**Bidirectional Pathfinding Experiment Report**

טל קליין – 209234103, יובל מירון – 209044148

**Introduction**

This project investigates pathfinding algorithms for pedestrian navigation around Ben-Gurion University (BGU) utilizing OpenStreetMap (OSM) data. The principal algorithm employed is Bidirectional A\* search, which operates by initiating two concurrent A\* searches, one progressing from the origin and the other from the destination, converging at an intermediate point to enhance efficiency. This method employs admissible heuristics, specifically straight-line distance, to ensure optimal routing while substantially decreasing both the search space and computation time relative to single-directional A\* search.

The developed pathfinding system accommodates a range of routing objectives (cost functions), extending beyond shortest-distance calculations to include minimization of traffic light stops, maximization of safety, and reduction of travel time. These varied optimization criteria provide a comprehensive framework for analyzing the impact of each factor on route selection.

**Experiment Goal**

The experiment is structured to evaluate the performance of a Bidirectional A\* pathfinding algorithm across different route optimization objectives. The main hypothesis is that bidirectional search will effectively identify optimal paths for each specified criterion, and that each routing objective will result in distinct route features consistent with its optimization target. Specifically:

* **The Shortest Distance algorithm** generates routes with the minimum walking distance, which may involve more road crossings or traffic lights.
* **The Fewest Traffic Lights algorithm** produces routes with fewer stops at traffic signals or crossings than the shortest distance path, possibly resulting in longer distances or travel times.
* **The Safest Route algorithm** finds paths with lower safety penalties—such as fewer unlit or higher-risk segments, compared to the shortest route, potentially increasing total distance or time due to detours through safer areas.
* **The Fastest Time algorithm** identifies the quickest routes in terms of travel time, which may not correspond to the shortest distance; for instance, it can avoid congested intersections or slow segments to achieve reduced overall travel time.

Overall, the experiment assesses whether each custom cost function (distance, delays, safety, time) reliably directs the A\* search toward an optimal path reflecting the intended priority and quantifies the trade-offs associated with each objective, such as additional distance traveled to minimize signal stops or enhance safety.

**Experiment Description**

Software & Data: The experiment was conducted using Python and a custom OpenStreetMap (OSM) pathfinding system. OSM map data ([**https://github.com/TalKleinBgu/osm-pathfinder-bgu/blob/main/map.osm**](https://github.com/TalKleinBgu/osm-pathfinder-bgu/blob/main/map.osm)) for the Ben-Gurion University campus and adjacent Beer-Sheva neighborhoods was parsed into a pedestrian graph tailored to walking. During parsing, all OSM nodes (with coordinates and tag**s** like traffic signals and lighting) and ways (ordered node lists with tags) are loaded. A walkability filter excludes car-only roads and only includes pedestrian-suitable paths like footways**,** sidewalks, and residential streets.

The designated destination was a specific campus student house (בית הסטודנט), identified by its OSM node ID. Three student neighborhoods in Beer-Sheva—Neighborhood B (שכונה ב), Neighborhood D (שכונה ד), and Old V (ו הישנה), were selected as source areas. For each neighborhood, the locations of all buildings were determined to serve as starting points for routing. In total, approximately 4,724 building-to-campus routes were calculated **(1,181 building \* 4 path = 4724 path).**

Before searching, both endpoints are snapped to the walking graph. For OSM nodes or building ways, the nearest walkable node is found by straight-line distance. building sources use the centroid, also snapped to the closest graph node. This approach keeps searches on graph vertices and avoids mid-segment geometry interpolation.

