Project Phase 2 Deliverable 2: Proof of Concept Implementation

Group 1: Haeri Kyoung, Oishani Ganguly, Akash Shrestha, Nasser Hasan Padilla, Kannekanti Nikhil Kanneganti, Venkata Tadigotla

University of the Cumberlands

MSCS-532-M20 – Algorithms and Data Structures

Professor Brandon Bass

July 26, 2025

Proof of Concept: Pareto-Optimal Pathfinding with Multi-Criteria Tradeoffs

Partial Implementation Overview

In this second phase, the focus was on building out the foundational logic behind a multi-criteria pathfinding system using Python. The core idea was to support decisions based on more than one factor, such as travel time, toll cost, and scenic quality. A graph-based model was used where each edge carried a 3-dimensional cost tuple.

Four modules were implemented:

- The Graph class provides a dynamic adjacency list that supports adding and removing nodes and edges. All edge weights are stored as cost tuples, and the graph supports BFS and DFS traversals.
- The Label class tracks cost vectors and maintains a reference to the prior label, allowing path reconstruction later.
- The LabelSet class holds the non-dominated labels at each node. It keeps the Pareto frontier minimal by removing dominated entries on insert.
- The LabelPriorityQueue wraps heapq and ensures that labels are expanded in lexicographic order.

All modules were designed to be modular, cleanly separated, and reusable for future integration.

Demonstration and Testing

To verify the correctness of each module, unit tests and a demo script were created.

- test_graph.py confirms that the Graph can add/remove nodes and edges, retrieve neighbors, and handle invalid inputs gracefully.
- test_label.py tests Label initialization and the correctness of dominance comparisons.

- test_label_set.py checks that dominated labels are correctly rejected and valid labels are added.
- test_priority_queue.py validates queue ordering and error handling.

The script demo.py ties everything together by creating a simple graph, performing traversals, testing label pruning and queue pops, and running a full expansion loop.

Here are a few outputs from demo_results.txt:

- Graph traversal from node A returns BFS: ['A', 'B', 'C', 'D'] and DFS: ['A', 'B', 'D', 'C'].
- LabelSet rejects labels that are strictly worse and keeps only the non-dominated ones.
- Priority queue pops labels in correct lexicographic order.
- Combined expansion loop shows label propagation from source to destination, maintaining the Pareto frontier.

All test files passed successfully and were run using run_tests.sh.

Implementation Challenges and Solutions

- Strict input validation was required on edge costs to avoid bugs caused by malformed tuples. The graph and label constructors now check that cost vectors are exactly 3 floats.
- Dominance logic had to be precise. A single-pass pruning strategy in LabelSet was implemented to remove dominated labels while adding new ones.
- Priority queue comparison presented a challenge since tuples do not have a natural priority in multiple dimensions. Python's default tuple comparison was used for lexicographic ordering.
- Traversal and queue errors were improved by adding explicit exceptions with helpful error messages. Now invalid node lookups and empty pops are clearly logged.

Next Steps

- Build the full unified expansion loop in a reusable function or class that can be tested independently.
- Add support for user-defined dominance policies or weighted priorities.
- Handle path reconstruction and return multiple full routes, not just label costs.
- Incorporate real-world datasets, such as those from OpenStreetMap.
- Build a UI to allow users to enter start and end nodes and view multiple routes.

Code Snippets and Documentation

Example: LabelSet add() method (from label_set.py)

```
def add(self, label: Label) -> bool:
    new = []
    for existing in self.labels:
        if existing.dominates(label):
            return False
        if not label.dominates(existing):
            new.append(existing)
    new.append(label)
    self.labels = new
    return True
```

Example: Label expansion loop (from demo.py)

```
while pq:
    lbl = pq.pop()
```

```
    for nbr, cost in g.neighbors(lbl.node):
        new_cost = tuple(x + y for x, y in zip(lbl.cost, cost))
        new_lbl = Label(new_cost, nbr, lbl)
        if label_sets[nbr].add(new_lbl):
            pq.push(new_lbl)
```

GitHub Repository: https://github.com/hkyoung38554/hkyoung38554-MSCS532_Residency_Project_Phase2

References

Ehrgott, M. (2005). Multicriteria Optimization. Springer.

Mandow, L., & De la Cruz, J. L. (2005). A new approach to multiobjective A* search. Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI).

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. ETH Zurich.