Towards Pareto-optimality with Multi-level Bi-objective Routing: A Summary of Results

Mingzhou Yang
University of Minnesota
Minneapolis, USA
yang7492@umn.edu

Ruolei Zeng
University of Minnesota
Minneapolis, USA
zeng0208@umn.edu

Arun Sharma
University of Minnesota
Minneapolis, USA
sharm485@umn.edu

Shunichi Sawamura
University of Minnesota
Minneapolis, USA
sawam002@umn.edu

William F. Northrop
University of Minnesota
Minneapolis, USA
wnorthro@umn.edu

Shashi Shekhar
University of Minnesota
Minneapolis, USA
shekhar@umn.edu

ABSTRACT

Given an origin, a destination, and a directed graph in which each edge is associated with a pair of non-negative costs, the bi-objective routing problem aims to find the set of all Pareto-optimal paths. This problem is societally important due to several applications, such as route finding that considers both vehicle travel time and energy consumption. The problem is challenging due to the potentially large number of candidate Pareto-optimal paths to be enumerated during the search, making existing compute-on-demand methods inefficient due to their high time complexity. One way forward is the introduction of precomputation algorithms. However, the large size of the Pareto-optimal set makes it infeasible to precompute and store all-pair solutions. In addition, generalizing traditional single-objective hierarchical algorithms to bi-objective cases is non-trivial because of the non-comparability of candidate paths and the need to accommodate multiple Pareto-optimal paths for each node pair. To overcome these limitations, we propose Multi-Level Bi-Objective Routing (MBOR) algorithms using three novel ideas: boundary multigraph representation, Pareto frontier encoding, and two-dimensional cost-interval based pruning. Computational experiments using real road network data demonstrate that the proposed methods significantly outperform baseline methods in terms of online runtime and precomputation time.

CCS CONCEPTS

• **Applied computing** → **Multi-criterion optimization and decision-making**; • **Information systems** → **Geographic information systems**.

KEYWORDS

Bi-objective routing, Pareto optimality, Spatial query processing, Spatial algorithms

IWCTS'24, October 29-November 1 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1151-0/24/10…$15.00
https://doi.org/10.1145/3681772.3698215

ACM Reference Format:
Mingzhou Yang, Ruolei Zeng, Arun Sharma, Shunichi Sawamura, William F. Northrop, and Shashi Shekhar. 2024. Towards Pareto-optimality with Multi-level Bi-objective Routing: A Summary of Results. In *17th ACM SIGSPATIAL International Workshop on Computational Transportation Science GenAI and Smart Mobility Session (IWCTS'24), October 29-November 1 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3681772.3698215

1 INTRODUCTION

Given a graph in which each edge has a pair of non-negative costs, the bi-objective routing (BOR) problem aims to identify the set of all Pareto-optimal paths (also known as the Pareto frontier, the non-dominated set) between an origin and a destination. To compare paths with two (conflicting) costs, the dominance relation between the paths is defined as follows [6]: path p dominates another path p' if both components of the cost of p are less than or equal to the corresponding components of the cost of p', and their costs are not the same. For example, consider the spatial network shown in Figure 1 and the origin n_0 and destination n_7. Among the five paths connecting the origin and destination, the Pareto frontier contains two non-dominated paths: $[n_0, n_3, n_5, n_7]$ and $[n_0, n_1, n_3, n_5, n_7]$, with costs (10, 17) and (11, 16), respectively. By comparison, a path like $[n_0, n_2, n_4, n_6, n_7]$ with cost (18, 20) is not Pareto-optimal, since it is dominated by path $[n_0, n_3, n_5, n_7]$ with cost (10, 17).

The computation of the complete Pareto frontier in BOR problems is crucial as it provides a comprehensive spectrum of optimal solutions. Given that users' preferences vary and are typically unknown to the computing system, the most desirable solution can differ. Thus, considering only a subset of the Pareto frontier risks overlooking the solution that best aligns with a particular user's preferences. Drivers of electric vehicles, for example, are keenly interested in routes that minimize battery drainage as well as travel time. Different drivers (e.g., eco-conscious drivers, drivers in a hurry, etc.) may be interested in different subsets of the Pareto-optimal paths at different times. The BOR problem has other applications as well. One example is collaborative path selection, where stakeholders with differing objectives must negotiate a universally acceptable path (e.g., municipal light rail routing through a University campus [13]). Without the complete Pareto-optimal set, the negotiation space would have been severely limited. In sum, computing the complete Pareto-optimal set in BOR provides a holistic range of

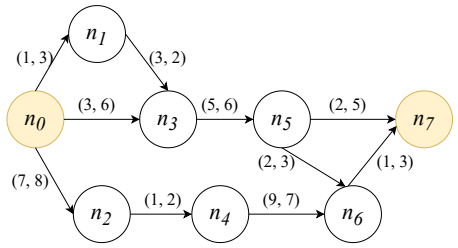


Figure 1: Example input bi-objective network. Paths $[n_0, n_3, n_5, n_7]$ and $[n_0, n_1, n_3, n_5, n_7]$ form the Pareto frontier between n_0 and n_7.

alternatives that can be intelligently filtered to identify the most desirable paths given a specific user profile and preference system [18, 30]. This exhaustive approach ensures that solutions are robust, flexible, and adaptable to various real-world scenarios [28].

The BOR problem is challenging due to the non-comparability of candidate paths and the potentially large volume of Pareto-optimal paths that need to be enumerated, which leads to the high time complexity of the path search process. In fact, it is an NP-hard problem and the number of solutions may be exponential in the number of nodes in the worst case [20].

Current methods for bi-objective routing broadly include exact methods [11, 14, 19, 20, 26] that compute the complete Pareto-optimal set, as well as non-exact methods based on subset approximation [9, 21, 27] and constrained shortest path search [1, 16, 22, 23, 31, 32]. Of concern here are the exact methods for solving BOR problems. An early work by Raith et al. [20] studied how traditional path enumeration strategies like ranking, label-correcting, and label-setting (e.g., Dijkstra) could be applied to the BOR problem. These Dijkstra-like methods were later extended [26] to include a bi-directional search. A well-received method, NAMOA*dr [19], shortens the search process using precise label setting to reduce the count of dominance checks. Hernández et al.'s bi-objective A* (BOA*) algorithm [11] further refines the dominance check operation. However, since all these methods are pure compute-on-demand methods, their time complexity remains unreasonably high due to the enormous number of Pareto-optimal subpaths encountered during the online path search process.

One way forward is to rethink techniques used to facilitate single-objective routing and adapt them for today's online on-demand BOR environment. A common approach in single-objective routing involves precomputing all-pair solutions and storing them online, simplifying the routing process to merely retrieving requested origin-destination pairs from the precomputed results [24]. Additionally, prior work on single-objective hierarchical routing [2, 3, 10, 12, 15, 29] has explored graph partitioning and storing only the shortest paths between specific nodes within subgraphs to reduce precomputation time and storage requirements. These paths are then concatenated across fragments over a summary graph to generate the shortest paths throughout the entire graph.

However, generalizing traditional single-objective hierarchical algorithms to bi-objective cases is non-trivial due to multiple factors. First, traditional hierarchical algorithms rely on the assumption that candidate paths can be sorted based on a single objective. This assumption does not hold in bi-objective scenarios where paths may be non-comparable (i.e., non-dominated) with each other. Second, whereas traditional hierarchical algorithms focus on maintaining only the shortest path for each node pair, bi-objective scenarios require the precomputation and retrieval of multiple Pareto-optimal paths (i.e., the Pareto frontier) between node pairs. This complexity further motivates the development of innovative pruning techniques that efficiently reduce the search space for hierarchical algorithms.

