CSC4120 Group Project Report

Party Together Problem (PTP)

Rui Meng 122090405 Yike Chen 123090047 Chunrui Pan 123090438

December 2025

Contents

1 Project Description

1.1 Problem Setting

We are given an undirected weighted graph $G = (V, E)$ (nodes are locations, edge weights are nonnegative road lengths). Node 0 is the party owner's home. Let $H \subseteq V$ be the set of friends' home locations (each home corresponds to exactly one friend). A constant $\alpha \in [0, 1]$ represents the relative cost of driving (car) versus walking.

A solution must specify, for each friend with home $h \in H$, a pickup location $p \in V$ satisfying the *local pickup constraint*:

$$p \in \{h\} \cup N(h),$$

i.e., the pickup is either at the friend's home or at a neighboring node of that home.

The car must start at node 0, traverse a *tour* (nodes/edges may repeat), visit all pickup locations, and finally return to node 0.

1.2 Objective Function

Let the car tour be $\tau = (u_0, u_1, \ldots, u_n)$ with $u_0 = u_n = 0$ and each consecutive pair $(u_{i-1}, u_i) \in E$. Let w_{uv} be the edge weight, and let $d(x, y)$ denote the shortest-path distance in G. If each friend at home h_m is assigned pickup p_m, the total cost is:

$$\text{Cost}(\tau, \{p_m\}) \;=\; \alpha \sum_{i=1}^{n} w_{u_{i-1}u_i} \;+\; \sum_{m} d(p_m, h_m).$$

The goal is to minimize this total cost.

2 Approach (PTP Solver)

2.1 High-level idea

Our solver is a heuristic local-search method that directly optimizes the PTP objective. Instead of jointly solving the car route and pickup assignments in one large combinatorial problem, we maintain a *macro tour* (a short sequence of key nodes beginning and ending at 0), and evaluate this tour by assigning each friend to the closest feasible pickup location that already appears on the tour.

Concretely, for each home node $h \in H$, the set of allowed pickup nodes is $\{h\} \cup N(h)$, i.e., the friend can be picked up either at home or at a neighboring node. Given a candidate macro tour τ, we compute: (i) a driving cost equal to α times the shortest-path distance between consecutive nodes of τ, and (ii) a walking cost where each friend chooses the closest allowed pickup node that lies on the tour. This yields a total objective value $\alpha \cdot \text{Drive}(\tau) + \text{Walk}(\tau)$.

Starting from the trivial tour $[0, 0]$, we apply iterative improvement via two neighborhood moves: *(1) insertion* of a node not currently on the tour, and *(2) deletion* of an existing non-zero node from the tour. At each iteration, we scan all nodes and select the best improving candidate according to a lexicographic rule: first minimize the number of infeasible friends (friends with no allowed pickup on the current tour), and then minimize the total cost. We stop when no single insertion/deletion yields improvement.

Finally, after the macro tour is optimized, we *expand* it into a valid step-by-step tour that only uses edges in the input graph by concatenating shortest paths between consecutive macro nodes. We also run a feasibility repair step to guarantee that every friend has at least one allowed pickup node appearing on the final tour, and output the pickup mapping as required.

2.2 Implementation details

Graph preprocessing. Although the input is a directed graph, it is equivalent to an undirected one by construction. We build an undirected weighted graph and precompute all-pairs shortest-path distances $d(u, v)$ and corresponding shortest paths using Dijkstra. Distances are used to score candidate macro tours, while shortest paths are used to expand the final tour.

Macro tour representation. A macro tour is a list of nodes that begins and ends at 0. We apply a small normalization step that removes consecutive duplicates and enforces boundary nodes to be 0. This keeps the representation compact and avoids degenerate moves.

Tour evaluation. Given a macro tour $\tau = (u_0, \ldots, u_k)$, the driving term is computed as

$$\alpha \sum_{i=1}^{k} d(u_{i-1}, u_i).$$

Let T be the set of nodes on τ. For each friend home h, we compute the allowed pickup set $A(h) = \{h\} \cup N(h)$ and intersect with T: if $A(h) \cap T \neq \emptyset$, we assign h to the pickup node

$$p(h) \in \arg \min_{p \in A(h) \cap T} d(h, p)$$

and add $d(h, p(h))$ to the walking cost. If $A(h) \cap T = \emptyset$, the tour is marked infeasible for this friend. During local search we still compute a fallback walking distance to the nearest tour node as a generalized score, but feasibility is explicitly tracked and prioritized.

Local search moves. We iterate over all candidate nodes in the graph (excluding 0). If a node is already present in the tour, we consider deleting it; otherwise, we consider inserting it into the best position. For insertion, we test all possible insertion indices and keep the one with the best evaluation. A candidate is accepted only if it strictly improves according to:

- If the current solution is infeasible, prefer any move that reduces the infeasible count.
- If the current solution is feasible, accept only feasible moves that reduce the total cost.

