approximate T-join solution. (b) Even pseudoflow $g$ resulting from parity adjustment to initial target lengths (c) Bi-MCF problem using $g$: Node labels indicate demands $b = \sigma g$, edge labels show cost $\omega$ and capacity $u$. (d) Induced asymmetric double cover. Node labels again indicate demand; edges are labeled with costs. Thin edges have $u = 1$, medium edges $u = 2$, and thick edges $u = \infty$.

Note that the sign $\sigma$ of the new edges at $u^+$ is $\sigma(u, e)$, while it is $-\sigma(u, e)$ at $u^-$ (for $u \in \{v, w\}$).

Vertex demands $b'$, edge costs $\omega'$, and bounds $l'$, $u'$ are inherited from the corresponding nodes and edges in $M$; however, demands are negated for $v^-$ nodes. A feasible flow $f$ on $M$ directly translates to a feasible flow $f'$ on $M'$ by setting $f'(e_i^0) = f'(e_i^1) = f(e_i)$. Conversely, a feasible flow $f'$ on $M'$ translates to a feasible half-integer flow $f$ on $M$ with $f(e_i) = \frac{1}{2}\left(f'(e_i^0) + f'(e_i^1)\right)$. The objective value is preserved with a scale factor of two in either direction. Translating an optimal solution on $M'$ thus yields an optimal solution for the real-valued relaxation of $M$, but not necessarily for $M$ when we include integer constraints.

*3.6.1 Even right-hand side and bounds.* We denote a Bi-MCF problem $M$ as *even* if its demands $b$, lower bounds $l$, and upper bounds $u$ all consist only of even entries. In this case, the basic feasible solutions of $M'$ are integer and even, and the corresponding BFS of $M$ are always integer.

We can obtain such a basic solution for $M'$ using the network simplex method and thus obtain an integer optimal solution for $M$.

Equivalently, we can halve all entries $b$, $l$, $u$ in the double cover, find any (potentially non-basic) integer optimal solution, and subsequently double the resulting flow on $M'$ for an integer solution. This perspective ultimately allows for our *asymmetric* generalization of the double cover approach.

*3.6.2 Parity adjustment.* In general, our problem instances do not only contain even values. To allow the creation of the double cover, we slightly modify the input Bi-MDF to ensure even parity before reducing it to Bi-MCF. This is different from directly adjusting the Bi-MCF problem: a naive rounding procedure here would generally not result in solutions feasible for the original Bi-MDF.

We perform the following constructions independently for each connected component of the bi-directed graph. Thus, w.l.o.g., we assume it to consist of a single connected component.

Let us first look into making the per-node demands $b$ even. Remember $b = -Cg$ for a pseudo-flow $g$, which we interpret as an initial guess. Let $g_e$ be an integer in the interval $[l_e, u_e]$ closest to the target length $\ell_e$. We aim to find an adjusted $\tilde{g} = g+\delta$ with a minimal change $\delta \in \{-1, 0, 1\}^{|E|}$, such that $\tilde{b} = -\sigma\tilde{g} = -\sigma(g + \delta)$ only has even entries. A simple solution would be choosing $\tilde{g}_e = \min_{l \le 2k \le u} c_e(2k)$, i.e., adjusting each entry of $g$ to the even number resulting in the lowest cost. However, an all-even $g + \delta$ is sufficient but not necessary for even $\tilde{b}$. In our application, it can be especially hurtful for coarse quadrangulations, as we could represent neither target lengths around 1 nor lower bounds of 1 properly. Additionally, this method requires $l < u$ for every edge with odd $g_e$, so a ±1 adjustment is even possible.

Note that the parity ($b_v \bmod 2$) of a node $v$ is determined by the parity of the sum of $g_e$ for incident edges $e$, independent of their orientation. Let $T \subseteq V$ be the set of vertices $v$ for which $\sigma \cdot g$ is odd. Any edge with any odd entry in $\sigma$ must have exactly two odd entries; therefore, by the handshake lemma [Euler 1736], $|T|$ must be even. Denote by $\hat{\sigma} = (V, E)$ the undirected node-edge incident matrix of the Bi-MDF network (excluding loops). Given a subset $J$ of edges such that the set of vertices of odd degree in the sub-graph $(V, J)$ is exactly $T$, adjusting the target lengths of the edges in $J$ by ±1 will result in a flow problem with even demands. Increasing or decreasing the target length by one unit is equivalent for parity, so for each edge, we can independently pick the preferred adjustment direction based on feasibility and cost function changes. To guide the search for a suitable $J$ beyond the defining hard constraint, we define a per-edge adjustment cost function $c_a : E \to \mathbb{R}$, $e \mapsto \min c_e(g_e \pm 1) - c_e(g_e)$. Finding a feasible $J$ that minimizes $c_a(J)$ is known as the *T-join* problem or *matching with parity constraints* [Edmonds and Johnson 1973].

Note that if the network admits any feasible flow $f$, a T-join exists: $\sigma f - b$ is zero; thus $\hat{\sigma} f - b$ is even, and the odd elements of $f - g$ form a feasible – but generally not optimal – T-join.

*Solving T-Join problems.* It is possible to determine an optimal $T$-join in polynomial time by computing a minimum-weight matching on a complete graph of $T$ with edge weights corresponding to costs of shortest paths in $(V, E)$ [Edmonds and Johnson 1973]. Berman et al. give a more efficient, albeit more complex, construction for sparse graphs [Berman et al. 1999].

In our application, we chose to use a very simple approximation algorithm instead to obtain a not necessarily optimal $T$-join in near-linear time. We compute a minimum-weight spanning tree using the adjustment costs $c_a$ as weights. For a given spanning tree with edges $S \subseteq E$, there is a unique $T$-join $J \subseteq S$, which we can determine in linear time. Figure 7a shows the T-join problem and its approximate solution.

Conceptually, this algorithm can be viewed as incrementally constructing $J$ while iteratively removing leaf nodes from the tree until it is empty. In each iteration, if the valence of $v$ in $(V, J)$ needs to be adjusted, we add its (sole) incident edge $e$ to $J$.

For an efficient implementation, we do not need to modify the tree but can instead find a suitable vertex ordering by depth-first search, as illustrated in Algorithm 1 in the appendix.

A linear algebra view on this algorithm is elucidating: A spanning tree of an undirected connected graph corresponds to a selection of $|V| - 1$ linearly independent columns of its node-edge incidence matrix $\hat{\sigma}$. This forms a basis $B$ for its column space, corresponding to all possible node adjustments. If we represent subsets $F$ of $E$ by an indicator vector $I_F \in \{0, 1\}^{|E|}$, then $J$ is a $T$-join iff $\hat{\sigma} \cdot I_J = I_T$ holds over the finite field $GF(2)$. If a $T$-join exists, the basis $B$ uniquely determines a solution. $B$ consists of $n$ rows but only $n - 1$ columns. We turn it into a full basis $B'$ by adding a special column with a one in an arbitrary row and zeroes everywhere else. As each column of $B$ only has two non-zero entries, we can permute the rows and columns of $B'$ into an upper triangular matrix and perform back-substitution to solve the system. The solution entry corresponding to our special column will be zero iff the desired adjustment is in the span of $B$.

Algorithm 1 implements this back-substitution directly in the graph representation without explicitly constructing $B$ or $B'$. Note that similar ideas of using a spanning tree as a vector space basis are part of various classic graph algorithms, such as the network simplex algorithm [Orlin 1997].

Note that in contrast to rounding each edge target to the nearest feasible even number, which requires being able to adjust every edge, this T-join-based parity adjustment scheme can easily be extended to handle fixed-flow edges with $l_e = u_e$ by excluding them from the MST.

In addition to adjusting $g$ for even $b$, we also need to adjust Bi-MDF bounds $l$ and $u$, such that each entry of $g-l$ and $u-g$ is even. We achieve this by incrementing entries of $l$ or decrementing entries of $u$ where necessary. This way, the feasible region is only shrunk and never expanded. Where $l_e + 1 = u_e$ – common in our representation of non-linear cost functions – this results in undesired fixed-flow edges. Therefore we modify our cost function representation to only produce bounds with even offset from $g$. Section 3.6.3 discusses how we minimize the impact this has on accuracy.

In what we coin the *Asymmetric Double Cover*, we later counteract the adjustment of Bi-MDF bounds in either $e^+$ or $e^-$, raising the lower or lowering the upper bound by one. By only re-adjusting one copy, the sum of both lower (or upper) bounds is the original un-adjusted bound, so any solution cannot violate the bounds of the original problem. The choice of which copy to adjust affects the feasible region, and it is unclear if there is any way to find a good choice a priori. In our implementation, we simply deterministically choose one of the two edges. In practice, this proved sufficient to roughly halve the approximation error incurred compared to the symmetric double cover (cf. Figure 16).

*3.6.3 Cost function approximation.* Previously, we translated non-linear Bi-MDF per-edge cost functions to Bi-MCF linear objectives using capacity-1 edges. For our double cover approach to work correctly, we need to model costs using capacity-2 edges. Effectively, we can now only choose the values of our piecewise-linear approximation $\tilde{c}_e$ for even integer deviations from $\tilde{g}$ instead of choosing them for every integer. Figure 8 shows the error incurred when

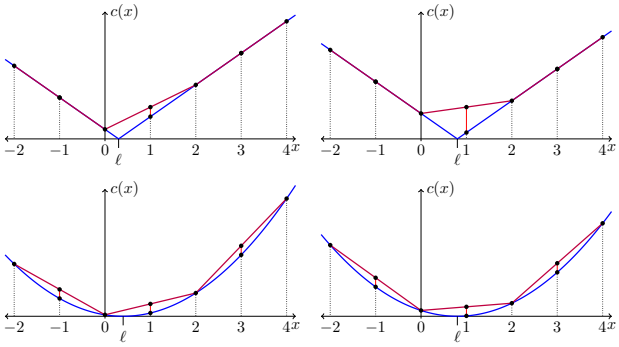



Fig. 8. Approximation (red) of per-arc objective functions $c_e(x)$ (blue) around $g = 0$ measuring the deviation from a target value $\ell$ using only capacity-two edges: Exact $c_e(x)$ values were chosen for the available sample points of an absolute (top row) and quadratic (bottom row) cost functions. $x = 0$ matches the integer minimizer in the left column, but in the right column, $x = 1$ would be the integer minimizer of $c_e$, but $x = 0$ becomes the minimizer of our approximation.

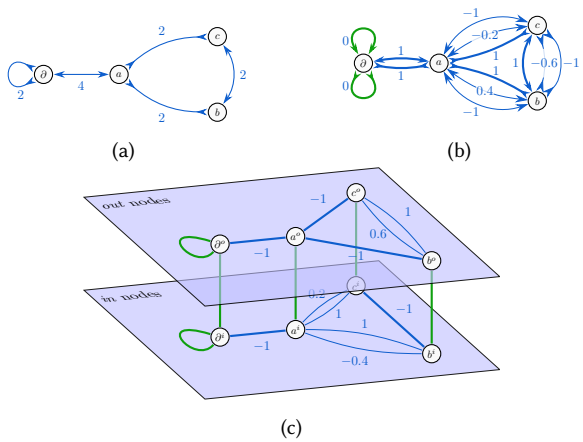



Fig. 9. (a) Initial flow $g$ from double cover approximation (b) Bi-MCF with maximal deviation $U = 2$ constructed using $g$ results in zero-demand problem for exact refinement. Edges labeled with costs; Bold edges are $u = 2$, others $u = 1$. (c) Resulting $b$-Matching problem. Edges drawn bold have capacity $u = 2$, others $u = 1$. Green color indicates zero-weight edges; labels show the weight of blue edges. Every node has a demand of 2.

choosing the exact $c_e(x)$ values for all available sample points. The error would be acceptable if we could always choose $g_e$ as an integer minimizer of $c_e$. However, in cases where our T-join-based rounding had to adjust $g$ from its preferred parity, the relative order of costs can become inverted, and the integer minimizer of $\tilde{c}_e$ changes. In our example, $\tilde{c}_e(1) < \tilde{c}_e(0)$, although $c_e(1) > c_e(0)$.

Instead of choosing $c_e(x)$ to be exact at sample points, we can distribute the error by computing more appropriate edge costs, e.g., minimizing some last-squared energy. With usual cost functions only parameterized by one parameter ($\ell$) and symmetry modulo 1, it is feasible to pre-compute edge weights that can quickly be interpolated by the solver, avoiding costly additional per-edge optimization steps.

Nevertheless, even in the best case, if we had to adjust $g$, the only way to avoid the mentioned cost inversion is to utilize zero-cost arcs, such that the two points will have identical cost – $\tilde{c}_e(1) = \tilde{c}_e(0)$ in our example.

We chose only to implement the first-mentioned weight assignment method, as we do not rely heavily on the quality of the approximation results and expect only marginal runtime improvements from an improved $f_0$ in our exact solver.