Algorithm Implementation: Routing uses a **Bidirectional A\***. Two A\* searches run simultaneously: the forward search from the start and the backward search from the goal using the reverse graph. Each maintains its own open queue keyed by f(n)=g(n)+h(n) and a closed set to prevent re-expansion. The implementation keeps *g-scores* (gF, gB) and predecessor maps (predF, predB) on both sides. A priority-queue routine removes entries whose g-score doesn't match the current best score. At every step the side with the smaller current f expands next. After expanding a node, the algorithm (i) **tightens an upper bound (UB)** whenever it discovers a vertex already reached by the opposite search, and (ii) records a **meeting state** that may be either a node (both searches reached the same vertex) or an edge hand-off (forward reached u and backward reached v with an admissible connection across u→v). The global termination rule follows the standard bidirectional A\* invariant:

Correctness and optimality are enforced by admissible heuristics and non-negative edge costs across all profiles, with each profile pairing a suitable heuristic to its cost model and keeping the reverse search symmetric so a front-to-front meet remains optimal. For ***shortest*** ,the heuristic is the Haversine distance, which never overestimates, for ***fastest***, it is Haversine distance divided by a generous upper-bound walking speed (1.6 m/s), ensuring actual segment speeds never exceed the bound, for ***few\_traffic\_lights***, the heuristic scales Haversine distance by distance factor then applies a 0.95 safety margin to stay strictly optimistic even when penalties for entering signal or crossing nodes appear, for ***safest***, the heuristic remains Haversine distance while the edge cost starts at that distance and only adds non-negative safety penalties (lighting, sidewalks, road class, surface, narrowness, crossings), so the adjusted cost is never below the base distance, preserving admissibility.

Execution Details: The experiment was run end-to-end on a standard desktop using Python. For each area of interest (the three Be’er-Sheva neighborhoods), the pipeline loads the OSM map, parses nodes/ways, and builds a pedestrian graph in both directions (forward and reverse). Each building is then snapped to its nearest walkable graph vertex, the fixed destination (the BGU Student house, OSM id 135310103) is likewise snapped to the closest walkable node. From that start–goal pair the system executes **four independent searches**—shortest, few\_traffic\_lights, safest and fastest—each via proper **Bidirectional A\*** with the profile’s admissible heuristic. Because the graph window is compact (≈2 km around campus) and bidirectional search halves effective depth, queries complete on the **millisecond scale** per search. During search the solver records diagnostics (forward/backward node expansions, edges scanned) and, after reconstructing the path, computes uniform **route metrics** over the final node sequence: geometric distance, time, counts of traffic signals and crossings, cumulative safety penalties, detour factor (vs. straight-line), and realized average walking speed.

All source code, scripts, and raw data files used for this experiment are publicly available in GitHub repository -   
[**https://github.com/TalKleinBgu/osm-pathfinder-bgu**](https://github.com/TalKleinBgu/osm-pathfinder-bgu)

Additionally, interactive map visualizations and all computed paths can be explored through the following links:

* [**https://talkleinbgu.github.io/osm-pathfinder-bgu/NeighborhoodB.html**](https://talkleinbgu.github.io/osm-pathfinder-bgu/NeighborhoodB.html) **–** displays all computed paths from Neighborhood B to the campus
* [**https://talkleinbgu.github.io/osm-pathfinder-bgu/NeighborhoodD.html**](https://talkleinbgu.github.io/osm-pathfinder-bgu/NeighborhoodD.html) **–** shows optimal routes from Neighborhood D
* [**https://talkleinbgu.github.io/osm-pathfinder-bgu/NeighborhoodOldV.html**](https://talkleinbgu.github.io/osm-pathfinder-bgu/NeighborhoodOldV.html) **–** presents the route visualizations for Old V neighborhood

**Results**

Despite all routes sharing the same destination, the outcomes demonstrate clear differences in path choices and metrics for the four algorithms, confirming the experiment’s expectations.