Thus, we propose a Multi-level Bi-Objective Routing (MBOR) approach that incorporates three innovative concepts: boundary multigraph representation, Pareto frontier encoding, and two-dimensional cost-interval based pruning. Our main contributions are as follows:

- We introduce the concept of a boundary multigraph to facilitate bi-objective searching across fragments and describe a novel materialization representation named a Multi-level Encoded Pareto Frontier View (MEPFV).
- We propose a basic multi-level bi-objective routing algorithm (MBOR-Basic) to efficiently encode and retrieve Pareto frontiers from an MEPFV.
- We introduce an advanced version of the algorithm to reduce the search space, MBOR-Adv, which utilizes two novel pruning techniques, one based on a two-dimensional cost-interval and the second, a multi-edge pruning technique.
- We prove the correctness and completeness of our proposed methods and validate them through extensive experiments on real road network datasets. Results show methods with precomputation have significantly faster online runtime than the compute-on-demand methods. Further, MBOR-Adv can significantly reduce the online runtime and precomputation time compared with the baseline.

Compared with existing data structures for BOR-like problems [4, 7, 8, 16, 33], our paper introduces novel data structures: the boundary multigraph and the Multi-level Encoded Pareto Frontier View, which facilitate bi-objective searching across fragments. In contrast, [4, 7, 8, 33] utilize contraction hierarchies, a data structure fundamentally different from the boundary multigraph proposed in our paper. Working on a different problem that identifies one constrained shortest path in the multi-objective scenario, [16] employs a traditional boundary graph data structure [29], whereas our work advances this concept by proposing a boundary multigraph structure. By definition, the traditional boundary graph data structure [16, 29] does not permit multiple edges between a given node pair, whereas the boundary multigraph does. Our multigraph representation offers several advantages over the traditional boundary graph. For instance, it facilitates min-cut partitioning in multi-level hierarchical representations by naturally accounting for the number of edges between each node pair. Furthermore, the known properties of multigraphs can be leveraged to reason about the properties of boundary multigraph data structures and their associated algorithms. Moreover, our proposed multigraph representation provides a natural way to model real-world transportation systems that involve various modes, such as walking, cycling, buses, driving, and trains.

Scope. This paper focuses on spatial networks where nodes have specific locations on the surface of the Earth, and edges geographically connect nodes (e.g., road networks). Such networks typically feature low density and large diameter. These properties

Table 1: Table of notations

Symbol	Explanation
G	Graph
N, E	Node set, edge set
$\mathbf{c}$	A bi-objective cost function
c^1, c^2	The first/second cost component of $\mathbf{c}$
$P_G(o,d)$	The set of all origin-destination paths in G
$PC_G(o,d)$	The set of costs for all origin-destination paths in G
$PO_G(o,d)$	The set of all Pareto-optimal paths in G
$POC_G(o,d)$	The set of costs for all Pareto-optimal paths in G

might not hold in other types of graphs, such as social networks. For simplicity, this paper starts with a two-level hierarchy of the network for the examples and deployment. The proposed method can be generalized to a multi-level hierarchy as shown in Sec. **??**. The updating issue is not considered. We use min-cut partitioning to decompose the original graph into fragments [25].

2 PROBLEM FORMULATION

2.1 Notations and Definitions

Definition 2.1. A **spatial network** refers to a directed graph $G = (N, E, \mathbf{c})$, where N is a set of nodes (e.g., n_0 in Figure 1), and E is a set of edges connecting nodes (e.g., $[n_0, n_1]$ in Figure 1). $\mathbf{c} : E \rightarrow \mathbb{R}^+ \times \mathbb{R}^+$ is a **cost function** associating a pair of non-negative costs with each edge (e.g., $\mathbf{c}([n_0, n_1]) = (1, 3)$). We denote the first and second components of $\mathbf{c}$ by c^1 and c^2, respectively (e.g., $c^1([n_0, n_1]) = 1$, $c^2([n_0, n_1]) = 3$).

Definition 2.2. A **path** p is a sequence of edges (e.g., $[n_5, n_6, n_7]$ in Figure 1). A **path cost** is the sum of the costs on all the edges in a path (e.g., $\mathbf{c}([n_5, n_6, n_7]) = \mathbf{c}([n_5, n_6]) + \mathbf{c}([n_6, n_7]) = (2+1, 3+3) = (3, 6)$ in Figure 1). For a given origin node o and a destination node d, the **set of all origin-destination paths** is denoted by $P_G(o, d)$ (e.g., $P_G(n_5, n_7) = \{[n_5, n_7], [n_5, n_6, n_7]\}$ in Figure 1). The **set of costs for all origin-destination paths** is denoted by $PC_G(o, d)$ (e.g., $PC_G(n_5, n_7) = \{(2, 5), (3, 6)\}$ in Figure 1). The "$\oplus$" symbol denotes the Minkowski sum [17] between two sets: $A \oplus B = \{a + b | a \in A, b \in B\}$. For example, with $A = \{(1, 0), (0, 1)\}$ and $B = \{(0, 0), (1, 1)\}$, $A \oplus B = \{(1, 0), (2, 1), (0, 1), (1, 2)\}$

Definition 2.3. Let $p, p' \in P_G(o, d)$ be two paths leading from node o to node d. Then, $\mathbf{c}(p) \prec \mathbf{c}(p')$ denotes that $c^1(p) \leq c^1(p')$, $c^2(p) \leq c^2(p')$, and $\mathbf{c}(p) \neq \mathbf{c}(p')$. The symbol "$\prec$" denotes the dominance relationship between paths: $p \prec p'$ if $\mathbf{c}(p) \prec \mathbf{c}(p')$. Path p is said to **dominate** p' iff $\mathbf{c}(p) \prec \mathbf{c}(p')$. For example, in Figure 1, path $[n_5, n_7]$ with cost (2, 5) dominates $[n_5, n_6, n_7]$, whose cost is (3, 6).

Definition 2.4. Given an origin node o and a destination node d, the **Pareto-optimal set (also known as the Pareto frontier, the non-dominated set)**, denoted by $PO_G(o, d) \subseteq P_G(o, d)$, contains all the origin-destination paths that are not dominated by another path. The set of costs for all non-dominated paths is defined as a **Pareto-optimal cost set**, denoted by $POC_G(o, d) \subseteq PC_G(o, d)$. For example, in Figure 1, $PO_G(n_5, n_7) = \{[n_5, n_7]\}$ since path $[n_5, n_6, n_7]$ is dominated by $[n_5, n_7]$, and $POC_G(n_5, n_7) = \{(2, 5)\}$.

2.2 Problem Definition

We formally define the bi-objective routing problem as follows:

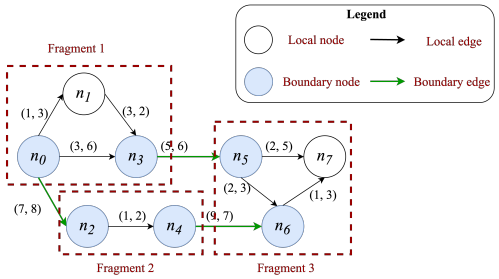


Figure 2: Graph partitioning of the spatial network in Figure 1 with 3 fragments.

- **Input:** A spatial network $G = (N, E, \mathbf{c})$ in which each edge is associated with a pair of non-negative costs $\mathbf{c}$, an origin node o, and a destination node d.
- **Output:** A set of paths $PO_G(o, d)$ between the origin and destination.
- **Objective:** $PO_G(o, d)$ is the complete Pareto-optimal set for the pair of costs $\mathbf{c}$.
- **Constraints:** The costs of the edges are non-negative. We did not consider traffic flow variability.

Example. Consider the example network shown in Figure 1. An instance of the bi-objective routing problem on this network with origin n_0 and destination n_7 (i.e., $PO(n_0, n_7)$) contains two paths: $[n_0, n_3, n_5, n_7]$ and $[n_0, n_1, n_3, n_5, n_7]$, whose cost (i.e., elements in $POC(n_0, n_7)$) is (10, 17) and (11, 16), respectively. Path $[n_0, n_2, n_4, n_6, n_7]$ whose cost is (18, 20), for example, is not Pareto-optimal since it is dominated by $[n_0, n_3, n_5, n_7]$.

3 PROPOSED APPROACH

First, we introduce the concept of a boundary multigraph and define the Multi-level Encoded Pareto Frontier View (MEPFV). We then present the basic Multi-level Bi-Objective Routing algorithm, which is designed to efficiently encode and retrieve Pareto frontiers from the MEPFV, followed by an advanced version of the algorithm with two novel pruning techniques.