### 3.7 Exact Bi-MDF solutions by iterated refinement using $b$-matching

While we have so far focused on approximate solutions for Bi-MCF, such problems can also be solved exactly by a reduction to capacitated *Weighted Perfect b-Matching (WPbM)* [Edmonds and Johnson 1970; Pulleyblank 2012, 1973].

A (capacitated) perfect $b$-matching in an undirected graph $(V, E)$ is an assignment $x : E \to \mathbb{N}_0$, $x_i \leq u_i$ for edge capacities $u : E \to \mathbb{N}$ that fulfills degree constraints $b_v = \sum_{e:e\ni v} x_e$. Per-edge weights $\omega : E \to \mathbb{R}$ induce a matching cost $c(x) = \sum_e \omega_e x_e$ that is subject to maximization in the WPbM problem. $b$-matching can be defined with and without edge capacities, but either form easily reduces to the other.

In this section, we follow an approach by Edmonds [Edmonds 1967; Lawler 1976] to reduce a Bi-MCF problem to a WPbM problem whose size is linear in the Bi-MCF problem and quadratic in a parameter $M \in \mathbb{N}$ that corresponds to the maximum flow through a node.

We make a number of simplifying assumptions on the Bi-MCF problem that are fulfilled in our use case (an iterative refinement scheme detailed in Section 3.9).

- All node demands are zero (true for feasible $g$),
- Lower flow bounds $l$ are zero (by construction), and
- Upper flow bounds (capacities) $u$ are finite.

It follows that $f = 0$ is a valid flow. We refer to the referenced literature for more general constructions that allow for non-zero node demands.

For clarity of exposition, we perform the reduction in two separate steps: First, we create a modified Bi-MCF instance, which directly turns into a capacitated $b$-matching problem with a slight adjustment. Figure 9 illustrates its result. We replace each vertex $v$ with a pair of vertices $v^{\text{in}}$ and $v^{\text{out}}$. Each head (tail) of an edge at a vertex $v$ is changed to be incident to $v^{\text{in}}$ ($v^{\text{out}}$) instead:

$$\sigma'(v^{\text{in}}, e) := \max\{0, \sigma(v, e)\} \tag{24}$$

$$\sigma'(v^{\text{out}}, e) := \min\{0, \sigma(v, e)\} \tag{25}$$

We then add a new edge $e_v = v^{\text{out}} \to v^{\text{in}}$ with zero weight[5]. Its capacity $u(e_v)$ is chosen as the minimum of $M$ and the maximal sum of capacities of edges that have either heads or tails at $v$. Let the flow demand $b(v^{\text{in}}) = u(e_v)$, and $b(v^{\text{out}}) = -u(e_v)$. This is equivalent to the result of moving $u(e_v)$ units of flow from $v^{\text{in}}$ to $v^{\text{out}}$ along $e_v$.

[5]Using multiple edges with non-zero weights could be used instead to assign costs to node throughflow

The resulting problem is equivalent to the original Bi-MCF for all flows that do not exceed the node throughflow limit $M$. Observe that $v^{\text{in}}$ only has incident heads, and $v^{\text{out}}$ only has incident tails.

Finally, we flip the sign of all $v^{\text{out}}$ nodes by multiplying the corresponding rows of the constraint matrix with $-1$. This results in an equivalent problem, but all demands are now positive, and all edges are of the head-head type.

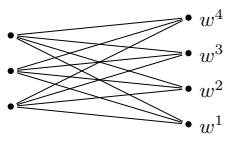
The constraint matrix thus corresponds to an undirected graph, while node demands correspond to $b$-matching degree constraints. The effect of this construction on the various edge types is depicted graphically in Figure 4.

With $b$-matching edge weights set to negated Bi-MCF edge costs, we obtain the equivalent WPbM problem.

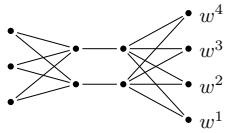
### 3.8 Reducing $b$-Matching to Matching

Efficient $b$-matching solvers have been described in the literature [Anstee 1987; Müller-Hannemann and Schwartz 2000, 2001], however their implementation is intricate. We are not aware of any such publicly available implementations. Therefore, we instead rely on a reduction of WPbM to the well-known *Weighted Perfect Matching (WPM)* problem, for which multiple efficient solvers are publicly available [Dezso et al. 2011; Kolmogorov 2009]. WPM can be considered the special case of WPbM where all entries of $b(v)$ are 1.

If any $b(v) > 1$, we perform a simple reduction, adapted from the work of Edmonds [Edmonds 1967; Lawler 1976]. For a WPbM problem $(V, E, \omega, b, u)$, we construct a WPM problem $(V', E', \omega')$. For each node $v_i \in V$, we add $b(v_i)$ copies $v_i^1 \dots v_i^b$ to $V'$. For any edge $e_{vw} \in E$ whose capacity can never be exceeded (i.e., $u(e_{vw}) \geq \max\{b(v), b(w)\}$), we fully connect the corresponding duplicates of $v$ and $w$ in a complete bipartite sub-graph and assign the weight $\omega(e)$ to each of these new edges.




Capacitated edges, for which this is not the case, are represented in the WPM by $u(e)$ pairs of nodes $(v_e, w_e)$, each connected by a zero-weight edge, where each $v_e$ is connected to all $v$-nodes with the original weight $\omega(e)$, and each $w_e$ is connected to all $w$-nodes using a zero-weight edge.




For every matching in the WPM problem, there is a matching of identical total weight for the WPbM problem, and vice versa.

### 3.9 Exact solution by iterative refinement

The reduction to WPbM requires upper limits $M$ for the amount of flow through each node. Tight bounds that do not exclude the optimal solution are not easily found a priori, and conservatively large bounds will result in large runtimes as the WPM problem size grows quadratically with $M$.

We side-step this issue by using these reductions for iterative refinement with small $M$ values instead of trying to obtain the optimal solution in a single step.

We start with an initial feasible flow $f_0$, which we obtain using the method detailed in Figure 3.6.3. Then we iteratively improve this solution by using $f_k$ as $g$ for the reduction of Bi-MDF to Bi-MCF (with $U = M$). Using our chain of reductions, we obtain $f_{k+1}$ as a new feasible flow. We iterate until the cost of $f_k$ has converged. Note that the actual flow values may not converge, as – for instance – zero-cost edges allow multiple optimal flows with identical costs.

Note that each but the last step results in a feasible flow that strictly improves the objective, and we can stop the iteration early to trade computation time against accuracy.

We still need to choose appropriate $M$ values for each step: smaller values will generally require more iterations, while larger values increase the cost of each iteration. Our experiments suggest that choosing a constant $M = 2$ gives a good runtime performance in practice. In contrast, more complex $M$ schedules, such as iterating with $M = 1$ until convergence and only then switching to $M = 2$, did not yield consistent runtime improvements for our inputs. Perhaps unsurprisingly, using $M > 2$ usually performed much worse.

First, we want to prove the termination of the algorithm after a finite number of steps.

Lemma 3.6. *If the cost function of each edge in a Bi-MDF problem has a minimal value, the number of distinct costs of flows that are below $c(f_0)$ is finite.*

This does not imply that the number of distinct flows with a lower cost is finite – for example, zero-cost circuits on unbounded edges may be added arbitrarily many times to a flow.

Proof. Consider a pseudo-flow $p$ that takes on a flow value $p_e$ of minimal cost $c_e(p_e)$ on each edge. The cost of changing the flow through an edge $e$, $c_e^\delta(x) = c_e(x) - c_e(p_e)$, is non-negative and convex. A positive or negative adjustment of the amount of flow through an edge is either zero-cost or the total adjustment in that direction is bounded by some value $m_e$ that would already bring the per-edge cost $c_e(x \pm m_e)$ above $c(f_0) - c(p)$. In both cases, the number of different cost values the edge can incur is finite. Therefore, the number of values the sum of all per-edge costs can attain must also be finite. □

Corollary 3.7. *The algorithm converges after a finite number of steps for any fixed $M$.*

Theorem 3.8. *If an iteration with $M \geq 2$ does not improve the objective function, the optimum has been attained.*

Proof. Assume the algorithm has converged to a sub-optimal $f$, i.e., the optimal solution of the Bi-MCF sub-problem has a cost of 0. Then there must be a negative-cost solution $\delta = \left|f^\star - f\right|$ that leads to a flow $f^\star$ of lower cost with a throughflow greater than $M$ on at least one node. This solution corresponds to a circulation on the Bi-MCF. However, any circulation that contains a node with a throughflow greater than 2 is not a circuit of the network and is conformally decomposable into a sum of circuits $\delta_i$, each of which is a feasible solution for the sub-problem. From $\omega^T(\sum_i \delta_i) = \omega^T\delta < 0$, it follows that at least one of the $\delta_i$ is a negative cost solution. Proof by contradiction. □

Corollary 3.9. *The refinement algorithm finds an optimal solution for any Bi-MDF problem in which each edge cost function has a minimal value.*

### 3.10 Bi-MDF simplification

In addition to splitting Bi-MDF networks into their connected components to solve them separately, we perform another pass of simplification that is cheap to compute and greatly reduces network size and runtime for many real-world Bi-MDF instances we encounter.

We denote a sequence of nodes $n_1, \ldots, n_k \in N$ a *chain* if there exists exactly one edge between $n_i$ and $n_{i+1}$ of arbitrary bi-direction for all $1 \leq i < k$, and every non-terminal node $n_2, \ldots, n_{k-1}$ has a demand of zero and exactly one head and one tail edge. Every edge in a chain has to carry the same amount of flow due to flow conservation (Eqn. 15). Additionally, upper and lower bounds for any edge in the chain apply to the whole chain. We collapse such chains into a single edge with appropriate upper and lower bounds and a cost function that is the sum of the individual edge cost functions. This construction could be extended to chains where nodes have non-zero demand; however, this is unnecessary in our case, as demands are always zero.

Lyon et al. perform a similar construction directly on T-meshes by only introducing a single ILP variable for dual edge chains [Lyon et al. 2021a]. Our approach achieves the same result but can be automatically applied to arbitrary Bi-MDF instances from any source.

## 4 MODELING QUANTIZATION AS BI-MDF PROBLEM

### 4.1 Preliminaries

*4.1.1 T-Mesh Inputs.* We consider manifold T-Mesh inputs, which we formalize as sets of vertices $\mathcal{V}$, undirected arcs $\mathcal{A}$, directed half-arcs $\mathcal{H}$, and patches $\mathcal{P}$. Figure 2 illustrates the utilized nomenclature.

An arc $a_{cd} \in \mathcal{A}$ is represented by a set of two distinct vertices $c, d \in C$: $a_{cd} = a_{dc} = \{c, d\}$. For each arc $a_{cd}$, there is a pair of opposing half-arcs $h_{cd} = c \to d$, $h_{dc} = d \to c$. For $x \to y$, $x$ is considered the *tail*, and $y$ is the *head* of the half-arc.

Each half-arc is either a boundary half-arc with no incident patch (in which case its opposite half-arc must not be a boundary) or is incident to exactly one patch. $p(a)$ denotes the incident patch, or $\partial$ in case of a boundary.

A path is a sequence of two or more distinct vertices $c_{i_0} c_{i_1} \cdots c_{i_n}$, where for each pair of successive vertices $c_{i_a} c_{i_{a+1}}$, a half-arc $c_{i_a} \to c_{i_{a+1}}$ exists in $\mathcal{H}$. We denote the first vertex in a path as its *tail* and its last vertex as its *head*. We identify such sequences of vertices with their associated half-arc sequences $(c_{i_0} \to c_{i_1})(c_{i_1} \to c_{i_2}) \cdots (c_{i_{n-1}} \to c_{i_n})$.

Finally, a patch $p \in \mathcal{P}$ is a tuple of paths $(p^0, \ldots, p^{n-1})$ making up its sides, such that the head of $p^k$ is the tail of $p^{k+1 \mod n}$ (denoted as *corners* of the patch), and the paths are otherwise vertex-distinct. We consider the number of sides (or, equivalently, corners) the valence of the patch. For quadrangular patches (valence $n = 4$), we consider the pairs of $p^k$ and $p^{k+2 \mod 4}$ *opposite sides*.

*4.1.2 T-Mesh Quantization.* We call an assignment $x : \mathcal{A} \to \mathbb{N}_0$ of non-negative integer lengths (subdivision numbers) to arcs of a T-mesh a *quantization*. We consider such a quantization *valid* if each T-mesh patch can be tessellated as a pure quadrangular mesh conforming to the edge subdivision numbers. This is equivalent to requiring the sum of side subdivisions of each patch to be even [Takayama et al. 2014; Tarini 2022].

With such hard constraints, we can guarantee the existence of a conforming quadrangulation. However, we still require a suitable objective function to obtain a high-quality quad mesh that follows user preferences such as mesh resolution.