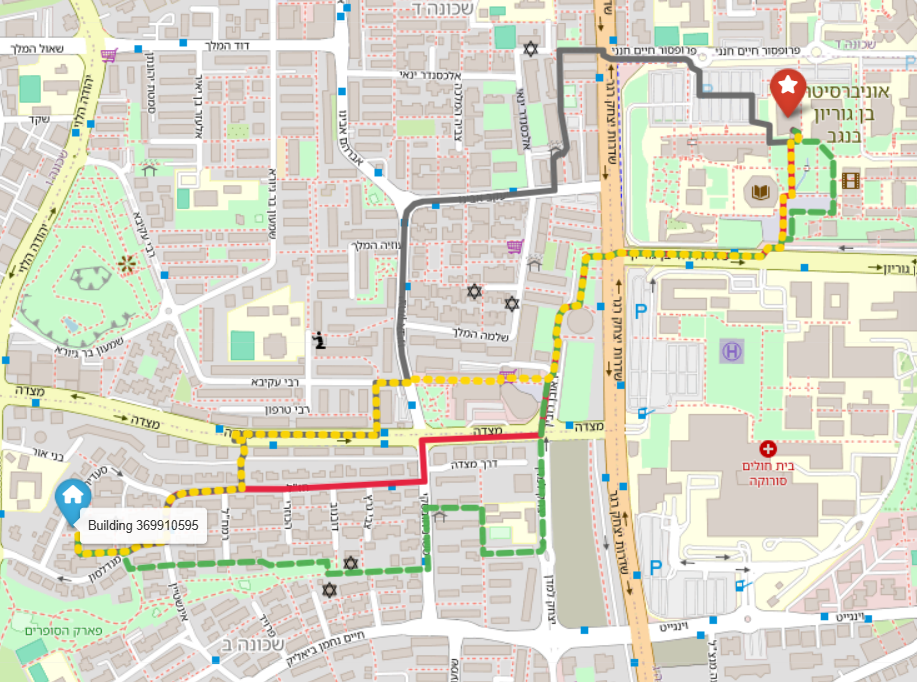
All statistics for every buildings are located in the link - <https://github.com/TalKleinBgu/osm-pathfinder-bgu/blob/main/neighborhood_pathfinding_detailed_data.csv>

The next table summarizes an example set of results for a representative building (369910595) under each routing profile:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Path Distance (m) | Travel Time (min) | Traffic Signals | Detour factor | Num nodes in path | Avg walking speed | Signal density per 100km | Edge scanned | Node expanded |
| **Shortest (distance)** | 1787 | 25.07 | 6 | 1.33 | 80 | 4.28 | 0.34 | 1357 | 536 |
| **Fewest Traffic Lights** | 2152 | 27.76 | 2 | 1.6 | 114 | 4.65 | 0.09 | 7539 | 3257 |
| **Safest Route** | 2178 | 28.28 | 5 | 1.62 | 76 | 4.62 | 0.23 | 2962 | 1213 |
| **Fastest Time** | 1852 | 24.5 | 3 | 1.38 | 86 | 4.53 | 0.16 | 2395 | 981 |

\* **Detour factor** is (route distance ÷ straight-line (haversine) distance); 1.0 is perfectly direct, 1.3 means the walk is 30% longer than the haversine.

The **Shortest** profile provides the geometric baseline (1,787 m in 25.07 min) but traverses 6 signals, which depresses realized speed (4.28 km/h). Optimizing for **Fewest Traffic Lights** does exactly what it promises—signals drop to 2 and signal density plunges from 0.34 to 0.09 per 100 km, but at a notable cost: distance grows by to 2,152 m, time to 27.76 min, and search work inflates substantially (≈3,257 expansions, ~6× baseline). The **Safest** profile lands in the middle: distance rises to 2,178 m, time to 28.28 min, and signals fall modestly (to 5), with realized speed improving to 4.62 km/h; computational effort is moderate (1213 expansions). **Fastest** achieves the most practical efficiency: with only a small geometric detour (1,851 m) it halves signal encounters (3 vs. 6) and finishes faster than Shortest, at a modest computational. In short, for this building the profiles trace a clean trade-off surface: ***Shortest*** minimizes meters, ***Fewest-Lights*** minimizes interruptions at the price of extra distance, ***Fastest*** converts targeted signal avoidance into a small but real time win.



The figure shows four alternatives from **Building 369910595** (house icon, left) to the **BGU student house** (red star, right).

The **Shortest** route (red) goes mostly straight east, then into campus. It crosses the big main road near the university at several traffic lights. So it is shortest in meters, but you have to stop many times.