3.1 Multi-level Encoded Pareto Frontier View

In the Multi-level Encoded Pareto Frontier View (MEPFV) model, we partition the graph into non-overlapping partitions and encode the Pareto-optimal path view within each fragment graph. To facilitate bi-objective searching across these partitions, we introduce the concept of a boundary multi-graph that encodes the Pareto frontiers between boundary nodes. The MEPFV is then composed of a set of Fragment Pareto-optimal Path Views (FPPV) and a Boundary Pareto-optimal Path View (BPPV), as defined below.

Definition 3.1. **Fragment graph.** Given a graph $G = (N, E, \mathbf{c})$, a **partition** of the graph is a set of subgraphs $S = \{S_1, S_2, \ldots, S_f\}$ where $S_i = (N_i, E_i, \mathbf{c}_i)$ includes node set N_i where $N_i \cap N_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{f} N_i = N$, and E_i is a set of local edges (e.g., black edges in Figure 2) connecting two nodes in N_i. Each $S_i = (N_i, E_i, \mathbf{c}_i)$ is a **fragment graph** (e.g., 3 fragment graphs in Figure 2). **Boundary nodes** N^b are defined as the set of nodes that have a neighbor in more than one fragment (e.g., n_3 in Figure 2), while the remaining nodes are **local nodes** (e.g., n_1 in Figure 2). Edges connecting two

nodes from different fragment graphs are called **boundary edges**, while other edges are **local edges**.

Definition 3.2. Given a set of fragment graphs, the encoded **fragment Pareto-optimal path view** (FPPV) stores the Pareto-optimal sets between all *node-to-boundary* and *boundary-to-node* pairs within each fragment. The encoded path view for each fragment is defined as a set of tables of 4-tuples [destination, the next hop, path cost 1, path cost 2] where each table is associated with an origin node. Here, "next hop" refers to the direct successor node within the Pareto-optimal path. For example, in Figure 3 (a), the fragment encoded path view of node n_0 in fragment S_1 denoted by the tuple $[n_3, n_1, 4, 5]$, tells us that there is a Pareto-optimal path from n_0 to n_3 in S_1, whose cost is (4, 5) and whose next-hop is n_1.

Definition 3.3. **Boundary Multigraph.** Given a graph G partitioned into fragments, along with each fragment graph's Fragment Pareto-optimal Path View (FPPV), a boundary multigraph is defined as $G^b = (N^b, E^b, C^b)$. Here, N^b represents the set of boundary nodes, E^b consists of multi-edges[1] connecting nodes from N^b, and $C^b : E^b \rightarrow \mathcal{P}(\mathbb{R}^+ \times \mathbb{R}^+)$, where $\mathcal{P}$ denotes the power set, is a cost function that associates each multi-edge with a set of pairs of non-negative costs. The construction of E^b and the corresponding cost function depends on the relationship between its endpoints:

(1) If both endpoints are within the same fragment, for each encoded fragment Pareto-optimal path between them, a local Pareto edge is defined (refer to the brown dashed edges in Figure 3 (b)). The presence of several Pareto-optimal paths between a pair of boundary nodes classifies G^b as a multigraph. In the implementation of G^b, we use a **multi-edge** connecting these nodes, associated with the set of Pareto-optimal costs, to represent the set of local Pareto edges between the same node pairs. For example, two local Pareto edges between n_0 and n_3 in Figure 3 can be represented as a multi-edge with cost set $\{(3, 6), (4, 5)\}$.
(2) If two endpoint nodes belong to different fragments, there is a corresponding multi-edge for each boundary edge in the original graph G (refer to the green edges in Figure 3 (b)). Each such multi-edge is associated with a single cost pair, identical to the cost of the original edge in G.

Definition 3.4. Given a boundary multigraph, the encoded **Boundary Pareto-optimal Path View** (BPPV) stores the Pareto frontiers between all boundary-to-boundary node pairs within the boundary multigraph. As shown in Figure 3 (b), the encoded boundary path view is defined as a collection of tables, each consisting of 5-tuples: [destination, the fragment ID of the next hop, the next hop, path cost 1, path cost 2], where each table is associated with an origin boundary node. In this context, "the next hop" in the boundary multigraph refers to the successor boundary node along with the corresponding cost to differentiate edges in a multigraph.

3.2 Basic Multi-level Bi-objective Routing

Our basic method for multi-level bi-objective routing has three parts, an algorithm to precompute MEPFV, a helper function to encode Pareto-optimal path views during precomputation, and last, an algorithm to retrieve Pareto frontiers for a given origin-destination query using MEPFV in online routing.

[1]In a multigraph, multiple edges connecting a pair of nodes can be represented by separate edges with identifiers or by a multi-edge with a set of costs. Fig. 3 uses the first notation for visualization, but the rest of this paper uses the second for implementation.

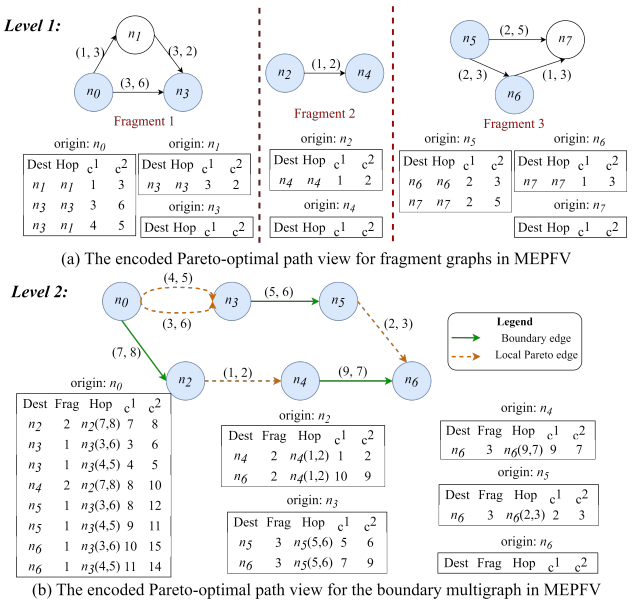


Figure 3: An example MEPFV with boundary multigraph encoding based on the graph partitioning shown in Figure 2.

3.2.1 MBOR-Basic: Precomputation. The precomputation algorithm (Algorithm 1), begins by identifying boundary edges that connect nodes from different fragments, designating the endpoints of these edges as boundary nodes (lines 1 - 4). It then encodes the fragment Pareto-optimal path view (FPPV) for each fragment by invoking the *paretoOptEncoding* function (see Algorithm 2). Next, the boundary multigraph is constructed (lines 6 - 12) following the procedure outlined in Definition 3.3. In this process, the *Succ* function retrieves the set of successor nodes connected to a node, and the *insertMultiEdge* function creates a multi-edge if one does not exist and then appends a cost to the multi-edge's cost set. The costs are retrieved from $FPPV_{Frag(n)}(n, n').costs$, which holds the local Pareto frontier costs between nodes n and n' within their fragment stored in the FPPV. Finally, the boundary Pareto-optimal path view (BPPV) is encoded by applying *paretoOptEncoding* (Algorithm 2) to the boundary multigraph.