Despite its attractiveness, it appears intractable to optimize for quad mesh quality metrics directly. Minimizing deviation of subdivision numbers from real-valued target lengths computed on the T-mesh commonly serves as a proxy objective for geometric quality.

Additional soft constraints can encourage certain patch tessellations that lead to simpler topological structure, e.g., preferring subdivision numbers that do not require excessive amounts of irregular vertices, or that lead to *aligned* singularity pairs, i.e., short separatrices connecting irregular vertices [Pietroni et al. 2021].

Admitting zero-length quantizations for edges is a powerful strategy that can significantly reduce the topological complexity of the resulting quad layout. However, this requires additional *separation constraints* to prevent collapsing singularities, feature entities, and boundaries in the parametric domain.

In their simplest form, they constitute lower bounds $x(a_i) \geq 1$ for some T-mesh arcs. This approach, however, often is too restrictive, and various kinds of non-convex min-one-path constraints are preferred [Campen et al. 2015; Lyon et al. 2021a,b]. Campen et al.'s QGP algorithm forbids paths between singularities (and, by extension, features and boundaries) where a signed sum of quantized offsets of either parametric direction is zero. This formulation captures the non-collapse condition precisely; however, the set of such paths grows very quickly with T-mesh size, therefore lazily checking them is expedient. The authors implement such a lazy approach by performing breadth-first graph searches during optimization to check if an iteration would violate the constraint.

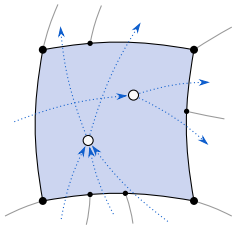
In contrast, Lyon et al. compute a small set $\mathcal{M}_1$ of straight min-one-paths a priori. At least one arc in each min-one-path must be quantized to a strictly positive value. These constraints are sufficient to guarantee separation but not necessary. However, in addition to preventing the aforementioned collapses, the authors' construction guarantees adherence to a given angular deviation threshold $\alpha$ between resulting mesh edge directions and directions implied by the parameterization.

We adopt their approach in our modeling strategy to quantize quadrangular T-meshes. It is more compatible with our Bi-MDF-based problem modeling, and allows us to directly employ our quantization implementation to implement the method of Lyon et al. without requiring an ILP solver.

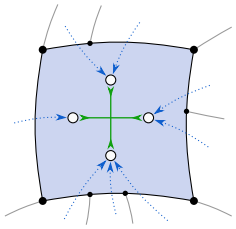
For our method of quantizing polygonal T-meshes, we extend the implementation of [Pietroni et al. 2021], which does not allow for zero-quantized arcs; therefore, no such explicit separation constraints are necessary.

### 4.2 Quadrangular T-meshes

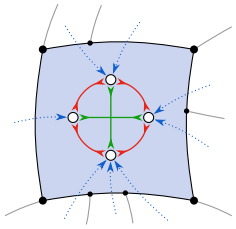
The first real-world application we cover is the quantization of T-meshes consisting entirely of quadrangular patches (i.e., with four corners) under strict consistency constraints, i.e., pairs of opposite sides must be quantized to the same lengths. We will allow quantizing arcs to zero length while adhering to non-zero path constraints $\mathcal{M}_1$, as discussed in Section 4.1.2. We individually map each T-mesh



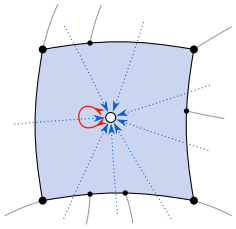
(a) The *simple* patch template enforces quantization equality of opposite sides.



(b) Split nodes enable measuring total throughflow, corresponding to quantized patch side length.



(c) Additional arcs (red) allow non-regular quantizations. Self-loops could be used to allow even more solutions.



(d) The *minimal* template only enforces the hard constraint of an even-sum boundary quantization, but offers very little control.

Fig. 10. A selection of bi-flow network templates for quadrangular patches. Starting with (b), edge orientations are chosen such that patches only receive inflow from their neighbors. This simplifies network assembly.

patch $p \in \mathcal{P}$ to a Bi-MDF sub-problem, with a mapping of each patch side to a node in the Bi-MDF network, considered its respective *side node*. Subsequently, we assemble the per-patch sub-problems by connecting the networks of neighboring patches.

*4.2.1 Per-patch construction.* The simplest case, equivalent to the formulation discussed in Section 1 and illustrated in Figure 10a, only consists of two nodes, each corresponding to a pair of opposite sides. Each node is the side node of two patch sides. With this basic construction, our network arcs always correspond to the quantized lengths of a single T-mesh arc, and we have no way of utilizing the total quantized length of a patch side in the objective function. This total side length corresponds to the node throughflow.

If, instead, we use one node per side and connect the opposite sides' nodes (Figure 10b), the flow through this inner edge corresponds precisely to the quantized length of the whole side. This is our preferred template, as it allows using that length both in the objective function and for min-one constraints. The edge orientations are chosen so that each side node receives inflow only, simplifying later patch assembly.

*4.2.2 Patch network assembly.* We will add a network edge $e_a$ for each arc $a \in \mathcal{A}$. For each of its half-arcs $h$ with an incident patch $p$, this edge is incident to the side node $n$ of the side $p^i$ that contains $h$. If we use the *simple* template, $e_a$ has a head or tail at $n$ depending on a predetermined orientation of $p$. For any other discussed template, it will always have a head, i.e., $\sigma(n, e_a) = 1$. This construction results in outer-edges for boundary arcs that we later eliminate, as discussed in Section 3.2.1. Figure 6 illustrates this construction using the *simple* template.

*4.2.3 Non-zero path constraints.* The first variant of handling such constraints we will consider is to replace the non-zero path constraints $\mathcal{M}_1$ with min-1 constraints for a set of network edges chosen heuristically such that their satisfaction implies non-zero path constraint fulfillment. A simple heuristic would be to pick one T-mesh arc per non-zero path and set its corresponding network edge's lower bound to one. However, this is not always necessary: Strict consistency constraints ensure that a min-one constraint on one arc or patch side implies min-one quantization of every opposite patch side and every $\mathcal{M}_1$ that fully contains such a patch side. Thus it is usually possible to construct a non-zero constraint set $C_1$ smaller than $|\mathcal{M}_1|$, consisting of only arcs and sides.

We explicitly capture this relationship by constructing a directed *Non-zero implication graph* $G_1 = (N_1, E_1)$. The node set $N_1$ is the disjoint union of non-zero paths $\mathcal{M}_1$, T-Mesh arcs $\mathcal{A}$ and sides $\mathcal{S}$. An edge $v \to w$ denotes that non-zero quantization of $v$ implies a non-zero quantization of $w$. We add $v \to w$ to $E_1$ in the following cases:

- an arc or side $v$ is fully contained in a side or path $w$,
- a path $v$ is a subset of a path $w$, or
- a side $v$ is opposite a side $w$ in a T-mesh patch.

Algorithm 2 then greedily computes a sufficient set of min-one constraints by adding sides that cover new paths from $\mathcal{M}_1$ until every path is covered. Deleting visited parts of the graph ensures linear runtime.

This procedure shrinks our feasible region but allows for efficient optimization and a simple Bi-MDF construction. We cover a more advanced approach in Section 4.3 that naturally models non-zero path constraints exactly.

Another (unexplored) option would be to iteratively solve and lazily add min-one constraints to counteract violated separation constraints until we reach a feasible solution.

*4.2.4 Relaxed conservation constraints.* Not every quadrangular T-mesh admits a valid quantization under hard quad patch regularity constraints [Myles and Zorin 2013]. In other cases, allowing some irregular quantized quad patches may improve objectives. If our downstream pipeline is capable of tessellating non-regular quantized quad patches, we can adjust our flow network construction to allow such solutions, choosing edge costs to balance objectives.

Figure 10c illustrates one modeling option, with additional arcs (marked in red) extending the feasible quantizations by allowing flows to not only pass straight through a patch but also connect arbitrary distinct patch sides. This variant does not yet allow all possible tessellatable (even-sum) quantizations, as a flow cannot use only one side of a patch twice. This could be facilitated by adding self-loops at the interior nodes or, for a very simple network at the cost of little control, by only using a single interior node (Figure 10d).

We would have to be careful when combining such relaxed quantization constraints with zero-length quantizations: Quad patches with opposite zero-quantized sides can always be collapsed, but irregular patches with some zero edges may lead to degenerate

configurations. One option to solve this is lazily adding additional constraints when such situations occur.

### 4.3 Half-arc quantization

In this section, we only consider the case of purely quadrangular T-meshes, i.e., every patch contains exactly four corners. The idea could, however, easily be extended to polygonal meshes.

A strictly positive arc quantization completely fixes the order of two T-junctions on either side of a T-mesh arc path. The introduction of zero-length arcs relaxes this and enables merging subsequent T-junctions, however, not reversing their order. This is sensible as long as we only consider T-junctions on the same side of the path, where a changed order would imply unwanted inverted elements. The same is not true when we merely change the order of two T-junctions on opposite sides of the path – in this case, the additional degrees of freedom may allow superior solutions. Figure 11 shows an example where the desired alignment is only possible due to this relaxation. This example is inspired by Brückler et al., who implement a similar relaxation in the three-dimensional case by allowing negative quantized lengths [Brückler et al. 2022a]. While sub-sides may have negative lengths, total side lengths must still be non-negative to prevent inversion. We present an alternative implementation: Instead of using negative lengths of sub-sides shared by patches on either side of the path, we quantize either side of an arc – its two half-arcs – independently, only constrained by an equal quantized sum on each side of the path.

If every T-mesh corner is a T-junction, the approaches are equivalent and respective quantizations can easily be translated. However, crossings necessarily form fixtures at which opposite arcs cannot be moved relative to each other, even when admitting negative solutions. Splitting a crossing into a pair of T-junctions enables the relaxation for one of the two opposite pairs, making the feasible region of both approaches equivalent again.

This is especially beneficial in conjunction with min-one-path constraints $\mathcal{M}_1$, which we can now model by virtually splitting the T-mesh along each such constraint path by introducing a two-gon patch (Figure 12a). We can then directly apply the min-one constraint to a network edge whose throughflow corresponds to the quantized length of the entire path. This approach also works when such min-one paths have partial overlaps, which can occur when using Lyon's formulations [Lyon et al. 2021a,b]. Figure 11 shows such a configuration. Therefore, half-arc quantization allows us to model such separation constraints exactly in Bi-MDF formulations without a heuristic to turn them into per-arc or per-side min-one constraints.

A half-arc quantization is a function $H : \mathcal{H} \to \mathbb{N}_0$ assigning lengths to T-mesh half-arcs, together with a collection of simple, straight, directed half-arc paths $H_p \subseteq \cup_{i\in\mathbb{N}}\mathcal{H}^i$. We consider $H$ valid, if

(1) Opposite half-arcs that are both not part of any path in $H_p$ are assigned the same length,
(2) For each patch, quantizations of opposite sides' half-arcs sum up to the same length, and
(3) For each path in $H_p$, the sums of quantized length on either side are equal

If we are more concerned with control over proximity to the original T-mesh than modeling separation constraints exactly, an alternative construction illustrated in Figure 12b enables fine-grained cost control over T-junction reordering. In this modified approach, we can assign costs to limit how far a half-arc quantization may be from an arc quantization. This does come at the cost of losing exactness in min-one path constraint modeling, as it forces us again to pick a sufficient set of min-one constraints, thus excluding some valid solutions. This may still be a good compromise in applications where fine quad meshes – and thus fewer quantizations to zero length – are desired.

After successful half-arc quantization, we adapt the T-mesh topology and update its embedding to obtain an arc quantization with non-negative quantized lengths. We describe our method for T-mesh adaptation and re-embedding in Appendix A.

Existing unmodified algorithms (e.g., [Campen et al. 2015; Lyon et al. 2021a]) can then process this adapted and re-embedded T-mesh to collapse zero-length arcs and obtain a valid quad layout. An integrated approach that directly operates on half-arc quantizations without intermediate conversion to an arc-quantized T-mesh may be beneficial for performance reasons. However, we did not explore that option due to additional implementation complexity.