This produces a monotonic improvement process and terminates at a local optimum under these moves.

Feasibility repair and output construction. After local search, we repair the tour if some friend has no allowed pickup node on the tour. For each missing home h, we greedily insert h into the macro tour position that minimizes the increase in driving distance (based on shortest-path distances):

$$\Delta = d(a, h) + d(h, b) - d(a, b),$$

where (a, b) is a consecutive pair in the current macro tour. After all repairs, every friend has at least one allowed pickup node on the tour.

We then expand the macro tour into an edge-by-edge tour by concatenating shortest paths. Finally, we build the required dictionary output $\{\text{pickup node} \mapsto [\text{homes}]\}$ by assigning each home h to the closest allowed pickup node that appears on the (repaired) tour.

3 Theoretical Questions

3.1 Q5.1: PTP is NP-hard

Theorem 1. *PTP is NP-hard.*

Proof. We reduce from PHP, which is NP-hard (it contains Metric TSP as a special case).

Given an instance (G, H) of PHP, we construct an instance (G, H, α) of PTP on the same graph and same set of homes, but choose $\alpha > 0$ to be sufficiently small.

Assume edge weights are integers and strictly positive[1], so any nonzero walking distance is at least 1. Let B be an upper bound on the length of any feasible tour in G that starts/ends at 0; for example, one can take $B = 2(|V| - 1) \cdot w_{\max}$ (visit nodes via a spanning tree traversal), which is polynomially bounded.

Pick α such that $0 < \alpha < \frac{1}{2B}$. Then for any feasible PTP solution:

- If *any* friend walks a positive distance, the walking term is at least 1, so total cost ≥ 1.
- There exists a feasible solution with *zero* walking: pick each friend up at their home ($p_m = h_m$) and drive any tour that visits all homes. Its cost equals $\alpha \cdot (\text{tour length}) \leq \alpha B < \frac{1}{2}$.

Therefore, any optimal PTP solution must have total walking cost 0 (otherwise it would cost at least 1 and be worse than the $< \frac{1}{2}$ solution).

Walking cost $\sum_m d(p_m, h_m) = 0$ implies $d(p_m, h_m) = 0$ for all m, hence $p_m = h_m$ for all friends. So an optimal PTP solution must pick everyone up at home and thus the car tour must visit every node in H.

Among solutions with zero walking, minimizing the PTP objective is exactly minimizing the driving tour length, which is precisely the PHP objective. Hence, an oracle that solves PTP can be used to solve PHP in polynomial time. Since PHP is NP-hard, PTP is NP-hard. □

3.2 Q5.2: PHP cost is at most twice of optimal PTP cost (and tight)

In this question we assume $\alpha = 1$ for simplicity. Let C_{php} be the optimal cost of PHP on (G, H) and C_{ptp} be the optimal cost of PTP on $(G, H, \alpha = 1)$. Define $\beta = C_{\text{php}}/C_{\text{ptp}}$.

Theorem 2. *For $\alpha = 1$, we have $\beta \leq 2$.*

Proof. Let an optimal PTP solution consist of:

- a car tour τ with total driving length D,
- pickup assignments $\{p_m\}$ with total walking cost $W = \sum_m d(p_m, h_m)$.

Thus $C_{\text{ptp}} = D + W$.

We construct a feasible PHP tour (must visit every home) from this PTP solution: follow the PTP tour τ, and for each friend m, when the car reaches p_m, add a detour: go from p_m to h_m along a shortest path (length $d(p_m, h_m)$) and return back to p_m along the reverse path. This ensures the tour visits every home in H.

The added driving length for friend m is at most $2\,d(p_m, h_m)$, so the constructed PHP tour has length

$$D' \leq D + 2W.$$

[1]This is without loss of generality for NP-hardness: restricting to positive integer weights remains NP-hard for the TSP-type problems we reduce from.

Therefore, the optimal PHP cost satisfies

$$C_{\text{php}} \le D' \le D + 2W \le 2(D + W) = 2C_{\text{ptp}},$$

which implies $\beta \le 2$. □

Tightness (asymptotically $\beta = 2$). Consider a star graph with center 0 and k leaves, where every leaf is a home (so $|H| = k$) and each edge $(0, \text{leaf})$ has weight 1. With $\alpha = 1$:

- In PTP, pick everyone up at node 0 (allowed since each leaf's neighbor includes 0): car drives length 0, walking cost is k, so $C_{\text{ptp}} = k$.
- In PHP, the car must visit every leaf. Since leaves are only connected through 0, any tour must traverse each leaf edge twice, so $C_{\text{php}} = 2k$.

Hence $\beta = \frac{2k}{k} = 2$, achieving the bound.