The *paretoOptEncoding* function (Algorithm 2), encodes the Pareto frontiers for all boundary-related node pairs (node-to-boundary and boundary-to-node) within a given multigraph. It enumerates all nodes as potential starting points and utilizes a multigraph bi-objective Dijkstra search to compute one-to-all Pareto frontiers from each node. Each path explored from a starting node is represented by a label l, where $node(l)$ indicates the current node, $\mathbf{c}(l)$ specifies the path costs, and $parent(l)$ points to the previous label in the path (lines 1-3). Then, a Dijkstra-like search proceeds using a priority queue that organizes labels by their cost vectors in lexicographic order (line 5). The design of the priority queue ensures that the Pareto frontier search progresses such that the first cost component increases while the second decreases, thereby requiring only the maintenance of the current minimum of the second cost

Algorithm 1: MBOR-Basic: Precomputing MEPFV

Input: A graph $G = (N, E, \mathbf{c})$, a partition of G: $S = \{S_1, S_2, \ldots, S_f\}$
Output: The MEPFV for G including FPPV and BPPV

1 $N^b := \{\}$
2 **for** *each* $e \in E$ **do**
3 **if** $Frag(e.endpointU) \neq Frag(e.endpointV)$ **then**
4 $N^b = N^b \cup \{e.endpointU, e.endpointV\}$
5 $FPPV_{S_i} = paretoOptEncoding(S_i), \forall S_i \in S$
6 **for** *each* $(n_i^b, n_j^b \in N^b) \wedge n_i^b \neq n_j^b$ **do**
7 **if** $Frag(n_i^b) \neq Frag(n_j^b)$ **then**
8 **if** $n_j^b \in Succ(n_i^b)$ **then**
9 $G^b.insertMultiEdge(n_i^b, n_j^b, \mathbf{c}([n_i^b, n_j^b]))$
10 **else**
11 **for** *each cost* $\in FPPV_{Frag(n_i^b)}(n_i^b, n_j^b).costs$ **do**
12 $G^b.insertMultiEdge(n_i^b, n_j^b, cost)$
13 $BPPV = paretoOptEncoding(G^b)$
14 **return** *BPPV and* $FPPV_{S_i}, \forall S_i \in S$

Algorithm 2: MBOR-Basic: ParetoOptEncoding for G_0

Input: A multigraph $G_0 = (N_0, E_0, C_0)$
Output: The Pareto-optimal path view (PPV) of G_0

1 **Function** `initializeNodeStates`(N_0, n_{start})**:**
2 $frontiers(n) := \emptyset, \forall n \in N_0$, $c^2_{min}(n) := \infty, \forall n \in N_0$
3 **return** new label l_0 with $node(l_0) = n_{start}$, $\mathbf{c}(l_0) = (0, 0)$, $parent(l_0) := null$
4 **Function** `initializePriorityQueue`(l_0)**:**
5 Initialize a priority queue: *Open*, where labels are prioritized by their **c**-vectors in lexicographic order and insert l_0 to *Open*
6 **return** *Open*
7 **for** *each* $n_{start} \in N_0$ ***in parallel*** **do**
8 l_0 := `initializeNodeStates`(N_0, n_{start})
9 *Open* := `initializePriorityQueue`(l_0)
10 **while** *Open* $\neq \emptyset$ **do**
11 $l := Open.pop()$
12 **if** $c^2(l) \geq c^2_{min}(node(l))$ **then**
13 **continue**
14 $c^2_{min}(node(l)) = c^2(l)$, Add l to $frontiers(node(l))$
15 **for** *each* $n' \in Succ(node(l))$ **do**
16 **for** *each* **edgeCost** $\in C_0[node(l), n']$ **do**
17 l' := a new label with $node(l') = n'$
18 $\mathbf{c}(l') = \mathbf{c}(l)$ + **edgeCost**, $parent(l') := l$
19 **if** $c^2(l') \geq c^2_{min}(node(l'))$ **then**
20 **continue**
21 Add l' to *Open*
22 $PPV(n_{start})$:= Create PPV from $frontiers$
23 **return** *PPV*

for each node to determine if a new label is dominated (lines 12 and 19). Whenever an edge cost is encountered in the multigraph, then a corresponding new label is generated and inserted into the priority queue if not dominated (see lines 15-21). The search continues until all potential paths are evaluated. After each search from a starting node, line 22 encodes the computed Pareto frontiers between boundary-related node pairs into a Pareto-optimal path view to reduce storage costs.

3.2.2 MBOR-Basic: Online Pareto Frontier Retrieval. With the MEPFV precomputed, Algorithm 3 proceeds to retrieve the Pareto frontier for a given *o-d* pair, including the following queries sent to MEPFV:

- *Boundary*(S_i): Returns the set of boundary nodes for S_i.
- *Graph.ParetoOpt*(o', d'): Returns the Pareto-optimal path set between o' and d' within *Graph* by iteratively looking up MEPFV using (next-hop, c^1, c^2) as the keys.
- *ExpandBoundaryPathSet*(*boundaryPathSet*): Expands each boundary path in the input set and return the set that contains all corresponding paths in the original G by using the *ExpandBoundaryEdge*(*boundaryMultiedge*) for each boundary multi-edge of each boundary path.
- *ExpandBoundaryEdge*(*boundaryMultiedge*): If two endpoints of the boundary multi-edge are within the same fragment, expand it by iteratively looking up the MEPFV for this fragment using (next-hop, c^1, c^2) as the keys. Otherwise, returns the input edge.
- *DominanceCheck*(pathSet): Returns the Pareto-optimal subset from the given set.

The algorithm initially identifies a superset of the Pareto-optimal solutions, PO^{sup}, by enumerating all possible boundary node pairs between the origin fragment and destination fragment and combining the precomputed Pareto frontiers of origin-to-boundary, boundary-to-boundary, and boundary-to-destination paths (lines 2-4). Note that since every combination of the Pareto-optimal subpaths needs to be considered as a candidate Pareto-optimal path, we use a Minkawskin sum ("⊕") in line 4 to generate the full combination of the precomputed Pareto frontiers. If the origin and destination are within the same fragment, the local Pareto-optimal solution set is also added to the superset (lines 5-6). Finally, the Pareto frontier is generated with the non-dominated subset of PO^{sup}.

Example execution trace: Consider the example MEPFV shown in Figure 3, and a BOR query from n_0 to n_7. As shown in Table 2, MBOR-Basic generates the cost of Pareto superset (PO^{sup}) by retrieving the encoded Pareto-optimal path costs (*POC*) from the origin to the origin boundary node, from the origin boundary node to the destination boundary node, and from the destination boundary node to the destination node for all possible pairs of origin boundary nodes (i.e., n_0, n_3) and destination boundary nodes (i.e., n_5, n_6). Then *DominanceCheck*($\{(12, 17), (11, 18), (11, 16), (10, 17)\}$) = $\{(11, 16), (10, 17)\}$, and the corresponding Pareto-optimal paths can be expanded by querying the next-hop in MEPFV.

3.3 Advanced Multi-level Bi-objective Routing

We developed an advanced version of the MBOR algorithm, MBOR-Adv, that reduces the search space through novel pruning techniques, namely two-dimensional cost-interval based pruning and multi-edge pruning.

3.3.1 Two-dimensional Cost-interval based Pruning in Online Routing. During the online Pareto frontier retrieval phase in MBOR, the Pareto-optimal superset (PO^{sup} in Algorithm 3) is constructed by

Algorithm 3: MBOR-Basic: Pareto Frontier Retrieval

Input: MEPFV of a graph G, o: origin, d: destination
Output: Pareto-optimal path set $PO_G(o, d)$

1 $PO^{sup} := \{\}$
2 **for** *each* $oBN' \in Boundary(Frag(o))$ **do**
3 **for** *each* $dBN' \in Boundary(Fragment(d))$ **do**
4 $PO^{sup} = PO^{sup} \cup$
 $DominanceCheck(Frag(o).ParetoOpt(o, oBN') \oplus$
 $ExpandBoundaryPathSet(G^b.ParetoOpt(oBN', dBN')) \oplus$
 $Frag(d).ParetoOpt(dBN', d))$
5 **if** *Fragment(o) == Fragment(d)* **then**
6 $PO^{sup} = PO^{sup} \cup Frag(o).ParetoOpt(o, d)$
7 $PO := DominanceCheck(PO^{sup})$
8 **return** PO

Table 2: Pareto frontier retrieval using MEPFV in Fig. 3.