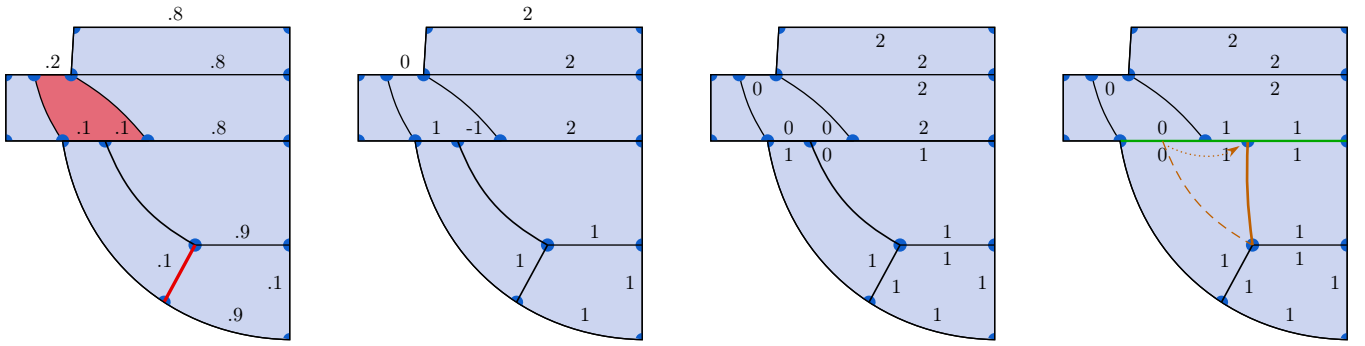
### 4.4 Polygonal T-meshes

We will now discuss using Bi-MDF to model the quantization of non-quadrangular T-meshes as they appear in the quad-meshing pipeline of Pietroni et al. [Pietroni et al. 2021].

Their approach produces a T-mesh consisting of patches ("charts") with valences between three and six. The associated quantization problem is to find an assignment of strictly positive integers lengths to T-mesh arcs ("sub-sides"), such that the sum of quantized lengths for each patch is even to allow local tessellation. In addition to this minimalistic hard constraint, soft constraints guide the quantization toward goals chosen to obtain high-quality results:

- *Isometry*: Deviations from target lengths should be minimal;
- *Regularity*: Quadrangular patches should have a regular quantization; other valence-$n$ patches shall only contain one irregular valence $n$ vertex, preferably strictly inside the patch; and
- *Singularity alignment*: The singular vertices of the quad meshes of nearby non-quad patches should be aligned, i.e., connected by a straight edge chain.

While isometry is straightforward and regularity can be encouraged by simple linear soft constraints, singularity alignment is more complicated. The authors search for pairs of non-quad patches that are either directly adjacent with a mutual side consisting of only a single T-mesh arc or connected through a chain of quadrangular patches, with each adjacent pair again sharing a common side (cf. [Pietroni et al. 2021], Figure 17). Under the assumption of regular quantization of all involved patches, the integer parametric position of the irregular vertices along the corresponding sides of the non-quad patches is linear in the assigned side lengths, leading to a single linear soft constraint per aligned pair.



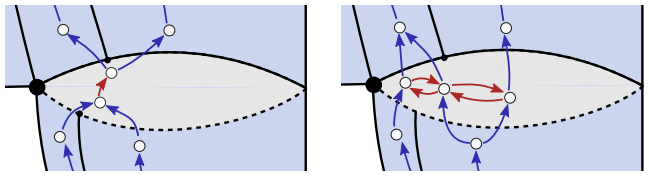

(a) T-mesh, arcs labeled with target lengths $\ell$. In a non-negative arc quantization, a min-one constraint (red arc) prevents the desired patch collapse

(b) Integer length arc quantization: a negative quantized arc signifies a change in the order of T-junctions

(c) Half-arc quantization. Some half-arc quantization values omitted where they are equal for both half-arcs of an arc.

(d) T-mesh adaptation moves the brown arc along the green path to obtain a non-negative arc quantization

Fig. 11. Half-arc quantization or negative lengths both enable quantizations that flip the order of T-junctions on opposite sides of a path, allowing the collapse of the undesired patch highlighted in (a) while respecting a min-one constraint.



(a) A pair of nodes allows using the quantized length (flow through red arc) of the path in objective and constraints.

(b) Cost functions on red arcs enable fine-grained costs for T-junction reordering

Fig. 12. Options for modeling half-arc quantization by inflating a path into a virtual two-gon. By construction, both sides of the trace will be quantized to the same lengths, as required for later deflation. Example taken from the central part of Figure 11.

Violated soft constraints may negatively influence the solution: while their associated cost penalties did not suffice to ensure constraint satisfaction, their influence skews the result. This is an issue, particularly for the alignment constraints, where only exact satisfaction is advantageous, and any degree of violation can be considered equally undesired. We mirror the authors' approach of performing a second round of optimization in which we drop unfulfilled alignment constraints. We perform the same search for singular pairs as implemented by Pietroni et al. and mark all involved T-mesh sides (i.e., those sides whose dual forms a chain between paired patches) and their constituent arcs as *paired* for different treatments in per-patch and global network construction.

While each unpaired arc is still be represented by one dual edge as before, we now associate paired arcs with two parallel dual edges. Their sum determines the quantized arc length, but their individual values (which we can view as the length of virtual sub-arcs) indicate the parametric position of the path connecting the involved irregular vertices. This representation obviates the linear expressions determining the singular vertex position and can be seen as a re-parameterization of variables: Instead of inferring the parametric position of singular vertices from side length variables, we represent these positions along arcs explicitly and obtain side lengths as simple sum of quantized virtual sub-arcs.

*4.4.1 Per-patch construction.* Patches with paired and unpaired sides require different templates: Each unpaired side should have a single side node, but for a paired side, we require two side nodes, one for each virtual sub-arc. We will define all per-patch templates to have two side nodes per side, which we collapse into a single side node for each non-paired side. This approach solely serves the clarity of exposition; in our implementation, we perform a direct construction without explicit collapses.

To derive suitable templates, consider the tessellation of a non-quad patch that fulfills the regularity condition and has its irregular vertex strictly inside the patch. It must be a regular subdivision of the quad tessellation obtained by connecting a point on each side with a central irregular vertex. This leads to the basic per-patch Bi-MDF templates illustrated in Figure 13 (only blue and green arcs). Every entering flow unit cancels out with a unit from another side, ensuring even length sums. Note that the amount of flow into either node of a side directly corresponds to the parametric position of the irregular vertex along the side. The resulting feasible set of quantizations directly corresponds to the conditions posed by Tarini [Tarini 2022], where flow amounts correspond to $s_i$ variables in his formulation.

An additional per-patch *emergency node* can receive an even sum of flow from its patch node via additional *emergency edges*, which it sinks via a tail-tail loop. Every flow unit through this loop sinks two flow units, maintaining even total parity.

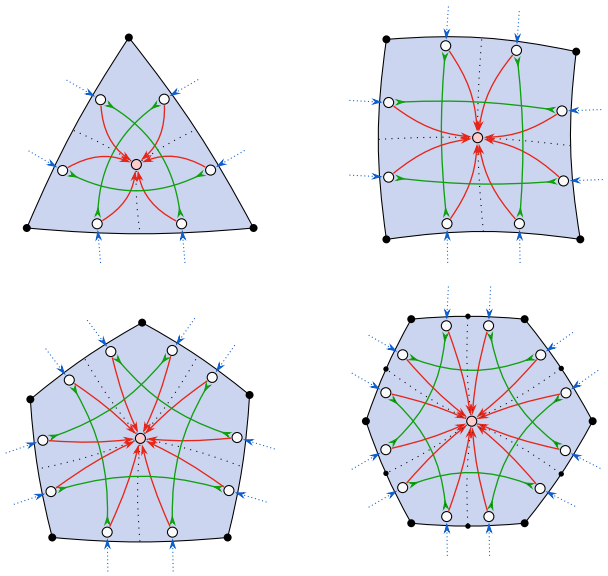


Fig. 13. Templates for polygonal patches of valence 3 to 6 with paired sides. Outer-edges (blue, dashed) connect to other patches or the boundary. An emergency node (pink) allows for non-regular quantizations by sinking even amounts of flow using a tail-tail self-loop (omitted here to reduce clutter) from emergency arcs (red) that have a target of zero, are unbounded, and carry a high weight.

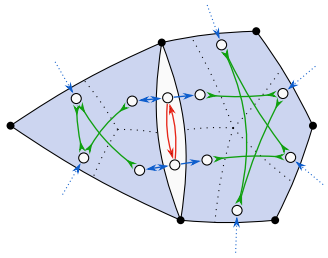


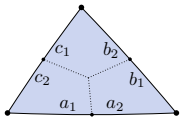
Fig. 14. Singularity alignment for paired patches: Side node pairs for all but the paired patch sides have been collapsed. Interior emergency nodes and arcs were omitted for clarity.

This construction enables non-regular quantizations that still result in an even boundary sum and offers flexibility in penalizing deviation from regular quantization.

*Penalizing singularities on patch boundaries.* Such quantizations are often undesired, as the singular vertices of neighboring patches may merge into one singularity with higher or lower valence than desired. This configuration incurs a penalty in the original IQP formulation, which we can also model in our approach. In a regular quantized non-quad patch, the singularity is located on the patch boundary exactly if the flow on any interior non-emergency edges (green) is zero. Therefore we assign a cost function with a positive cost for a flow of zero and zero cost for any positive flow.

*4.4.2 Patch network assembly.* We can assemble the complete Bi-MDF network from individual per-patch networks independently per T-mesh arc $a \in \mathcal{A}$. In the case of a non-paired arc, we connect the corresponding side nodes of the incident patches with a bi-directed head-head edge whose target length $\ell$ is the length of $a$. Boundary arcs are handled identically to the previous quadrangular T-mesh case. If the arc is paired, the incident sides of patches $p$ and $q$ each have two side nodes, $p_{0,1}$ and $q_{0,1}$, respectively. Directly connecting the corresponding nodes would enforce singularity alignment as a hard constraint. Therefore, we add two additional helper nodes $h_{0,1}$, connected by a pair of antisymmetric directed edges that allow penalized non-alignment. Connecting $p_i \longleftrightarrow h_i$ and $h_i \rightarrow q_{1-i}$ completes assembly (Figure 14).

*Target lengths for paired sides.* The QuadWild formulation handles target lengths of paired sides identically to non-paired sides. In our construction, we cannot use the sum of our virtual sub-arc quantizations in the objective function but must define targets for each. To this end, we compute the optimal real-valued parametric position of the singular vertex from the target edge lengths:




Given the regular subdivision of a triangle into three quadrilaterals, assume the lengths for $b$ and $c$ to be fixed to their real-valued target length. Then with $b_1 = c_2$, $a_1 = b_2$ and $a_2 = c_1$ and some basic algebra, we obtain $a_1 = \frac{1}{2}(b - c + a)$ and $a_2 = \frac{1}{2}(c - b + a)$. Thus we obtain suitable target lengths for either subside. It would also be possible to adjust the distribution of target length among the virtual sub-arcs away from this initial split in iterative re-solves based on the actual quantization result; however, we did not investigate this heuristic further.

## 5 EVALUATION

### 5.1 Implementation notes

We release an open-source C++ library, *libSatsuma*which implements representations, reductions, and solvers for the graph problems discussed in this work. We intentionally separated this implementation from the quantization applications to allow use as a black-box solver for Bi-MDF or $b$-Matching problems arising from any application. Its modular structure should make it attractive for further research into any of the involved sub-problems, and we will be happy to accept improvements.

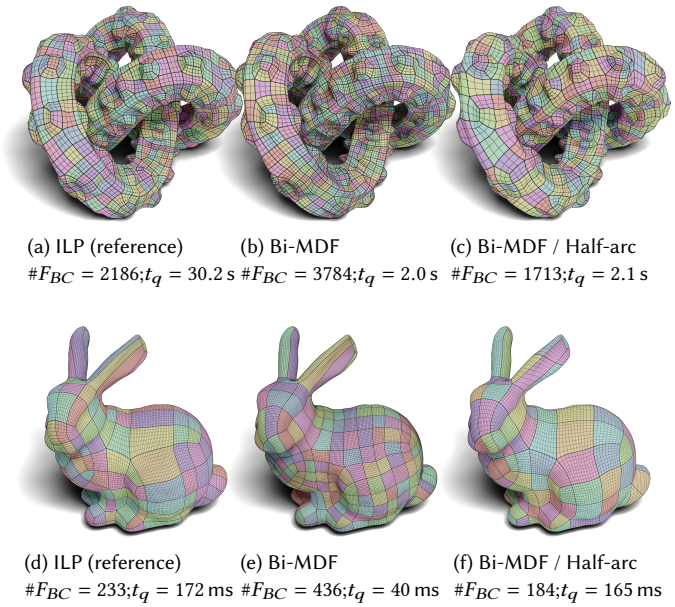
LibSatsuma utilizes the LEMON library [Dezso et al. 2011] for its implementation of directed and undirected graph data structures, solvers for the WPM and MCF problems, as well as various basic graph algorithms.

Additionally, we provide an interface to the Blossom-V solver for WPM [Kolmogorov 2009], which in our tests (cf. Figure 16) proved faster than the corresponding implementation in LEMON, but is not available under a free license.

### 5.2 Benchmark setup

We ran our tests on a server with two 64-core AMD EPYC 7742 CPUs and 2TiB of RAM, running Linux 5.10 and Gurobi 10.0.1. All timings are reported as *user* CPU time obtained using `getrusage()`.

We ran our methods on the *quadwild-300* dataset published together with QuadWild [Pietroni et al. 2021].