The **Fastest** route (yellow) uses much of the same streets as red, but makes two small turns to reach quicker crossings and wider entries to the student building. These small detours reduce long waits, so you arrive sooner with only a little extra distance.

The **Fewest Traffic Lights** route (dark blue) goes north earlier, follows a longer, smoother street, crosses the main road once, and comes in from the north side of campus. It has almost no stops, but it is the most indirect path.

The **Safest** route (green) prefers streets with lights and sidewalks, and it uses marked crossings. It passes near community places like the park and nearby schools/playgrounds, where there are people and better lighting. This adds a small detour for a more comfortable walk.

All four routes meet near the same campus entrance. The main differences come from how each one handles the busy, traffic-light area in the middle of the trip.

**When we aggregate by neighborhood, the same patterns hold while the magnitude of the trade-offs depends on local street morphology.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Neighborhood | Algorithm | Path Distance (m) | Travel Time (min) | Traffic Signals | Detour factor | Num nodes in path | Avg walking speed | Signal density per 100km | Edge scanned | Node expanded |
| **B** | Shortest (distance) | 1677.68 | 23.2 | 5.04 | 1.37 | 80.71 | 4.33 | 0.31 | 907.91 | 2250.01 |
| Fewest Traffic Lights | 1919.29 | 25.39 | 3.16 | 1.57 | 89.16 | 4.52 | 0.18 | 3010.17 | 7034.16 |
| Safest Route | 1979.2 | 26.1 | 5.11 | 1.61 | 84.26 | 4.53 | 0.27 | 1203.96 | 2955.98 |
| Fastest Time | 1652.46 | 23.71 | 6.7 | 1.34 | 75.87 | 4.17 | 0.42 | 517.82 | 1312.96 |
| **D** | Shortest (distance) | 1356.74 | 17.79 | 2.63 | 1.44 | 59.24 | 4.55 | 0.21 | 544.23 | 1397.53 |
| Fewest Traffic Lights | 1426.14 | 18.58 | 2 | 1.52 | 69.91 | 4.57 | 0.15 | 1576.2 | 3903.07 |
| Safest Route | 1565.76 | 19.97 | 2.6 | 1.65 | 63.25 | 4.68 | 0.18 | 986.04 | 2500.22 |
| Fastest Time | 1353.51 | 18.43 | 3.37 | 1.44 | 61.77 | 4.38 | 0.27 | 360.79 | 932.74 |
| **Old V** | Shortest (distance) | 997.16 | 12.46 | 0.02 | 1.76 | 53.86 | 4.81 | 0 | 342.45 | 876.18 |
| Fewest Traffic Lights | 996.73 | 12.58 | 0.02 | 1.76 | 52.06 | 4.76 | 0 | 317.27 | 809.79 |
| Safest Route | 1154.19 | 14.58 | 0.5 | 2.01 | 51.66 | 4.77 | 0.04 | 591.81 | 1500.15 |
| Fastest Time | 992.01 | 12.63 | 0.24 | 1.75 | 51.69 | 4.72 | 0.02 | 302.16 | 772.9 |

When we aggregate by neighborhood, the same patterns hold while the magnitude of the trade-offs depends on local street morphology. In **Neighborhood B** (the farthest and most signal-dense), path lengths sit around 1.6–1.9 km and signal exposure is highest, so the objectives separate the most: Fewest-Lights yields the largest detour (detour factor ≈ 1.57) and the heaviest search footprint (≈3,010 edges scanned; ≈7,034 expansions. **Neighborhood D** is intermediate: distances ~1.35–1.43 km and baseline signal exposure are lower, so all profiles cluster closer to Shortest; Fewest-Lights still trims signal density (≈ 0.21→0.15) but with a smaller detour and moderate extra computation. **Old V** is the closest (≈1.0–1.1 km) and essentially signal-free (densities ≈0–0.03), so the profiles nearly collapse to the same solution; only Safest accepts a slight detour (≈ 2.01) to honor infrastructure preferences, and search costs are uniformly low.