$POC(o, oBN)$	$POC(oBN, dBN)$	$POC(dBN, d)$	Cost of PO^{sup}
$(n_0, n_0) = \{(0, 0)\}$	$(n_0, n_5) = \{(8, 12), (9, 11)\}$	$(n_5, n_7) = \{(2, 5)\}$	$\{(10, 17), (11, 16)\}$
$(n_0, n_0) = \{(0, 0)\}$	$(n_0, n_6) = \{(10, 15), (11, 14)\}$	$(n_6, n_7) = \{(1, 3)\}$	$\{(11, 18), (12, 17)\}$
$(n_0, n_3) = \{(3, 6), (4, 5)\}$	$(n_3, n_5) = \{(5, 6)\}$	$(n_5, n_7) = \{(2, 5)\}$	$\{(10, 17), (11, 16)\}$
$(n_0, n_3) = \{(3, 6), (4, 5)\}$	$(n_3, n_6) = \{(7, 9)\}$	$(n_6, n_7) = \{(1, 3)\}$	$\{(11, 18), (12, 17)\}$

retrieving and combining the encoded Pareto frontiers from the origin to origin boundary nodes, between pairs of boundary nodes, and from destination boundary nodes to the destination. This process involves all feasible combinations of boundary node pairs that facilitate transitions from the origin fragment to the destination fragment, ensuring the algorithm's completeness. However, this combination of Pareto frontiers can significantly increase the size of PO^{sup} due to the combinatorial nature of the Minkowski sum. For instance, the Minkowski sum of two sets with sizes m and n can produce a result set with a size of up to mn. This expansion makes it time-consuming to generate PO^{sup} and to perform dominance checks on it during the online routing phase. To address these challenges while preserving the correctness and completeness of the algorithm, we propose a novel pruning technique based on two-dimensional cost-intervals. This method efficiently prunes the search space of candidate boundary nodes without generating the entire combined Pareto frontiers. A two-dimensional cost-interval for the Pareto frontier between any node pair is defined as follows:

Definition 3.5. For a given node pair (n, n'), let $p_1^*, p_2^* \in P_G(n, n')$ be the shortest paths minimizing the first and second cost components, respectively. We define the **two-dimensional cost-interval (2DCI)** of the Pareto frontier between n and n' as $([c^1(p_1^*), c^1(p_2^*)], [c^2(p_2^*), c^2(p_1^*)])$.

The 2DCI, as defined in Definition 3.5, effectively represents a minimum bounding rectangle of the corresponding Pareto frontier in the objective space, based on the shortest paths that minimize each cost component. Specifically, for any path p within the Pareto-optimal set $PO_G(n, n')$, the following conditions hold: $c^1(p_1^*) \leq c^1(p) \leq c^1(p_2^*)$ and $c^2(p_2^*) \leq c^2(p) \leq c^2(p_1^*)$. These conditions confirm that the 2DCI captures the full range of Pareto-optimal costs between the nodes n and n'. Furthermore, the optimal substructure property of shortest paths ensures that combining the 2D cost-intervals for sequential node pairs is straightforward, involving just the addition of corresponding cost components.

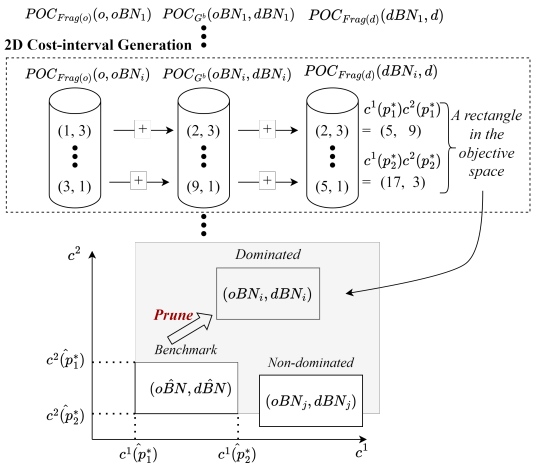


Figure 4: An example two-dimensional cost-interval based pruning in online Pareto frontier retrieval in MBOR-Adv.

Figure 4 shows an example of the generation of a 2DCI for the combined Pareto frontier corresponding to a boundary node pair (oBN_i, dBN_i) between an origin and a destination. Assume the costs in the precomputed Pareto-optimal path views (POC, bins in Fig. 4) are sorted in lexicographic order, then the summation of the first element in each bin produces $c^1(p_1^*), c^2(p_1^*)$ for the 2D cost-interval, while the summation of the last element yields $c^1(p_2^*), c^2(p_2^*)$. After generating the 2DCIs for all candidate boundary node pairs, the dominant relationship between 2DCIs is then defined as follows:

Definition 3.6. A 2D cost-interval $2DCI$ is said to **dominate** another $2DCI'$ if any of the following conditions holds:
1) $(2DCI.\min(c^1), 2DCI.\max(c^2)) \prec (2DCI'.\min(c^1), 2DCI'.\min(c^2))$
or 2) $(2DCI.\max(c^1), 2DCI.\min(c^2)) \prec (2DCI'.\min(c^1), 2DCI'.\min(c^2))$

As shown in Figure 4, the benchmark 2DCI dominates any 2DCIs whose lower-left corner $(\min(c^1), \min(c^2))$ is positioned within the shaded area (e.g., the 2DCI corresponding to (oBN_i, dBN_i)). The following lemma then facilitates the effective pruning of boundary node pairs, such as (oBN_i, dBN_i) in Fig. 4, during online routing:

Lemma 3.7. *If a 2D cost-interval is dominated, then all corresponding paths associated with it are dominated paths.*

Algorithm 4 outlines the pseudocode for online Pareto frontier retrieval in MBOR-Adv using 2DCI-based pruning. The algorithm starts by constructing the combined 2DCI for each potential boundary node pair from the MEPFV (see line 2), as shown in Figure 4. For brevity, we omit explicit mention of the origin and destination nodes in the combined representation. Thus, the 2DCI encompassing transitions from o to an origin boundary node (oBN), from oBN to a destination boundary node (dBN), and from dBN to d is collectively referred to as $2DCI(oBN, dBN)$. Given that the Pareto-optimal paths are explored in lexicographic order when precomputing the MEPFV, the costs in the precomputed POC are inherently sorted in lexicographic order, thus allowing for a constant time complexity for the generation of the combined 2DCI for each boundary node pair. Subsequently, we select the 2DCI with the minimum $(\min(c^1), \min(c^2))$ in lexicographic order (i.e., the most leftward rectangle in the objective space) as the benchmark 2DCI

(see lines 3-4). Then, a 2DCI-based pruning check is performed in lines 7-9 to directly prune boundary node pairs before performing the Minkowski sum to reduce the search space.

Algorithm 4: MBOR-Adv: Online Pareto Frontier Retrieval using 2D Cost-interval based Pruning

Input: MEPFV of a graph G, o: origin, d: destination
Output: Pareto-optimal path set $PO_G(o,d)$

1 $PO^{sup} := \{\}$
2 $2DCI(oBN_i, dBN_i) := 2DCIgeneration(MEPFV, oBN_i, dBN_i), \forall oBN_i \in Boundary(Frag(o)), dBN_i \in Boundary(Frag(d))$
3 Find the benchmark $\hat{2DCI}$ with the minimum $(c^1(\hat{p}_1^*), c^2(\hat{p}_2^*))$ in lexicographic order
4 Denote the bounds of $\hat{2DCI}$ as $c^1(\hat{p}_1^*), c^1(\hat{p}_2^*), c^2(\hat{p}_2^*), c^2(\hat{p}_1^*)$
5 **for** *each* $oBN \in Boundary(Frag(o))$ **do**
6 **for** *each* $dBN \in Boundary(Frag(d))$ **do**
7 $2DCI' := 2DCI(oBN, dBN)$
8 **if** $(c^1(\hat{p}_1^*), c^2(\hat{p}_1^*)) \prec (2DCI'.min(c^1), 2DCI'.min(c^2)) \vee (c^1(\hat{p}_2^*), c^2(\hat{p}_2^*)) \prec (2DCI'.min(c^1), 2DCI'.min(c^2))$ **then**
9 **continue**
10 $PO^{sup} = PO^{sup} \cup DominanceCheck(Frag(o).ParetoOpt(o, oBN) \oplus ExpandBoundaryPathSet(G^b.ParetoOpt(oBN', dBN') \oplus Frag(d).ParetoOpt(dBN, d))$
11 **if** *Fragment(o) == Fragment(d)* **then**
12 $PO^{sup} = PO^{sup} \cup Frag(o).ParetoOpt(o, d)$
13 $PO := DominancCheck(PO^{sup})$
14 **return** PO