(a) ILP (reference) $\#F_{BC} = 2186; t_q = 30.2\,\text{s}$ (b) Bi-MDF $\#F_{BC} = 3784; t_q = 2.0\,\text{s}$ (c) Bi-MDF / Half-arc $\#F_{BC} = 1713; t_q = 2.1\,\text{s}$

(d) ILP (reference) $\#F_{BC} = 233; t_q = 172\,\text{ms}$ (e) Bi-MDF $\#F_{BC} = 436; t_q = 40\,\text{ms}$ (f) Bi-MDF / Half-arc $\#F_{BC} = 184; t_q = 165\,\text{ms}$

Fig. 15. Quad layouts via angle thresholded quantization on Knot100K ($\alpha = 45°$) and Bunny ($\alpha = 20°$). Models courtesy Aim@Shape & Stanford Computer Graphics Laboratory.

## 5.3 Quadrangular T-mesh quantization

We added quantization methods based on the modeling approaches discussed in Section 4.2 and Section 4.3 to *libIGM*, a currently non-public quad-meshing library. Our implementation supports both target-length-based quantization in the QGP pipeline [Campen et al. 2015], as well as the method introduced in [Lyon et al. 2021a] for coarse quad layouts with bounded angle deviation (*ATQ*). In both cases, we implement heuristic per-side constraints as described in Section 4.2.3 (denoted as *greedy*), as well as our half-arc quantization with T-mesh adaptation and re-embedding. We only use the per-patch template shown in Figure 10b.

We demonstrate results for coarse quad layout generation in Figure 15 and Table 1. While our results using greedy constraints are usually not as good as ATQ's results, our half-arc formulation almost always creates layouts consisting of fewer patches than ATQ. It does not have the same guarantees on angular deviation. However, in practice, we find its results geometrically satisfying. For the simple meshes that comprise a large part of the 300 dataset, ATQ shows fast to acceptable runtimes despite its exponential worst-case behavior; however, our method is still notably faster for worst-case meshes.

## 5.4 Polygonal T-mesh quantization

We extend[6] *QuadWild*, the open-source reference implementation of [Pietroni et al. 2021], to optionally use our Bi-MDF-based quantization method as described in Section 4.4 instead of creating IQP problems and solving them with Gurobi.

[6]implementation available at https://www.algohex.eu/publications/bimdf-quantization.

We match their objective function exactly with alignment constraints disabled and thus always obtain optimality even when the original implementation does not. With alignment constraints enabled, our objective differs slightly, and we cannot always achieve optimality w.r.t. the original objective, although the final meshes results are of comparable quality. Our implementation supports the same trade-off parameter $\alpha$ between isometry and regularity/alignment as the original implementation. We use $\alpha = 0.005$ in our experiments, which yielded higher-quality layouts than the default values of 0.01 and 0.02 and better demonstrates the effect of the different alignment terms. Note that while we find the presented per-patch templates to produce high-quality results, for purposes of a fair comparison, our code uses a slightly modified variant with carefully chosen weights to match the QuadWild objective. For details, we refer to our published source code.

QuadWild uses Gurobi to solve IQP problems, with two techniques to accelerate this process: It partitions large T-meshes into smaller clusters of patches and quantizes each cluster individually instead of computing a global optimum. Additionally, for a list of configurable pairs $\{(t_i, \delta_i)\}_i$ of runtime and duality gap thresholds, it aborts optimization early if, for one of the entries, the runtime is surpassed and the duality gap is under the threshold. A per-cluster runtime limit defaulting to 200s is in place as well. These heuristics greatly aid in typical situations where an optimal or near-optimal solution is already found but has yet to be proven optimal. However, either technique can prevent finding a globally optimal solution, and it is notoriously challenging to select thresholds appropriate for a wide input variety.

*5.4.1 Runtime and energies without alignment constraints.* For this test, we disabled alignment constraints, so no re-solve is necessary, and we can directly compare energies with our implementation. We compare runtimes and achieved energies of QuadWild, both single-threaded in its default configuration, as well as without partitioning, early-abort heuristic disabled and using eight threads and a soft memory limit of 64GiB ("full solve") on the *300* dataset. The results depicted in Figure 16 demonstrate that mesh partitioning and early abort heuristics work well for most meshes of this dataset and significantly reduce runtime compared to a full solve. However, for a small number of meshes, the approximation error exceeds 100%. Our Bi-MDF-based methods still prove to be dramatically faster while simultaneously computing the global optimum, thus even surpassing the full IQP solve in terms of energies achieved. 95% of quantizations finish after 0.02s (Bi-MDF approximation), 0.14s (Bi-MDF solve), 35.02s (IQP with early abort), and 88 hours (full IQP solve) CPU time, respectively.

*5.4.2 Results with alignment constraints.* Our approach of splitting each involved arc into two virtual sub-arcs (Section 4.4) allows measuring and optimizing for alignment. However, instead of optimizing for the total length of involved sides, we create target lengths for the virtual sub-arcs. A priori, it is unclear which isometry objective is favorable, and the two-step solving method employed in QuadWild complicates comparisons. Depending on the results of a first solve, it drops some soft alignment constraints, making comparing the final achieved energies futile. Therefore, we resort to comparing qualities of the final mesh for which the optimization energies are only

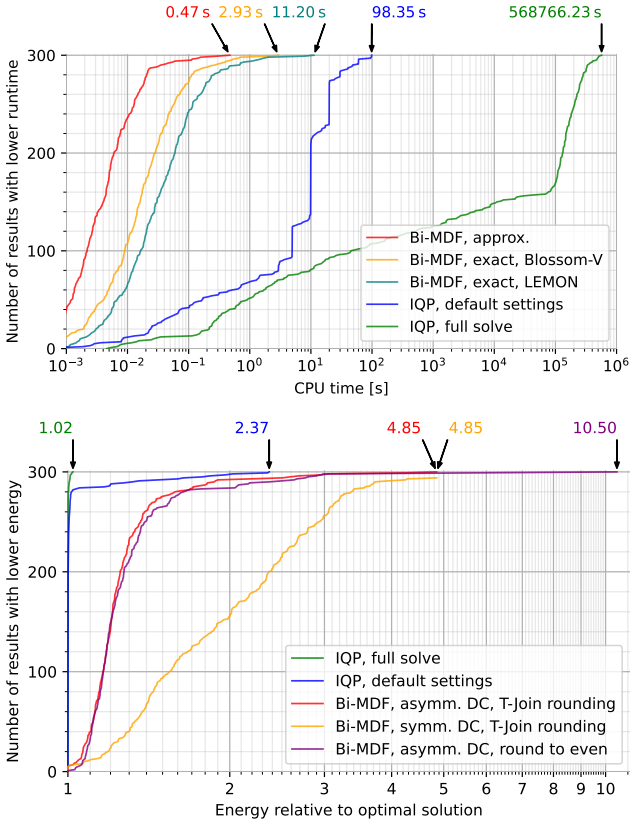



Fig. 16. Cumulative plots of achieved runtimes and energies for polygonal T-mesh quantization problems without alignment constraints on the *300* dataset. The full IQP solve reached its memory limit of 64GiB in 131 out of 301 cases.

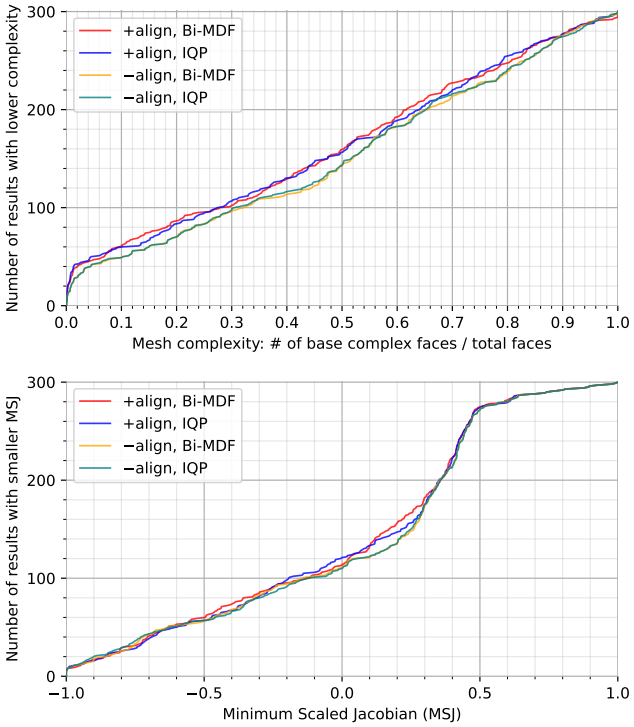
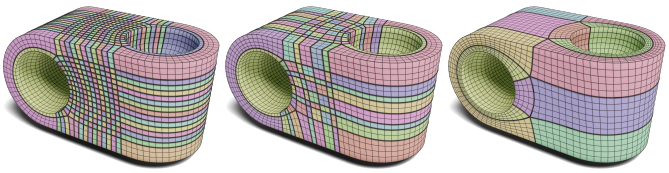



Fig. 17. Cumulative distribution of mesh complexity and minimum element quality on the *300* dataset when solving QuadWild problems with and without alignment term using our Bi-MDF formulation and QuadWild's IQP with default settings.



(a) Without alignment (b) IQP +align (c) Bi-MDF +align

Fig. 18. QuadWild's alignment constraints successfully produce a coarser base complex structure on UJOINT. Note that with the given alignment weight, in the first solver round, the IQP solver actually finds an optimum for its objective function in which 4/12 alignment constraints are not fulfilled and therefore dropped. The different formulation of alignment terms for paired sides in the Bi-MDF solver objective function changes has a different optimum in which all alignment terms are fulfilled at the expense of a higher isometry term. Therefore this example should not imply superiority of the Bi-MDF formulation. Depending on the alignment weight parameter, either solver can produce a spectrum of results.

proxies: element quality and topological complexity. We measure element validity and quality using the minimum scaled Jacobians (Figure 17). We observe comparable results between the different solving methods and slightly worse quality when alignment constraints are used.

To assess topological complexity, we compute base complex meshes and analyze the number of faces relative to the quad mesh. While this number usually would not be very meaningful – as it depends on input mesh complexity and element scale – we can compare it between different solvers that result in similarly sized meshes. The results (Figure 17, lower plot) again show comparable results between the IQP and Bi-MDF formulations.

Note that we discovered and fixed a bug in the QuadWild implementation for finding alignment pairs. All results use a version[7] which incorporates our bug fix.

## 5.5 Bi-MDF solver variants and parameters

We discussed several variants for solving Bi-MDF problems approximately. Our default configuration uses T-join-based rounding and our asymmetric double cover (DC). Comparing its performance to using the simple rounding scheme or symmetric DC, runtimes are nearly identical identical runtimes, however the asymmetric DC results in a greatly reduced approximation error (Figure 16). Using the simple round-to-even scheme does not significantly hurt result quality in most cases, but there are outliers with up to 950% error.

[7]https://github.com/nicopietroni/quadwild/pull/11

Our reduction of Bi-MDF to Bi-MCF (Section 3.5) depends on a parameter $U \in \mathbb{N}$, which determines the range in which it can faithfully approximate non-linearities in the objective function. In the refinement algorithm, we use $U = M$ so that the cost function is exact in the entire feasible region. However, in the initial double cover approximation, we must choose a finite $U$ despite the unbounded feasible region. This choice influences both the resources required to obtain the approximation and its quality. When using approximation results without refinement, we suggest evaluating the effect of $U$ in the specific application. However, we focus on using the approximations as starting point for the refinement algorithm that obtains an optimal solution independent of approximation quality; in this case, $U$ may only influence total runtime but not quality. We used a fixed $U = 5$ for the initial approximations in all our experiments without further experiments.

## 6 LIMITATIONS AND FUTURE WORK

There are extensive avenues for further research on the discussed topics.

First of all, while Bi-MDF solving performance is already satisfactory for our applications, there are numerous ways in which the runtime of our algorithms could be improved further.

*Algorithmic Improvements.* The individual WPM instances in the refinement process are highly related, so utilizing warm-start capabilities in the WPM solver may prove beneficial.

Our implementation of the refinement operation performs the various involved problem reductions sequentially and in isolation for simplicity of implementation. However, a more direct construction of the final WPM problem would enable further optimization options to produce smaller output problems. In our tests, the time spent computing the reductions and translating their solutions back is negligible compared to the WPM solving cost. However, this may not hold for problems in other domains, where a faster reduction process may make a more considerable difference.

The Bi-MDF instances from our applications also possess additional structure that we have not exploited yet: The problem graphs can have an embedding in the input T-mesh, a topological manifold surface of often low genus relative to their size. Results in solving weighted matching in planar graphs more efficiently based on the planar separator theorem [Lipton and Tarjan 1980] might extend to a generalized version of the planar separator theorem that considers graph genus [Gilbert et al. 1984].