**Conclusion**

This study demonstrates that a properly engineered **Bidirectional A\***, paired with profile-specific but admissible cost models, can compute thousands of optimal pedestrian routes on real OSM data quickly and reliably. Across **4,724 paths** (≈1,181 buildings × 4 profiles), each objective produced routes whose characteristics matched the intended priority: ***Shortest*** minimized distance but encountered the most regulated junctions; ***Fewest Traffic Lights*** reliably suppressed stop exposure (and raised realized walking speed) at the cost of longer, more circuitous paths and heavier search; ***Safest*** favored infrastructure quality—lighting, sidewalks, controlled crossings, accepting moderate detours; and ***Fastest*** achieved small yet consistent time gains by bypassing a subset of signals with minimal added distance. These trends held at both the single-building and neighborhood scales, confirming the hypotheses about how each cost function steers the search.

The **magnitude** of those trade-offs depended on local street morphology. **Neighborhood B**—farthest and most signal-dense—amplified differences: the few-lights profile delivered the largest reduction in signal exposure but demanded the biggest detours and the most computation, while fastest preserved near-baseline geometry and lean search effort. **Neighborhood D** showed the same ordering with smaller gaps, and **Old V**—closest and nearly signal-free—caused the profiles to collapse to near-identical routes, with only the safety profile choosing mild detours to honor infrastructure preferences.

Practically, these findings translate into clear guidance. When **arrival time** is paramount and the grid is moderately regulated, *Fastest* is a sensible default: it trims a few delays without meaningfully lengthening the walk. When **continuity** (few stops) matters—e.g., with strollers, running, or user preference—*Fewest Lights* is appropriate, with the transparent trade-off of more meters and minutes. Where **comfort and perceived safety** dominate (nighttime or unfamiliar areas), *Safest* provides infrastructure-aligned paths at moderate overhead. And in low-signal settings, *Shortest* (or *Fastest*) is typically sufficient since all profiles converge.

**Appendix: Key source code**

Below are the **essential, self-contained** snippets and patterns that implement the system’s core behavior. Each excerpt is trimmed to the minimum needed to understand structure, flow, and design trade-offs; it matches your repository exactly in naming and logic.

The whole code is in the link - [**https://github.com/TalKleinBgu/osm-pathfinder-bgu**](https://github.com/TalKleinBgu/osm-pathfinder-bgu)

The results json (with paths) in the link - <https://github.com/TalKleinBgu/osm-pathfinder-bgu/tree/main/results>

**1) Core data model**

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

**2) OSM parsing and pedestrian graph build (forward + reverse)**

**תמונה שמכילה טקסט, צילום מסך

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.**

תמונה שמכילה טקסט, צילום מסך, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

**3) Safety context (public places index) and geometry**



**4) Cost models (four profiles, symmetric and admissible)**

תמונה שמכילה טקסט, צילום מסך

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.



