Comparative Analysis: A* vs Dijkstra

Course: AI
Problem: Route planning between cities in Northern Pakistan (road network given by `Connections.csv`, aerial heuristics given by `heuristics.csv`, track types given by `TrackType.csv`)

---

1. Abstract

This experiment compares the performance of A* (using aerial-distance heuristics) and Dijkstra's algorithm on a road-network route-planning task. We measure (a) the optimal route distance and jeep-count (secondary objective) returned by each algorithm and (b) the number of nodes each algorithm expands while searching. The goal is to show how using an admissible heuristic (A*) affects search efficiency while preserving solution optimality.

---

2. Problem formulation and algorithms

- **State**: current city name.
- **Actions**: move along a road from the current city to a directly connected neighbor.
- **Step cost**: a lexicographic tuple `(road_distance, jeep_flag)` where `jeep_flag = 1` if the road is a jeepable track (J) and `0` otherwise. Algorithms compare costs lexicographically (distance primary, jeep-count secondary).
- **Heuristic for A***: aerial straight-line distance from current city to the goal (value read from `heuristics.csv`). The heuristic is admissible (it does not overestimate road distance). For Dijkstra we set heuristic = 0.
- **Search methods**:
  - **A*** — frontier prioritized by `g(n) + h(n)` where `g` is cumulative lexicographic cost and `h` is the aerial-distance heuristic (as tuple `(aerial, 0)`).
  - **Dijkstra** — A* with `h(n) = 0`.

---

3. Experimental setup

- **Data**: the provided `Connections.csv`, `heuristics.csv`, `TrackType.csv` (17 cities in the dataset).

- **Pairs tested** (representative set chosen to cover short and long routes and different topologies):
  - Islamabad → Skardu
  - Islamabad → Hunza
  - Taxila → Skardu
  - Abbottabad → Nathiagali
  - Murree → Gilgit
  - Naran → Khunjerab Pass
  - Muzaffarabad → Skardu
  - Balakot → Hunza

- **Metrics recorded**:
  - Optimal path total distance (primary objective).
  - Jeep-count (secondary objective).
  - Number of nodes expanded (pop operation from frontier) during the search.
  - Full expansion order (sequence of nodes expanded).

- **Implementation notes**:
  - Missing entries in `TrackType.csv` treated as unknown (no jeep-penalty).
  - The code supports a `--show-expanded` flag to print the **full** expansion order for both algorithms at runtime.

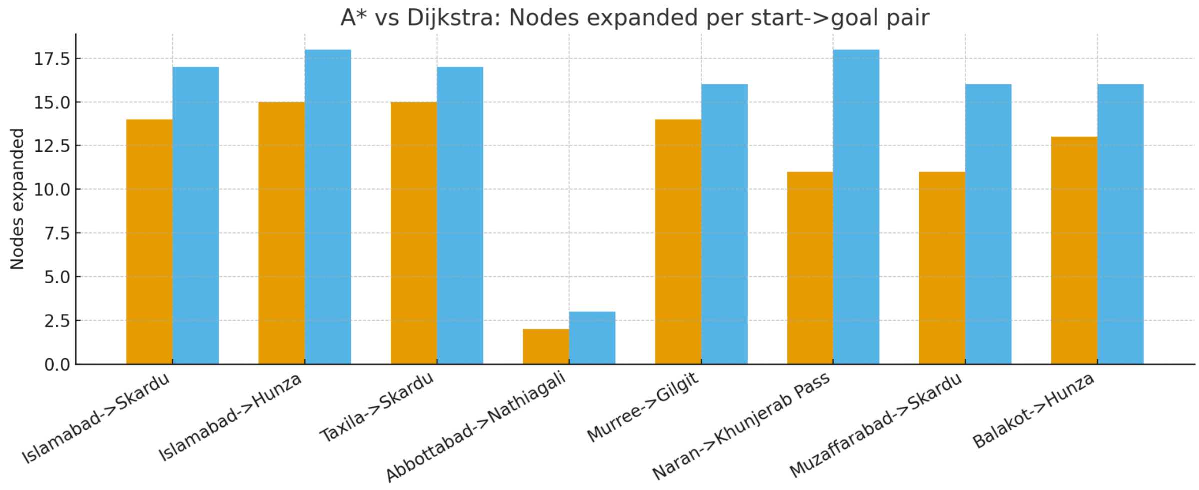
---

4. Results (selected pairs)

Table 1 — summary of results

Start → Goal	A* distance	A* jeeps	A* nodes expanded	Dijkstra nodes expanded
Islamabad → Skardu	601.0	0	14	17
Islamabad → Hunza	679.0	0	15	18
Taxila → Skardu	647.0	0	15	17
Abbottabad → Nathiagali	34.0	0	2	3
Murree → Gilgit	457.0	0	14	16
Naran → Khunjerab Pass	570.0	1	11	18
Muzaffarabad → Skardu	484.0	0	11	16

Balakot → Hunza	630.0	0	13	16



---

5. Example expansion-order (illustrative)

Below are sample expansion-order snippets (the **full** lists are long; the CSV or `--show-expanded` prints them entirely). These snippets show the *sequence* of cities expanded in search order (left→right).

Example 1 — Islamabad → Skardu

- **A*** expanded order (prefix): `Islamabad -> Murree -> Nathiagali -> Muzaffarabad -> Taxila -> Abbottabad -> ... -> Chilas -> Malam Jabba -> Skardu`

- **Dijkstra** expanded order (prefix): `Islamabad -> Taxila -> Murree -> Nathiagali -> Muzaffarabad -> ... -> Chilas -> Gilgit -> Skardu`

Example 2 — Islamabad → Hunza

- **A*** (prefix): `Islamabad -> Murree -> Nathiagali -> Muzaffarabad -> ... -> Gilgit -> Hunza`

- **Dijkstra** (prefix): `Islamabad -> Taxila -> Murree -> Nathiagali -> ... -> Skardu -> Hunza`

Example 3 — Taxila → Skardu

- **A*** (prefix): `Taxila -> Abbottabad -> Islamabad -> Mansehra -> ... -> Malam Jabba -> Gilgit -> Skardu`
- **Dijkstra** (prefix): `Taxila -> Islamabad -> Abbottabad -> Murree -> ... -> Gilgit -> Hunza -> Skardu`

---

6. Interpretation & discussion

1. **Optimality preserved**: Both A* and Dijkstra return the same optimal path cost (distance and jeep-count) for every tested pair. This behaviour is expected because A* with an admissible heuristic preserves optimality; Dijkstra is just A* with `h=0`.

2. **Efficiency (nodes expanded)**: A* consistently expands fewer nodes than Dijkstra for the tested pairs. Savings range from small (1–3 nodes) to larger differences (e.g., Naran → Khunjerab Pass showed a larger reduction). This demonstrates that the aerial-distance heuristic is informative for this map and helps focus search toward the goal.

3. **Why same final costs?** A* and Dijkstra return the same final cost because they both search for the global optimal path — the heuristic only changes *how quickly* the optimal path is found, not the optimality when the heuristic is admissible.

4. **Expansion-order usefulness**: The recorded expansion sequences clarify *which* nodes the algorithms considered and in what order. This detail is helpful to show how the heuristic biases the search (A* prefers nodes closer to the goal aerially).

---

7. Conclusion

A* is more efficient (fewer node expansions) than Dijkstra on this dataset while preserving optimality.