*Novel applications for Optimization based on Bi-directed Flow.* With our publication of *libsatsuma* as a generic Bi-MDF solver, it appears very attractive to apply it to other kinds of problems that fit into the modeling framework, but thus far, have been solved with more generalized solvers at the cost of higher runtime and worse runtime predictability. Topics in quad mesh research in which we identify the potential for applying Bi-MDF modeling include quad mesh refinement [Lyon et al. 2020] and coarsening [Couplet et al. 2021]. Even if Bi-MDF is not a perfect fit for a problem, some existing iterative or branch-and-cut solvers could likely be sped up by simply using an initial solution obtained from a Bi-MDF relaxation. In particular, this applies to QGP and further variants of T-mesh quantization problems (e.g., [Lyon et al. 2021b]). We also see our results as a chance to re-visit the multitude of applications that rely on solving more specialized problems – e.g., minimum-cost flow on directed graphs – and evaluate how modeling using the generalized problems might extend their capabilities.

*Generalizations.* Finally, the question arises of how far Bi-MDF can be generalized while preserving fast solvability. Exploring the entire class of binet matrices – strictly larger than the incidence matrices of bi-directed graphs we have considered here [Appa and Kotnyek 2000; Kotnyek 2002] may be a good starting point. Other avenues may be generalized networks where edges carry gain multipliers [Hochbaum 2004] or flow equality constraints [Ahuja et al. 1999; Meyers and Schulz 2009]. The latter would be especially useful to efficiently solve the kind of volumetric quantization problems discussed in [Brückler et al. 2022a].

As individual Bi-MDF problems can be solved relatively quickly and could be extended with warm-start capabilities, we can also see their use in branch-and-bound schemes to cover more complex constraints than already afforded. Support for lazily adding constraints without full re-solves could be an excellent middle ground between efficiency and extended constraint support.

## 7 CONCLUSION

We define the general Bi-MDF optimization problem and explore novel, simple-to-implement algorithms to solve it both approximately and exactly.

We then show how different kinds of T-mesh quantization problems can naturally be expressed as Bi-MDF problems and demonstrate that this treatment yields high-quality results and favorable runtime performance compared to more general ILP or IQP solvers.

Such general solvers are an attractive and valuable tool for quickly modeling optimization problems. However, as we demonstrate, identifying additional structure and creating specialized solvers allows for vastly improved performance, enabling the solution of larger problems with lower resource usage.

Ideally, the identified structure is common enough that the resulting methods are useful for a variety of tasks beyond the originally covered problem. We believe this to be the case with Bi-MDF problems.

## ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 853343).

We would further like to thank Seth Pettie for making a scan of the elusive [Edmonds 1967] available on his website[8].

## REFERENCES

Ravindra K. Ahuja, J.L. Batra, and S.K. Gupta. A parametric algorithm for convex cost network flow and related problems. *European Journal of Operational Research*, 16(2): 222–235, 1984.

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Pearson, 1993. ISBN 978-0136175490.

Ravindra K. Ahuja, James B. Orlin, Giovanni M. Sechi, and Paola Zuddas. Algorithms for the simple equal flow problem. *Management Science*, 45(10):1440–1455, 1999. doi: 10.1287/mnsc.45.10.1440.

[8] https://web.eecs.umich.edu/~pettie/matching/

Richard P. Anstee. A polynomial algorithm for b-matchings: An alternative approach. *Inf. Process. Lett.*, 24(3):153–157, 1987. doi: 10.1016/0020-0190(87)90178-5.

Gautam Appa and Balázs Kotnyek. Binet matrices, an extension of network matrices. Technical report, Computational, Discrete and Applicable Mathematics, London School of Economics, December 2000. URL http://www.cdam.lse.ac.uk/Reports/Files/cdam-2000-19.ps.gz.

Gautam Appa, Balázs Kotnyek, Konstantinos Papalamprou, and Leonidas S. Pitsoulis. Optimization with binet matrices. *Oper. Res. Lett.*, 35(3):345–352, 2007. doi: 10.1016/j.orl.2006.04.003.

Piotr Berman, Andrew B. Kahng, Devendra Vidhani, and Alexander Zelikovsky. The t-join problem in sparse graphs: Applications to phase assignment problem in VLSI mask layout. In Frank K. H. A. Dehne, Arvind Gupta, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, volume 1663 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 1999. doi: 10.1007/3-540-48447-7_3.

Ethan D. Bolker and Thomas Zaslavsky. A simple algorithm that proves half-integrality of bidirected network programming. *Networks*, 48(1):36–38, 2006. doi: 10.1002/net.20117. URL https://doi.org/10.1002/net.20117.

Hendrik Brückler, David Bommes, and Marcel Campen. Volume parametrization quantization for hexahedral meshing. *ACM Trans. Graph.*, 41(4), jul 2022a. ISSN 0730-0301. doi: 10.1145/3528223.3530123.

Hendrik Brückler, Ojaswi Gupta, Manish Mandad, and Marcel Campen. The 3D Motorcycle Complex for Structured Volume Decomposition. *Computer Graphics Forum*, 2022b. ISSN 1467-8659. doi: 10.1111/cgf.14470.

Marcel Campen, David Bommes, and Leif Kobbelt. Quantized global parametrization. *ACM Trans. Graph.*, 34(6), October 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818140.

Beifang Chen. Conformal decomposition of integral flows on signed graphs with outer-edges. *Graphs and Combinatorics*, 37(6):2207–2225, Nov 2021. ISSN 1435-5914. doi: 10.1007/s00373-021-02344-3.

Beifang Chen, Jue Wang, and Thomas Zaslavsky. Resolution of indecomposable integral flows on signed graphs. *Discrete Mathematics*, 340(6):1271–1286, 2017. ISSN 0012-365X. doi: https://doi.org/10.1016/j.disc.2016.12.013.

Mattéo Couplet, Maxence Reberol, and Jean-François Remacle. *Generation of High-Order Coarse Quad Meshes on CAD Models via Integer Linear Programming*. 2021. doi: 10.2514/6.2021-2991.

CPLEX. V12. 1: User's manual for cplex. *International Business Corporation*, 46(53):157, 2009.

George Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T.J. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. John Wiley and Sons, 1951.

George B. Dantzig. *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963.

Balázs Dezso, Alpár Jüttner, and Péter Kovács. LEMON - an open source C++ graph template library. *Electron. Notes Theor. Comput. Sci.*, 264(5):23–45, 2011. doi: 10.1016/j.entcs.2011.06.003.

Jack Edmonds and Ellis L. Johnson. Matching: a well-solved class of integer linear programs. In *Combinatorial structures and their applications*, pages 89–92, 1970.

Jack R. Edmonds. An introduction to matching. Lecture notes, 1967. URL https://web.eecs.umich.edu/~pettie/matching/Edmonds-notes.pdf.

Jack R. Edmonds and Ellis L. Johnson. Matching, euler tours and the chinese postman. *Math. Program.*, 5(1):88–124, 1973. doi: 10.1007/BF01580113.

David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Computer Graphics Forum*, 2008. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2008.01288.x.

Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

Lester Randolph Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. ISBN 9780691651842.

Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 448–456, New York, NY, USA, 1983. Association for Computing Machinery. ISBN 0897910990. doi: 10.1145/800061.808776.

John R. Gilbert, Joan P. Hutchinson, and Robert E. Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984. doi: doi: 10.1016/0196-6774(84)90019-1.

Gurobi Optimizer. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

Dorit S Hochbaum. Monotonizing linear programs with up to two nonzeroes per column. *Operations Research Letters*, 32(1):49–58, 2004.

A.J. Hoffman and J.B. Kruskal. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems*, 38:223–246, 1956.

Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J. Guibas. Quadriflow: A scalable and robust method for quadrangulation. *Computer Graphics Forum*, 37(5):147–160, 2018. doi: https://doi.org/10.1111/cgf.13498.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):189:1–189:15, 2015. doi: 10.1145/2816795.2818078.

Vladimir Kolmogorov. Blossom V: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1:43–67, 07 2009. doi: 10.1007/s12532-009-0002-8.

Balázs Kotnyek. *A generalization of totally unimodular and network matrices*. PhD thesis, London School of Economics and Political Science, 2002.

E. Lawler. *Combinatorial optimization - networks and matroids*. Holt, Rinehart and Winston, New York, 1976.

Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. doi: 10.1137/0209046.

M. Lyon, M. Campen, and L. Kobbelt. Quad layouts via constrained t-mesh quantization. *Computer Graphics Forum*, 40(2):305–314, 2021a. doi: https://doi.org/10.1111/cgf.142634.

M. Lyon, M. Campen, and L. Kobbelt. Simpler quad layouts using relaxed singularities. *Computer Graphics Forum*, 40(5):169–179, 2021b. doi: https://doi.org/10.1111/cgf.14365.

Max Lyon, David Bommes, and Leif Kobbelt. Cost minimizing local anisotropic quad mesh refinement. *Comput. Graph. Forum*, 39(5):163–172, 2020. doi: 10.1111/cgf.14076.

Giorgio Marcias, Kenshi Takayama, Nico Pietroni, Daniele Panozzo, Olga Sorkine-Hornung, Enrico Puppo, and Paolo Cignoni. Data-driven interactive quadrangulation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 34(4):65:1–65:10, 2015.

Paul Medvedev and Michael Brudno. Maximum likelihood genome assembly. *Journal of Computational Biology*, 16(8):1101–1116, 2009. doi: 10.1089/cmb.2009.0047.

Carol A. Meyers and Andreas S. Schulz. Integer equal flows. *Operations Research Letters*, 37(4):245–249, 2009. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2009.03.006. URL https://www.sciencedirect.com/science/article/pii/S0167637709000467.

M. Minoux. Solving integer minimum cost flows with separable convex cost objective polynomially. In Giorgio Gallo and Claudio Sandi, editors, *Netflow at Pisa*, pages 237–239. Springer, 1986. doi: 10.1007/BFb0121104.

Scott A. Mitchell. High fidelity interval assignment. *Int. J. Comput. Geom. Appl.*, 10(4):399–415, 2000. doi: 10.1142/S0218195900000231.

Scott A. Mitchell. Simple and fast interval assignment using nonlinear and piecewise linear objectives. In Josep Sarrate and Matthew L. Staten, editors, *Proceedings of the 22nd International Meshing Roundtable, IMR 2013, October 13-16, 2013, Orlando, FL, USA*, pages 203–221. Springer, 2013. doi: 10.1007/978-3-319-02335-9_12.

Scott A. Mitchell. Incremental Interval Assignment by Integer Linear Algebra. *Proceedings of the 29th International Meshing Roundtable*, October 2021. doi: 10.5281/zenodo.5559025.

Rolf H. Möhring, Matthias Müller-Hannemann, and Karsten Weihe. Using network flows for surface modeling. In Kenneth L. Clarkson, editor, *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1995. San Francisco, California, USA*, pages 350–359. ACM/SIAM, 1995.

Rolf H. Möhring, Matthias Müller-Hannemann, and Karsten Weihe. Mesh refinement via bidirected flows: Modeling, complexity, and computational results. *J. ACM*, 44(3):395–426, May 1997. ISSN 0004-5411. doi: 10.1145/258128.258174.

Matthias Müller-Hannemann and Alexander Schwartz. Implementing weighted b-matching algorithms: Towards a flexible software design. *ACM J. Exp. Algorithmics*, 4:7–es, dec 2000. ISSN 1084-6654. doi: 10.1145/347792.347815.

Matthias Müller-Hannemann and Alexander Schwartz. Implementing weighted b-matching algorithms: Insights from a computational study. *ACM J. Exp. Algorithmics*, 5:8–es, dec 2001. ISSN 1084-6654. doi: 10.1145/351827.384250.

Ashish Myles and Denis Zorin. Controlled-distortion constrained global parametrization. *ACM Trans. Graph.*, 32(4):105:1–105:14, 2013. doi: 10.1145/2461912.2461970.

Ashish Myles, Nico Pietroni, and Denis Zorin. Robust field-aligned global parametrization. *ACM Trans. Graph.*, 33(4):135:1–135:14, 2014. doi: 10.1145/2601097.2601154.

Matthias Müller–Hannemann. High quality quadrilateral surface meshing without template restrictions: A new approach based on network flow techniques. *International Journal of Computational Geometry & Applications*, 10(03):285–307, June 2000. ISSN 0218-1959, 1793-6357. doi: 10.1142/S0218195900000176.

Stefano Nuvoli, Alex Hernandez, Claudio Esperança, Riccardo Scateni, Paolo Cignoni, and Nico Pietroni. Quadmixer: layout preserving blending of quadrilateral meshes. *ACM Trans. Graph.*, 38(6):180:1–180:13, 2019. doi: 10.1145/3355089.3356542. URL https://doi.org/10.1145/3355089.3356542.

James B. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Math. Program.*, 77:109–129, 1997. doi: 10.1007/BF02614365.