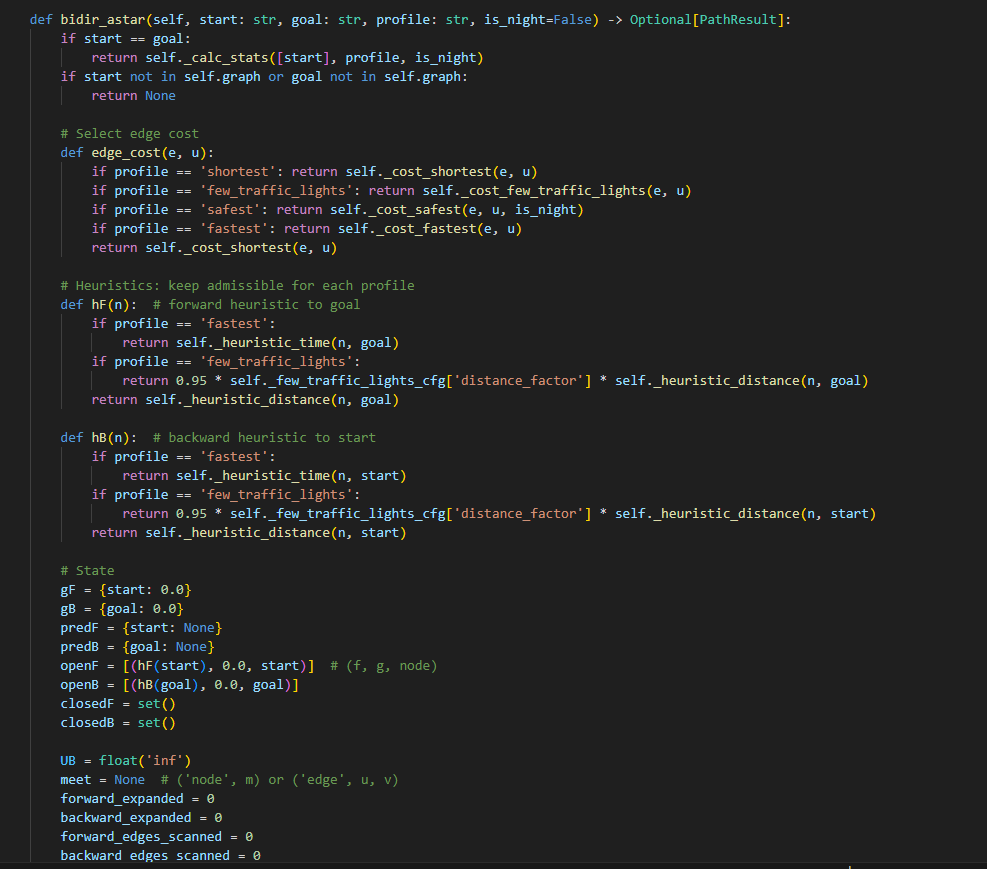


**Admissible heuristics (paired to profile):**

תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

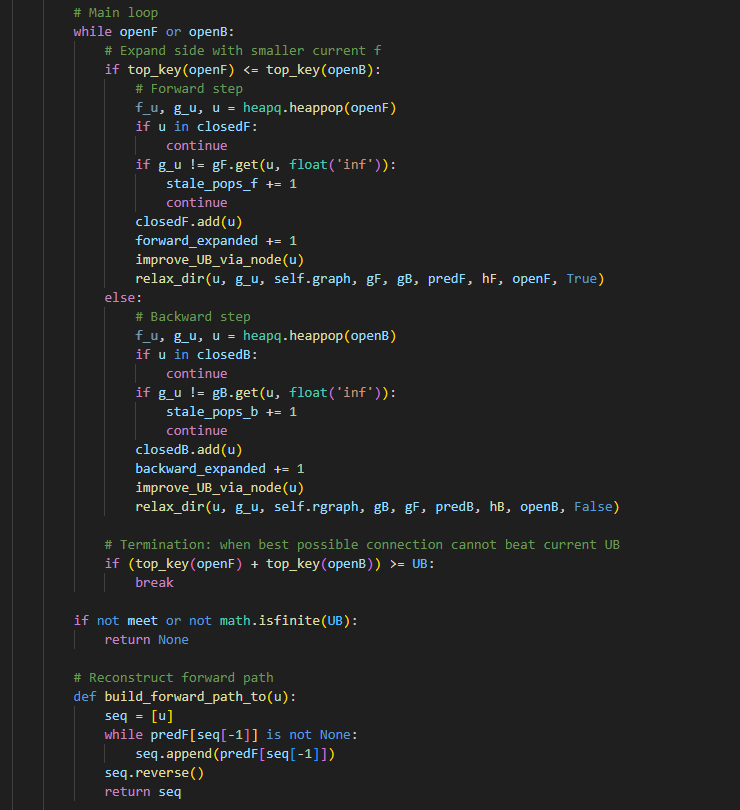
תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

**5) Bidirectional A\***



תמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.



תמונה שמכילה טקסט, צילום מסך, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

**6) Uniform metric computation (comparable times across profiles)**

תמונה שמכילה טקסט, צילום מסך, תוכנה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.