Example execution trace: Consider the MEPFV shown in Figure 3, and a BOR query from n_0 to n_7. The candidate boundary node pairs include (n_0, n_5), (n_0, n_6), (n_3, n_5), and (n_3, n_6). MBOR-Adv initially computes the combined 2DCIs corresponding to these boundary node pairs: $2DCI(n_0, n_5) = ([10, 11], [16, 17])$, $2DCI(n_0, n_6) = ([11, 12], [17, 18])$, $2DCI(n_3, n_5) = ([10, 11], [16, 17])$, and $2DCI(n_3, n_6) = ([11, 12], [17, 18])$. The algorithm selects $2DCI(n_0, n_5)$ as the benchmark and prunes the boundary node pairs (n_0, n_6) and (n_3, n_6) due to their dominance ((10,17)$\prec$(11,17)). Then, the Pareto-optimal superset (PO^{sup}) construction is documented in Table 3. It is worth noting that, in this example, each boundary node pair generates two combined Pareto-optimal paths, yielding a complexity for generating $2DCI$ equivalent to directly generating PO^{sup}. However, in practical scenarios, generating $2DCI$ is executed in constant time with exactly two operations (heads combination and tails combination), while the complexity of generating PO^{sup} is influenced by the number of Pareto-optimal subpaths due to the combinatorial nature in the Minkowski sum operation. More details are discussed in Sec. 5.

Table 3: Advanced Pareto frontier retrieval using MEPFV in Fig. 3. Boundary node pairs (n_0, n_6), (n_3, n_6) are 2DCI-pruned.

$POC(o, oBN)$	$POC(oBN, dBN)$	$POC(dBN, d)$	Cost of PO^{sup}
$(n_0, n_0) = \{(0, 0)\}$	$(n_0, n_5) = \{(8, 12), (9, 11)\}$	$(n_5, n_7) = \{(2, 5)\}$	$\{(10, 17), (11, 16)\}$
$(n_0, n_3) = \{(3, 6), (4, 5)\}$	$(n_3, n_5) = \{(5, 6)\}$	$(n_5, n_7) = \{(2, 5)\}$	$\{(10, 17), (11, 16)\}$

3.3.2 Multi-edge Pruning for Bounday Multigraph Encoding in Precomputation. In the precomputation phase, the boundary multigraph can become dense due to multiple local Pareto edges between boundary nodes from the same fragment. This density significantly contributes to the encoding time of the boundary multigraph (line 13 in Algorithm 1). To address this, we introduce two multi-edge pruning techniques that reduce the search space for encoding the boundary multigraph, detailed in Algorithm 5. The first pruning criterion (lines 10-11), is supported by the following lemma:

Lemma 3.8. ***Bi-objective Triangle Inequality:*** *In a bi-objective scenario, combining the Pareto-optimal paths between nodes n and n', and nodes n' and n'', results in paths that are either dominated by or belong to the Pareto frontier from n to n''.*

The validity of this lemma is established by integrating the definitions of Pareto-optimality and the triangle inequality. When encoding the boundary multigraph, if a label has already reached node n from another local node n_0 within the same fragment, further searches from this label to another local node n' can be pruned (line 10). This is because any path generated from n_0 to n' would either be dominated by or already explored through direct labels from n_0 to n'.

The second pruning technique leverages the fact that costs within boundary multi-edges are stored in lexicographic order. When generating new labels, we enumerate current edge costs in reverse order (line 12 in Algorithm 5), prioritizing an increase in the second cost component. If the current edge cost already exceeds the minimum possible second cost (lines 15-16), we can prune subsequent edge costs in the multi-edge.

4 THEORETICAL EVALUATION

We analyzed the proposed algorithms theoretically for correctness and completeness. An algorithm for the BOR problem is considered to be correct and complete if its output path set is both a subset and a superset of the Pareto-optimal set.

Theorem 4.1. ***Correctness and completeness:*** *At the termination of MBOR-Basic and MBOR-Adv, the solution set produced is the complete Pareto frontier from the origin to the destination.*

Proof Sketch. In a given origin-destination pair, the origin and destination nodes may be located either in the same fragment or in different fragments. The correctness and completeness of MBOR are guaranteed by the following lemmas for these two cases, respectively. (1) The Pareto frontier from origin o to destination d from different fragments corresponds to the Pareto-optimal subset of all possible concatenated Pareto-optimal paths composed of three parts: from o to a local boundary node n_i, from n_i to another local boundary node n_j of d, and from n_j to d. (2) If both the origin and destination are within the same fragment, then the Pareto frontier is the Pareto-optimal subset of the union of the two, that is: the local Pareto frontier within the fragment, and all possible concatenated Pareto-optimal paths composed of three parts: from origin o to a local boundary node n_i, from n_i to another local boundary node n_j, and from n_j to d. □

Algorithm 5: MBOR-Adv: ParetoOptEncoding for Boundary Multigraph

Input: A boundary multigraph $G_0 = (N_0, E_0, C_0)$
Output: The boundary Pareto-optimal path view (BPPV) of G_0

```
for each n_start ∈ N_0 in parallel do
    l_0 := initializeNodeStates(N_0, n_start)
    Open := initializePriorityQueue(l_0)
    while Open ≠ ∅ do
        l := Open.pop()
        if c²(l) ≥ c²_min(node(l)) then
            continue
        c²_min(node(l)) = c²(l), Add l to frontiers(node(l))
        for each n' ∈ Succ(node(l)) do
            if isLocal(l) ∧ Frag(n') == Frag(node(l)) then
                continue
            // costs are saved in lexicographic order
            for each edgeCost ∈ reverse(C^b[node(l), n']) do
                l' := a new label with node(l') = n'
                c(l') = c(l) + edgeCost, parent(l') := l
                if c²(l') ≥ c²_min(node(l')) then
                    break
                isLocal(l') := Frag(n') == Frag(node(l))
                Add l' to Open
    BPPV(n_start) := Creat PPV from frontiers
return BPPV
```

5 EXPERIMENTAL EVALUATION

5.1 Experiment Design

We validated the proposed methods with two types of analysis:

1) ***Comparative Analysis:*** To demonstrate the benefits of precomputation, we compared the online runtimes of our proposed methods against two compute-on-demand methods that we consider state-of-the-art. Additionally, we compared the precomputation times and online runtimes between MBOR-Basic and MBOR-Adv to evaluate the effectiveness of our pruning techniques.

2) ***Sensitivity Analysis:*** We assessed the sensitivity of the proposed methods to changes in the number of fragments.

Dataset: The dataset used was the 9th DIMACS Implementation Challenge: Shortest Path [5], provided by the Center for Discrete Mathematics and Theoretical Computer Science. This dataset comprises various real-world road network scenarios and is commonly used as a benchmark dataset in related work (e.g., [11, 19]). We used the San Francisco Bay Area (BAY) road network. The dataset provided two cost components: travel distance and time. To evaluate the methods on different network sizes, we partitioned the entire Bay Area into 5 fragments and randomly selected one fragment to generate a 1/5 Bay Area road network. We generated 1/10, and 1/20 BAY Area networks similarly. Then, we randomly generated 50 queries on each network. Table 4 lists key statistics of networks and the average number of solutions for the queries (i.e., the average size of the Pareto frontiers). The distribution of these queries and their impact on performance is analyzed in Section 5.2. The implementation of our methods is online available [2].

Table 4: Statistics of the road networks. The number of boundary nodes was calculated with a 50-fragment partitioning.