Chi-Han Peng, Michael Barton, Caigui Jiang, and Peter Wonka. Exploring quadrangulations. *ACM Trans. Graph.*, 33(1), Feb 2014. ISSN 0730-0301. doi: 10.1145/2541533.

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. Reliable feature-line driven quad-remeshing. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459941.

William Pulleyblank. Edmonds, matching and the birth of polyhedral combinatorics. *Documenta Mathematica*, 01 2012.
William R. Pulleyblank. *Faces of Matching Polyhedra*. PhD thesis, University of Waterloo, Waterloo, Canada, 1973.
Faniry H. Razafindrazaka and Konrad Polthier. Optimal base complexes for quadrilateral meshes. *Comput. Aided Geom. Des.*, 52:63–74, 2017. doi: 10.1016/j.cagd.2017.02.012.
Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. Perfect matching quad layouts for manifold meshes. *Comput. Graph. Forum*, 34(5):219–228, 2015. doi: 10.1111/cgf.12710.
Kenshi Takayama, Daniele Panozzo, and Olga Sorkine-Hornung. Pattern-Based Quadrangulation for N-Sided Patches. *Computer Graphics Forum*, 2014. ISSN 1467-8659. doi: 10.1111/cgf.12443.
T. K. H. Tam and Cecil G. Armstrong. Finite-element mesh control by integer programming. *International Journal for Numerical Methods in Engineering*, 36:2581–2605, 1993.
Marco Tarini. Closed-form quadrangulation of n-sided patches. *Computers & Graphics*, 107:60–65, 2022. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2022.06.015.
Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *CoRR*, abs/1605.04797, 2016. URL http://arxiv.org/abs/1605.04797.

## A T-MESH ADAPTATION AND RE-EMBEDDING

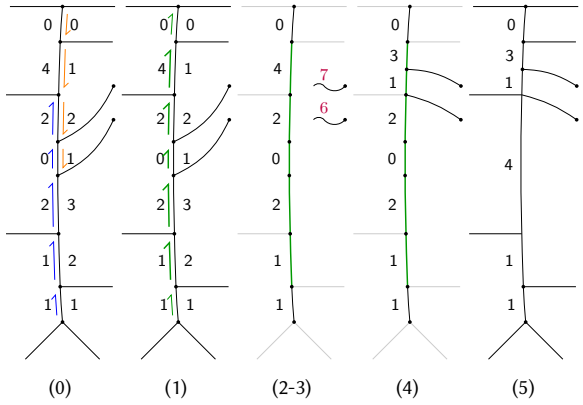



Fig. 19. T-mesh adaptation for a half-arc quantization: (0) A half-arc quantization with two overlapping paths (blue and orange). (1) The paths are combined into one path (green); only the bold sub-path needs correction. (2-3) The half-arc quantization of the left side gives an initial arc quantization; T-junctions are removed from the right-hand-side and subjects are left dangling, with target information (red) for their future location on the path. (4) An arc is split to create a nodes for the subjects' targets, subjects are re-connected with new T-junctions. (5) Valence-2 nodes collapse yields the final arc-quantized T-mesh.

Formally, we consider an embedding $e$ of a manifold T-Mesh $T = (\mathcal{V}, \mathcal{A}, \mathcal{P})$ into a manifold triangular base mesh $M = (V, E, T)$ a pair of functions $e_a : \mathcal{A} \to \cup_{i=0}^{\infty} E^i$ and $e_v : \mathcal{V} \to V$ that map T-mesh vertices to mesh vertices and T-mesh arcs to simple mesh edge paths. The embedding of a T-mesh arc $a$ may only be the empty tuple if a quantization $q : \mathcal{A} \to \mathbb{N}_0$ exists and $q(a) = 0$. If the embeddings $e_a(a_i)$, $e_a(a_j)$ of T-mesh arcs $a_i$, $a_j$ are incident to the same mesh vertex $v$, both $a_i$ and $a_j$ must be incident to a T-mesh node $n$ with $e_v(n) = v$. Additionally, $v$ must be an endpoint of both the embeddings.

We start out with a T-mesh $T$ with a valid half-arc quantization $H : \mathcal{H} \to \mathbb{N}_0$ and an embedding $e$ into $M$, and compute a modified T-mesh $T'$ with an arc quantization $q : \mathcal{A} \to \mathbb{N}_0$ and an embedding $e'$ into $M'$, which is a refinement of $M$. Our algorithm iteratively processes each trace, adapting the T-mesh and its embedding simultaneously, while constructing an arc quantization. We illustrate the T-mesh adaptation steps on an example in Figure 19.

(1) Each pair of overlapping paths is replaced by a longer directed path that contains the arcs of either path. This way, any half-arc not contained in more than one trace. For every path, we now analyze the half-arc quantization of its right-hand-side, extracting the shortest sub-path that requires correction: At the beginning and end, the half-arc quantization may already agree with the arc quantization and does not require correction. This may result in an empty sub-path, meaning no correction is required at all. For the remaining sub-path, T-junctions will have to be shifted along the path, such that the half-arc quantization agrees with the arc quantization.
(2) We initialize the arc quantisation $q$ as follows: Every arc that is not contained in a path gets assigned its length from the halfarc quantization (for a valid quantization, the quantization of both of its half-arcs has to agree). For every half-arc $h$ in a path, its arc $a(h)$ is assigned the quantized length of $h$. The result is that for every path, its left-hand-side is properly arc quantized, only its right-hand side may need correction.
(3) We remove all T-junctions from the right-hand side. The T-mesh arcs that formed these T-junctions – denoted *subjects* – are now "dangling" with only one node. We endow these subjects with a target value (cumulative sum of right-hand-side half-arc quantization values along the trace up to the subject's former T-junction) which determines the location of their future T-junction. We then iterate along the sub-path, keeping a cumulative sum of arc quantization values, which we compare to the subject targets to compute positions of new right-hand-side T-junctions. If there is no node where a new T-junction should be (i.e., inside an arc with a arc-quantization greater than one), we split this arc to introducing new T-mesh nodes. We also have to update the arc quantization to distribute the quantized length of the original arc among the resulting new arcs. In addition, we have to embed new arcs by splitting the arc embedding path. If there is no sufficient number of mesh vertices along the embedding of the original arc, we perform edge splits on the triangle mesh to create them. Now the dangling subject arcs are connected to their corresponding nodes in the sub-path, forming new T-junctions.
(4) To create new embeddings for the subject arcs, we consider the simply-connected triangle mesh region $R$ incident to the right-hand-side of the sub-path, bounded by the embeddings of any arcs that are not subjects, including the arcs of the path itself. This boundary is not considered part of $R$. Starting at one end of the sub-path, we iteratively re-embed the subjects by computing a triangle mesh path inside $R$ that leads from the embedding vertex of their non-T-junction node to the embedding vertex of the newly created T-junction. We compute this path using Dijkstra's algorithm. We may have to perform local face splits to accommodate this path, as the T-junction may not be reachable without using boundary edges of $R$ that are already used by the embeddings of other arcs. Finally, we split $R$ using the new embedding path, ensuring that path

computations for the remaining subjects cannot cross paths of already re-embedded subjects.

(5) Finally, we collapse valence-2 T-mesh nodes in the sub-path by merging the arcs and their embeddings.

The resulting topological T-mesh result is independent of trace processing order, its embedding however is not. While the embedding is always valid, the very local geometric operations can result in significant distortion if the half-arc quantization was "far" from an arc quantization, thus re-parameterization and smoothing steps will be beneficial for many downstream applications.

## B ALGORITHMS

**Input :** Vertices $V$,
Forest edges $F$,
Odd vertex set $T$ (with $|T|$ even)
**Output :** $J$: Set of edges, such that the degree of $v$ in $(V, J)$ is odd iff $v \in T$

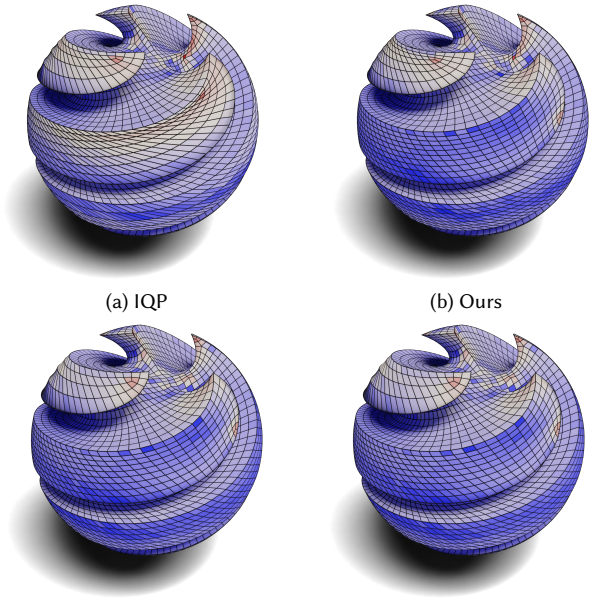
```
J ← ∅
forall v ∈ V do
    visited[v] ← false
    odd[v] ← (v ∈ T)
end
procedure Process(v)
    if visited[v] then
        return
    visited[v] ← true
    forall e = {v, w} ∈ F do
        Process(w)
        if odd[w] then
            J ← J ∪ {e}
            odd[v] ← ¬(odd[v])
    end
end
forall v ∈ V do
    Process(v)
end
return J
```

Algorithm 1. $T$-joins as unique subgraph of spanning forest

**Input :** non-zero implication graph $G_1 = (N_1, E_1)$,
non-zero arcs $\mathcal{A}_1$,non-zero paths $\mathcal{M}_1$,
T-mesh sides $\mathcal{S}$, side target lengths $\ell$
**Output :** Constraint set $C \subseteq \mathcal{S} \cup \mathcal{A}$

```
function Try-Cover(n)
    if n ∉ N_1 then
        return false
    r ← Depth-First-Search(G_1, n).visited
    G_1 ← G_1 − r                    // Remove covered nodes
    if r ∩ M_1 ≠ ∅ then
        M_1 ← M_1 − r
        return true
    return false
C ← A_1
forall a ∈ A_1 do
    Try-Cover(a)
end
forall s ∈ S  by decreasing  ℓ(s) do
    if M_1 = ∅ then
        return
    if Try-Cover(a) then
        C ← C ∪ {a}
end
return C
```

Algorithm 2. Computation of greedy min-one constraints from non-zero path constraints



(a) IQP (b) Ours

(c) IQP, no alignment (d) Ours, no alignment

Fig. 21. On sharp-sphere, enabling QuadWild's alignment term has negative impact on geometric quality while failing to produce a coarse base complex (cf. Table 2).

## C ADDITIONAL EVALUATION RESULTS

Table 1. Number of base complex faces and runtimes for coarse quad layouts with angle-thresholded quantization on simple meshes.

| | ATQ | | Ours, greedy | | Ours, half-arc q. | |
|---|---|---|---|---|---|---|
| $\alpha$ | $\#F_{BC}$ | $t_q$ [ms] | $\#F_{BC}$ | $t_q$ [ms] | $\#F_{BC}$ | $t_q$ [ms] |
| quadwild-300/Organic/duck | | | | | | |
| 10° | 459 | **128** | 1047 | 169 | **351** | 195 |
| 20° | 292 | 66 | 444 | **57** | **253** | 90 |
| 35° | 136 | 23 | 143 | **19** | **106** | 61 |
| 45° | 109 | **15** | 127 | 19 | **77** | 56 |
| quadwild-300/Organic/elk | | | | | | |
| 10° | 685 | 266 | 817 | **264** | **424** | 381 |
| 20° | 295 | 120 | 450 | **108** | **240** | 194 |
| 35° | 176 | 59 | 225 | **46** | **165** | 118 |
| 45° | 144 | 51 | 191 | **36** | **126** | 106 |
| quadwild-300/Organic/fertility | | | | | | |
| 10° | 365 | **234** | 549 | 249 | **306** | 539 |
| 20° | 186 | **94** | 227 | 113 | **153** | 170 |
| 35° | 123 | 55 | 153 | **41** | **114** | 112 |
| 45° | 119 | 34 | 145 | **31** | **103** | 94 |
| quadwild-300/Organic/bunny | | | | | | |
| 10° | 583 | 223 | 910 | **201** | **436** | 301 |
| 20° | 277 | 98 | 444 | **83** | **202** | 120 |
| 35° | 138 | 53 | 215 | **43** | **111** | 86 |
| 45° | 117 | 54 | 167 | **33** | **93** | 102 |
| quadwild-300/Organic/kitten | | | | | | |
| 10° | 692 | 187 | 1226 | **167** | **420** | 209 |
| 20° | 376 | **87** | 698 | 98 | **236** | 117 |
| 35° | 138 | **34** | 252 | 36 | **132** | 61 |
| 45° | 112 | 41 | 206 | **22** | **95** | 56 |
| quadwild-300/Mechanical/sculpt | | | | | | |
| 10° | 454 | **97** | 625 | 107 | **256** | 117 |
| 20° | 214 | 53 | 230 | **41** | **119** | 58 |
| 35° | 75 | 22 | 83 | **14** | **54** | 44 |
| 45° | 57 | 16 | 79 | **11** | **43** | 43 |

Table 2. QuadWild results for a selection of meshes taken from Figure 26 of [Pietroni et al. 2021]. The column *#i* shows the number of invalid faces (MSJ $< 0$), MSJ mean is computed only among valid elements.

| | | | | MSJ | | |
|---|---|---|---|---|---|---|
| | $\#F$ | $\#F_{BC}$ | #i | min. | mean | time [s] |
| quadwild-300/Mechanical/ujoint | | | | | | |
| IQP | 4537.0 | 686 | **0** | 0.53 | **0.96** | 0.102 |
| IQP (full) | 4537.0 | 686 | **0** | 0.53 | **0.96** | 0.588 |
| Ours | 4537.0 | 686 | **0** | 0.53 | **0.96** | **0.006** |
| IQP (+align) | 4457.0 | 183 | **0** | 0.48 | 0.96 | 0.222 |
| Ours (+align) | 4865.0 | **10** | **0** | **0.60** | 0.95 | 0.010 |
| quadwild-300/Mechanical/sculpt | | | | | | |
| IQP | 4095.0 | 1875 | **0** | **0.44** | **0.91** | 0.719 |
| IQP (full) | 4095.0 | 1875 | **0** | **0.44** | **0.91** | 1.939 |
| Ours | 4095.0 | 1875 | **0** | **0.44** | **0.91** | **0.005** |
| IQP (+align) | 4151.0 | 809 | **0** | 0.44 | 0.89 | 1.458 |
| Ours (+align) | 3974.0 | **460** | **0** | 0.42 | 0.88 | 0.022 |
| quadwild-300/Mechanical/bolt | | | | | | |
| IQP | 3885.0 | 468 | **0** | 0.62 | **0.95** | 4.936 |
| IQP (full) | 3885.0 | 468 | **0** | 0.62 | **0.95** | 15.45 |
| Ours | 3885.0 | 468 | **0** | 0.62 | **0.95** | **0.004** |
| IQP (+align) | 3862.0 | 166 | **0** | 0.62 | 0.95 | 0.331 |
| Ours (+align) | 3862.0 | **156** | **0** | **0.65** | 0.95 | 0.006 |
| quadwild-300/Mechanical/gear | | | | | | |
| IQP | 4961.0 | **1489** | **0** | **0.40** | **0.95** | 19.92 |
| IQP (full) | 4965.0 | 1495 | **0** | 0.40 | 0.95 | 201.6 |
| Ours | 4965.0 | 1495 | **0** | 0.40 | 0.95 | **0.009** |
| IQP (+align) | 5099.0 | 1507 | **0** | 0.40 | 0.95 | 16.33 |
| Ours (+align) | 5104.0 | 1514 | **0** | 0.40 | 0.95 | 0.016 |
| quadwild-300/Mechanical/sharp_sphere10k | | | | | | |
| IQP | 5950.0 | 5664 | **0** | **0.33** | **0.89** | 4.951 |
| IQP (full) | 5950.0 | 5664 | **0** | **0.33** | **0.89** | 76.48 |
| Ours | 5950.0 | 5664 | **0** | **0.33** | **0.89** | **0.016** |
| IQP (+align) | 5768.0 | **5338** | **0** | 0.23 | 0.87 | 4.939 |
| Ours (+align) | 6171.0 | 5607 | **0** | 0.04 | 0.89 | 0.032 |

Table 3. Quadwild examples for a selection of meshes taken from Figure 24 of [Pietroni et al. 2021]

| | #$F$ | #$F_{BC}$ | #i | MSJ min. | MSJ mean | time [s] |
|---|---|---|---|---|---|---|
| QUADWILD-300/MECHANICAL/BEARING_PLATE | | | | | | |
| IQP | 6474.0 | **1342** | **0** | **0.31** | **0.94** | 4.919 |
| IQP (full) | 6474.0 | **1342** | **0** | **0.31** | **0.94** | 288471 |
| Ours | 6474.0 | **1342** | **0** | **0.31** | **0.94** | **0.026** |
| IQP (+align) | 6448.0 | 1765 | 1 | -0.06 | 0.93 | 10.17 |
| Ours (+align) | 6451.0 | 1967 | **0** | 0.27 | 0.92 | 0.118 |
| QUADWILD-300/MECHANICAL/MID2FEM | | | | | | |
| IQP | 8109.0 | 4299 | **0** | **0.20** | **0.96** | 11.20 |
| IQP (full) | 8109.0 | 4299 | **0** | **0.20** | **0.96** | 317869 |
| Ours | 8109.0 | 4299 | **0** | **0.20** | **0.96** | **0.025** |
| IQP (+align) | 8017.0 | **2490** | **0** | 0.07 | 0.95 | 9.752 |
| Ours (+align) | 8406.0 | 2608 | **0** | 0.03 | 0.95 | 0.033 |
| QUADWILD-300/MECHANICAL/LEGO | | | | | | |
| IQP | 32323.0 | 9551 | **0** | 0.41 | 0.96 | 19.95 |
| IQP (full) | 32259.0 | 8741 | **0** | 0.42 | **0.96** | 109589 |
| Ours | 32259.0 | 8741 | **0** | 0.42 | **0.96** | **0.026** |
| IQP (+align) | 31936.0 | **7442** | **0** | **0.43** | 0.96 | 19.95 |
| Ours (+align) | 32842.0 | 9084 | **0** | 0.36 | 0.96 | 0.098 |
| QUADWILD-300/MECHANICAL/BLECH | | | | | | |
| IQP | 22955.0 | 10770 | **0** | **0.30** | 0.97 | 9.923 |
| IQP (full) | 22953.0 | 10502 | **0** | **0.30** | **0.97** | 121650 |
| Ours | 22953.0 | 10502 | **0** | **0.30** | 0.97 | **0.022** |
| IQP (+align) | 23181.0 | **7398** | **0** | 0.07 | 0.95 | 9.946 |
| Ours (+align) | 22589.0 | 8815 | 2 | -0.03 | 0.96 | 0.076 |
| QUADWILD-300/MECHANICAL/MECHANICAL02 | | | | | | |
| IQP | 3781.0 | 539 | **0** | 0.44 | 0.93 | 3.662 |
| IQP (full) | 3781.0 | 539 | **0** | 0.44 | 0.93 | 11.43 |
| Ours | 3781.0 | 539 | **0** | 0.44 | 0.93 | **0.009** |
| IQP (+align) | 3775.0 | **311** | **0** | **0.44** | **0.93** | 0.965 |
| Ours (+align) | 3506.0 | 356 | **0** | **0.44** | 0.92 | 0.012 |
| QUADWILD-300/MECHANICAL/MECHANICAL05 | | | | | | |
| IQP | 8085.0 | 1421 | **0** | 0.33 | 0.93 | 4.941 |
| IQP (full) | 8058.0 | 1376 | **0** | 0.33 | 0.93 | 246236 |
| Ours | 8058.0 | 1376 | **0** | 0.33 | 0.93 | **0.038** |
| IQP (+align) | 7887.0 | **1191** | **0** | 0.34 | **0.94** | 4.935 |
| Ours (+align) | 8071.0 | 1247 | **0** | **0.38** | 0.93 | 0.121 |
| QUADWILD-300/MECHANICAL/MECHANICAL08 | | | | | | |
| IQP | 5796.0 | **1** | **0** | **0.87** | **0.96** | 0.022 |
| IQP (full) | 5796.0 | **1** | **0** | **0.87** | **0.96** | 0.238 |
| Ours | 5796.0 | **1** | **0** | **0.87** | **0.96** | 0.006 |
| IQP (+align) | 5796.0 | **1** | **0** | **0.87** | **0.96** | 0.020 |
| Ours (+align) | 5796.0 | **1** | **0** | **0.87** | **0.96** | **0.004** |
| QUADWILD-300/MECHANICAL/JOINT | | | | | | |
| IQP | 4213.0 | 187 | **0** | 0.42 | 0.97 | 0.442 |
| IQP (full) | 4213.0 | 187 | **0** | 0.42 | 0.97 | 1.407 |
| Ours | 4213.0 | 187 | **0** | 0.42 | 0.97 | **0.007** |
| IQP (+align) | 4232.0 | **174** | **0** | **0.42** | **0.97** | 0.188 |
| Ours (+align) | 4069.0 | 281 | **0** | 0.02 | 0.97 | 0.008 |
| QUADWILD-300/MECHANICAL/BAMBOO_PEN | | | | | | |
| IQP | 10073.0 | 2279 | **2** | **-0.15** | 0.94 | 2.941 |
| IQP (full) | 10073.0 | 2279 | **2** | **-0.15** | 0.94 | 17.15 |
| Ours | 10073.0 | 2035 | **2** | -0.33 | 0.94 | **0.012** |
| IQP (+align) | 10078.0 | 2634 | 3 | -0.33 | 0.94 | 1.502 |
| Ours (+align) | 10118.0 | **1073** | 3 | -0.40 | **0.94** | 0.025 |

Table 4. Quadwild examples for a selection of meshes taken from Figure 25 of [Pietroni et al. 2021]

| | #$F$ | #$F_{BC}$ | #i | MSJ min. | MSJ mean | time [s] |
|---|---|---|---|---|---|---|
| QUADWILD-300/MECHANICAL/DELTA_ARM_BASE | | | | | | |
| IQP | 12131.0 | 3133 | **0** | 0.36 | 0.96 | 59.92 |
| IQP (full) | 12040.0 | 3920 | **0** | 0.41 | 0.96 | 115097 |
| Ours | 12040.0 | 3920 | **0** | 0.41 | 0.96 | **0.013** |
| IQP (+align) | 11914.0 | 2885 | **0** | 0.34 | **0.96** | 59.87 |
| Ours (+align) | 11116.0 | **2165** | **0** | **0.41** | 0.94 | 0.029 |
| QUADWILD-300/MECHANICAL/ROD | | | | | | |
| IQP | 3915.0 | 2093 | **0** | 0.21 | **0.94** | 0.316 |
| IQP (full) | 3915.0 | 2093 | **0** | 0.21 | **0.94** | 1.209 |
| Ours | 3915.0 | 2093 | **0** | 0.21 | **0.94** | **0.004** |
| IQP (+align) | 3740.0 | 1648 | **0** | **0.33** | 0.94 | 0.290 |
| Ours (+align) | 4473.0 | **274** | **0** | 0.10 | 0.94 | 0.014 |
| QUADWILD-300/MECHANICAL/ROLLING_STAGE100K | | | | | | |
| IQP | 7873.0 | 2022 | **0** | 0.51 | **0.95** | 29.90 |
| IQP (full) | 7873.0 | 2022 | **0** | 0.51 | **0.95** | 100970 |
| Ours | 7873.0 | 2022 | **0** | 0.51 | **0.95** | **0.016** |
| IQP (+align) | 7762.0 | 1053 | **0** | 0.51 | 0.95 | 30.55 |
| Ours (+align) | 7872.0 | **959** | **0** | **0.52** | 0.95 | 0.055 |
| QUADWILD-300/ORGANIC/3_HOLES | | | | | | |
| IQP | 5038.0 | **503** | **0** | **0.61** | **0.94** | 1.326 |
| IQP (full) | nan | nan | **0** | **0.61** | **0.94** | 3.988 |
| Ours | 5038.0 | **503** | **0** | **0.61** | **0.94** | **0.006** |
| IQP (+align) | 5127.0 | 693 | **0** | 0.60 | 0.94 | 1.441 |
| Ours (+align) | 4941.0 | 693 | **0** | 0.59 | 0.94 | 0.007 |
| QUADWILD-300/MECHANICAL/AIRCRAFT | | | | | | |
| IQP | 5859.0 | 1553 | **1** | -0.73 | **0.93** | 19.89 |
| IQP (full) | 5860.0 | 1476 | 2 | -0.38 | 0.93 | 6561 |
| Ours | 5860.0 | 1476 | 2 | -0.38 | 0.93 | **0.018** |
| IQP (+align) | 5637.0 | 973 | 2 | **-0.38** | 0.93 | 9.934 |
| Ours (+align) | 5739.0 | **878** | 2 | -0.72 | 0.92 | 0.032 |
| QUADWILD-300/MECHANICAL/SYDNEY | | | | | | |
| IQP | 6095.0 | 728 | **0** | **0.20** | **0.93** | 4.935 |
| IQP (full) | 6095.0 | 728 | **0** | **0.20** | **0.93** | 38.58 |
| Ours | 6095.0 | 728 | **0** | **0.20** | **0.93** | **0.017** |
| IQP (+align) | 5900.0 | **536** | 1 | -0.47 | 0.92 | 4.963 |
| Ours (+align) | 6170.0 | 626 | **0** | 0.19 | 0.93 | 0.034 |