Road Network	1/20 BAY	1/10 BAY	1/5 BAY	BAY
# nodes	15,366	32,205	64,684	321,270
# edges	41,180	76,230	156,682	800,172
# boundary nodes	876	696	873	1322
# boundary multi-graph edges	39,728	12,302	35,104	277,242
real size	804K	1.5M	3.2M	18M
# average solutions	15	13	47	119

5.2 Comparative Analysis

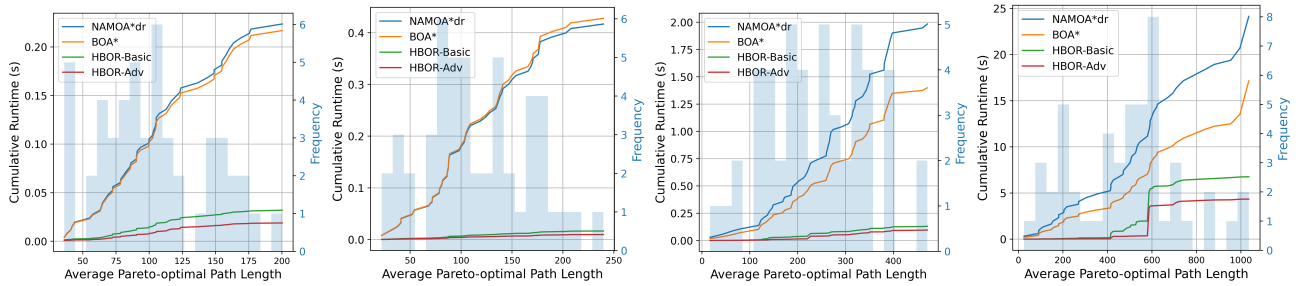
Baseline Methods: In order to make the evaluation results comparable, we only compared our proposed methods against other exact algorithms of the bi-objective problem, i.e., those that, given enough computational space and time, can find the set of all Pareto-optimal paths. Thus, the two state-of-the-art methods tested were: 1) NAMOA*dr [19] and 2) Bi-objective A* (BOA*) [11]. Both methods are best-first bi-objective search algorithms inspired by the A* search and have outperformed many other bi-objective routing methods [11] (e.g. bi-objective Dijkstra [14, 20] and bi-directional bi-objective Dijkstra [26]). We used the C implementations of these algorithms provided in [11], after fixing memory leak issues.

Table 5: Average and median online runtime (in milliseconds) on 50 instances of various BAY networks.

Method	1/20 BAY		1/10 BAY		1/5 BAY		Entire BAY	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med
NAMOA*dr	4.47	4.13	8.34	8.17	39.69	31.93	482.83	360.20
BOA*	4.34	4.02	8.56	8.31	27.99	20.43	343.05	221.69
Methods with Precomputation								
MBOR-Basic	0.64	0.53	0.33	0.32	2.61	1.20	135.31	29.54
MBOR-Adv	**0.38**	**0.30**	**0.19**	**0.18**	**1.93**	**0.70**	**86.84**	**4.56**

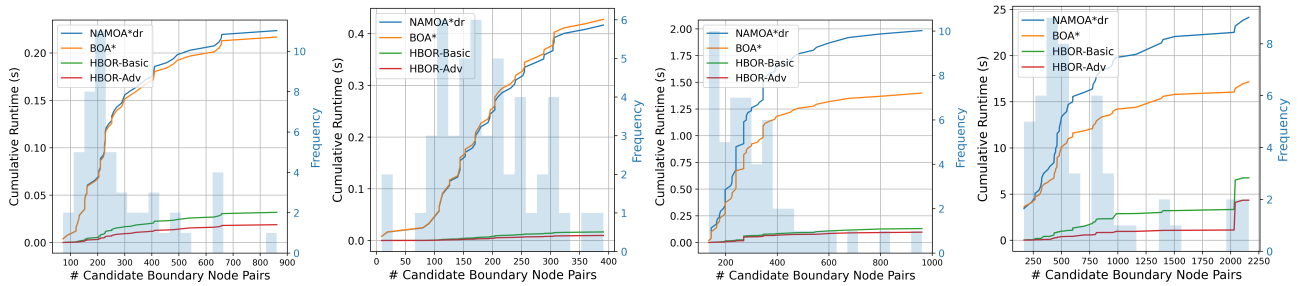
Table 5 presents the average and median runtimes (in milliseconds) for 50 queries on various BAY network sizes. MBOR-Basic and MBOR-Adv are under a 50-fragment partition. All four methods, being exact algorithms, produced the same solutions. Table 5 shows that the proposed methods with precomputation significantly outperform compute-on-demand methods in terms of online runtime, achieving more than a 10× improvement for most metrics. Further analysis sorted the queries by their average Pareto-optimal path length, with the cumulative runtime visualized on the left y-axis and the distribution of queries on the right y-axis in Figure 5. Similarly, Figure 6 plots the cumulative runtime against the number of candidate origin-destination boundary node pairs. One observation is that, unlike compute-on-demand methods—whose performance largely depends on the average Pareto-optimal path lengths—the performance of MBORs is heavily influenced by the number of candidate boundary node pairs enumerated during the online routing phase. Additionally, the network graph's architecture impacts the performance of MBORs. For instance, the 1/10 BAY network, despite having approximately twice the size compared to the 1/20 BAY network, exhibited significantly faster average online runtimes

[2] Our code: https://github.com/yang-mingzhou/MBOR



(a) 1/20 BAY (b) 1/10 BAY (c) 1/5 BAY (d) Entire BAY

Figure 5: Cumulative runtime in seconds (line graph, left y-axis) vs. Average Pareto-optimal Path Length (x-axis) with Histogram (right y-axis) on the 50 instances of various BAY networks.



(a) 1/20 BAY (b) 1/10 BAY (c) 1/5 BAY (d) Entire BAY

Figure 6: Cumulative runtime in seconds (line graph, left y-axis) vs. Number of candidate boundary node pairs (x-axis) with Histogram (right y-axis) on the 50 instances of various BAY networks.

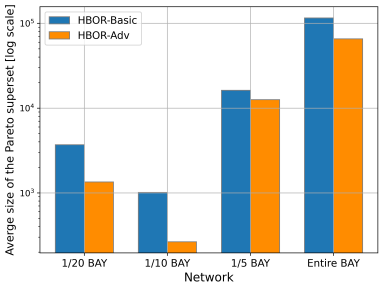


Figure 7: Average size of the Pareto-optimal superset (PO^{sup}) retrieved for online routing in MBOR-Basic and MBOR-Adv.

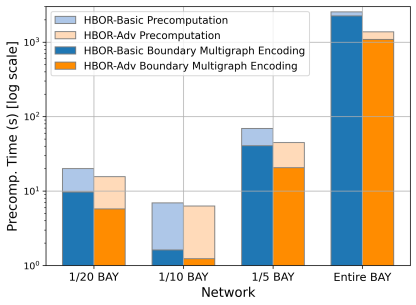


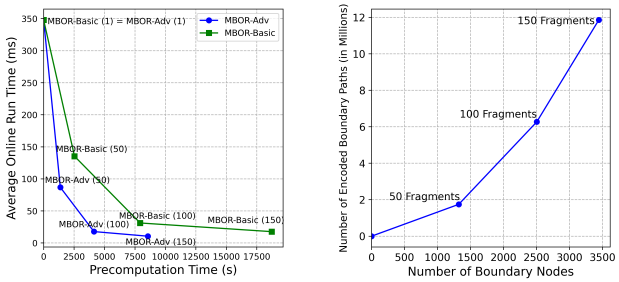
Figure 8: Precomputation time of MBOR-Basic and MBOR-Adv on various BAY networks.

in MBOR. This improvement is attributed to its more hierarchical structure with fewer boundary nodes (696 compared to 876).

Table 5 shows that HBOR-Adv's pruning technique significantly reduced the online runtime compared to MBOR-Basic, achieving a 35.8% reduction in the average time across the entire BAY network. This efficiency gain is corroborated by Figure 7, which shows a much smaller Pareto-optimal superset retrieved for online routing in MBOR-Adv. Finally, Figure 8 presents the precomputation times for MBOR-Basic and MBOR-Adv. It is evident that the encoding time for the boundary multigraph significantly contributes to the precomputation time, and the proposed multi-edge pruning techniques effectively reduce the precomputation time in MBOR-Adv.

5.3 Sensitivity Analysis

In the second phase of our experiments, we explored how the number of fragments affects the performance of the proposed methods. We varied the number of fragments on the Entire BAY network



(a) Precomp. time vs. online runtime (# fragments in parathesis). (b) # Boundary nodes vs. # Encoded boundary paths in MEPFV.

Figure 9: The impact of the number of fragments on the proposed methods over the Entire BAY network.

from 50 to 150 and computed the precomputation and average

online routing time. The results are shown in Figure 9a. This figure demonstrates that the proposed methods achieve an effective balance between all-pair precomputation and purely compute-on-demand approaches. Specifically, with only one fragment, all nodes in the network are contained within the same fragment without any boundary nodes, making MBOR equivalent to a compute-on-demand method, as indicated by the y-intercept in Figure 9a. Although precomputation time increases as the number of fragments increases, the average online runtime decreases due to the larger volume of information being precomputed and encoded. Also, the precomputation time for MBOR-Adv increases more slowly than for MBOR-Basic, thanks to its multi-edge pruning. If the number of fragments equals the number of nodes in the network, our method behaves similarly to an all-pair precomputation approach, where every node becomes a boundary node. This scenario is hypothetically represented at the x-intercepts. The increased precomputation time with a higher number of fragments is attributable to the significant rise in the number of encoded boundary Pareto-optimal paths in the MEPFV. When the number of fragments was increased from 50 to 150 in the entire BAY network, the boundary node count (Figure 9b) grew from 1,322 to 3,444, while the number of encoded boundary Pareto-optimal paths surged from 1,747,684 to 11,861,136. Considering that the network comprises 321,270 nodes, this substantial increase reinforces the impracticality of all-pair precomputation due to the large number of Pareto-optimal paths to be precomputed.

6 CONCLUSION AND FUTURE WORK

The bi-objective routing problem aims to find the full Pareto-optimal path sets for given origin-destination queries. In this work, we propose Multi-Level Bi-Objective Routing (MBOR) algorithms that incorporate three novel ideas: boundary multigraph representation, Pareto frontier encoding, and two-dimensional cost-interval based pruning. Our experiments on real-world road network data show that the proposed methods can significantly reduce the online runtime compared to state-of-the-art methods. In future work, we aim to extend our framework to multi-criteria routing scenarios. We also plan to investigate efficient strategies for updating the MEPFV in response to minor changes within the network.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants No. 1901099, the USDOE Office of Energy Efficiency and Renewable Energy under FOA No. DE-FOA-0002044, and Minnesota Supercomputing Institute (MSI). We also thank Kim Koffolt and the Spatial Computing Research Group for valuable comments and refinements.

REFERENCES

[1] Harry Kai-Ho Chan, Tiantian Liu, Huan Li, and Hua Lu. 2021. Time-constrained indoor keyword-aware routing. In *17th International Symposium on Spatial and Temporal Databases*. 74–84.

[2] Theodoros Chondrogiannis and Johann Gamper. 2016. ParDiSP: A partition-based framework for distance and shortest path queries on road networks. In *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, Vol. 1. IEEE, 242–251.

[3] Daniel Delling, Andrew V Goldberg, Thomas Pajor, and Renato F Werneck. 2011. Customizable route planning. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10*. Springer, 376–387.

[4] Daniel Delling and Dorothea Wagner. 2009. Pareto paths with SHARC. In *International Symposium on Experimental Algorithms*. Springer, 125–136.

[5] DIMACS. 2013. 9th DIMACS Implementation Challenge - Shortest Path. http://www.dis.uniroma1.it/challenge9/download.shtml. Accessed: May 2023.

[6] Matthias Ehrgott. 2005. *Multicriteria optimization*. Vol. 491. Springer Science & Business Media.

[7] Stefan Funke and Sabine Storandt. 2013. Polynomial-time construction of contraction hierarchies for multi-criteria objectives. In *2013 Proceedings of the Fifteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 41–54.

[8] Robert Geisberger, Moritz Kobitzsch, and Peter Sanders. 2010. Route planning with flexible objective functions. In *2010 Proceedings of the Twelfth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 124–137.

[9] Qixu Gong and Huiping Cao. 2022. Backbone Index to Support Skyline Path Queries over Multi-cost Road Networks. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*. OpenProceedings.org, 2:325–2:337. https://doi.org/10.48786/EDBT.2022.19

[10] Mohamed S Hassan, Walid G Aref, and Ahmed M Aly. 2016. Graph indexing for shortest-path finding over dynamic sub-graphs. In *Proceedings of the 2016 International Conference on Management of Data*. 1183–1197.

[11] Carlos Hernández et al. 2023. Simple and efficient bi-objective search algorithms via fast dominance checks. *Artificial Intelligence* 314 (2023), 103807.

[12] Ning Jing et al. 1998. Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *TKDE* 10, 3 (1998), 409–432.

[13] Mila Koumpilova. 2014. *Past battles aside, UMN sees bright future in Green Line.* https://www.twincities.com/2014/06/11/past-battles-aside-umn-sees-bright-future-in-green-line/

[14] Hans-Peter Kriegel, Matthias Renz, and Matthias Schubert. 2010. Route skyline queries: A multi-preference path planning approach. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, 261–272.

[15] Ken CK Lee, Wang-Chien Lee, Baihua Zheng, and Yuan Tian. 2010. ROAD: A new spatial object search framework for road networks. *IEEE transactions on knowledge and data engineering* 24, 3 (2010), 547–560.

[16] Ziyi Liu, Lei Li, Mengxuan Zhang, Wen Hua, and Xiaofang Zhou. 2023. Multi-constraint shortest path using forest hop labeling. *The VLDB Journal* 32, 3 (2023), 595–621.

[17] Tomas Lozano-Perez. 1990. *Spatial planning: A configuration space approach*. Springer.

[18] Matthias Müller-Hannemann and Karsten Weihe. 2006. On the cardinality of the Pareto set in bicriteria shortest path problems. *Annals of Operations Research* 147, 1 (2006), 269–286.

[19] Francisco-Javier Pulido et al. 2015. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research* 64 (2015), 60–70.

[20] Andrea Raith and Matthias Ehrgott. 2009. A comparison of solution strategies for biobjective shortest path problems. *COR* 36, 4 (2009), 1299–1331.

[21] Nicolás Rivera, Jorge A Baier, and Carlos Hernández. 2022. Subset Approximation of Pareto Regions with Bi-objective A. In *AAAI*, Vol. 36. 10345–10352.

[22] Dimitris Sacharidis and Panagiotis Bouros. 2013. Routing directions: Keeping it fast and simple. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 164–173.

[23] Dimitris Sacharidis, Panagiotis Bouros, and Theodoros Chondrogiannis. 2017. Finding the most preferred path. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–10.

[24] Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. 2008. Scalable network distance browsing in spatial databases. In *SIGMOD*. 43–54.

[25] Peter Sanders et al. 2013. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *SEA 2013, Rome, Italy,*, Vol. 7933. Springer, 164–175.

[26] Antonio Sedeño-Noda and Marcos Colebrook. 2019. A biobjective Dijkstra algorithm. *EJOR* 276, 1 (2019), 106–118.

[27] Michael Shekelyan, Gregor Josse, and Matthias Schubert. 2015. Linear path skylines in multicriteria networks. In *2015 IEEE 31st International Conference on Data Engineering (ICDE 2015)*. IEEE, 459–470.

[28] Michael Shekelyan, Gregor Jossé, and Matthias Schubert. 2015. Paretoprep: Efficient lower bounds for path skylines and fast path computation. In *SSTD*. Springer, 40–58.

[29] Shashi Shekhar et al. 1997. Materialization trade-offs in hierarchical shortest path algorithms. In *SSD'97 Berlin, Germany*. Springer, 94–111.

[30] Yuan Tian, Ken CK Lee, and Wang-Chien Lee. 2009. Finding skyline paths in road networks. In *SIGSPATIAL*. 444–447.

[31] Libin Wang and Raymond Chi-Wing Wong. 2023. QHL: A Fast Algorithm for Exact Constrained Shortest Path Search on Road Networks. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–25.

[32] Ye Yuan, Xiang Lian, Guoren Wang, Yuliang Ma, and Yishu Wang. 2019. Constrained shortest path query in a large time-dependent graph. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1058–1070.

[33] Han Zhang et al. 2023. Efficient multi-query bi-objective search via contraction hierarchies. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 33. 452–461.