

Robust Multiagent Combinatorial Path Finding

Technical Appendix

Anonymous submission

Proofs

In this section, we establish the theoretical guarantees of our approach. We begin by proving the correctness and completeness of the K-best-Sequencing procedure (Steps 1–3, including Alg. 2), which serves as the foundation for *RCbssEff*. Building on these results, we then prove that *RCbssEff*, based on the Robust CBSS framework (Alg. 1), is both complete and optimal.

Correctness of Step 1: Reducing the Multi-Agent Goal Sequence Allocation to E-GTSP

In Step 1, we reduce the multi-agent goal sequence allocation problem to an E-GTSP instance. We prove that this reduction preserves feasibility and cost, faithfully representing the original problem.

Notation 1. Let $\text{Tour}(\gamma)$ denote the E-GTSP tour constructed from a multi-agent goal sequence allocation γ using Step 1.

Lemma 1. For any feasible multi-agent goal sequence allocation γ , there exists a corresponding E-GTSP tour $\text{Tour}(\gamma)$ constructed via Step 1.

Proof. Let $\gamma = \langle sq(1), \dots, sq(n) \rangle$ be a feasible allocation, where $sq(i)$ is the ordered sequence of goals assigned to agent i .

Cluster construction. Each goal g_j is a cluster C_{g_j} containing one vertex per feasible approach orientation, ensuring each goal is visited exactly once with a chosen orientation. Each agent i is a singleton cluster C_0^i containing its start position s_i , ensuring all agents are included.

Edge construction. Type-1 edges connect agent starts to goals and goals to each other, with costs matching the original transitions, preserving true movement costs. Zero-cost Type-2 edges connect goals to agent-start clusters and agent-start clusters to each other. They enable concatenating all agents' paths into a single E-GTSP tour and prevent enforcing a fixed destination goal.

Tour construction. We form $\text{Tour}(\gamma)$ by sequentially traversing the clusters based on the sequences $sq(i)$: for each agent i , start at its singleton cluster C_0^i and visit the

orientation-specific copies of its assigned goals in the order they appear in $sq(i)$, selecting exactly one vertex per cluster. After finishing an agent's sequence, use a Type-2 edge to transition to the next singleton cluster C_0^k and continue with the sequence $sq(k)$.

Since every goal in γ appears exactly once across all agents and every agent's start position is included, the constructed path visits one vertex per cluster and is thus a valid E-GTSP tour. Moreover, no fixed destination cluster is enforced, so the tour accurately reflects the original problem's lack of destination goal constraints. \square

Lemma 2. The cost of the E-GTSP tour $\text{Tour}(\gamma)$ is equal to the cost of the original allocation γ .

Proof. The cost of each Type-1 edge between orientation-specific vertices (including edges from agent starts to goals) is inherited directly from the original problem's cost function, which accounts for movement distances and orientations. Zero-cost Type-2 edges do not contribute to the cost. Thus, the total cost of $\text{Tour}(\gamma)$ equals the total cost of executing the allocation γ . \square

Theorem 3. Let γ^* be an optimal multi-agent goal sequence allocation. Then $\text{Tour}(\gamma^*)$ is an optimal tour to the corresponding E-GTSP instance.

Proof. By the previous lemmas, Step 1 produces a valid E-GTSP tour from any feasible allocation while preserving both feasibility and cost. Thus, the optimal allocation γ^* corresponds to a tour $\text{Tour}(\gamma^*)$ with the same minimal cost in the E-GTSP formulation. \square

Completeness of Step 2: The K-best-EGTSP Algorithm (Alg. 2)

In Step 2, we enumerate the k best E-GTSP tours in non-decreasing order of cost. We prove that the K-best-EGTSP algorithm is complete, i.e., it returns all k best tours whenever they exist.

Let \mathcal{S} denote the set of all feasible tours for a given E-GTSP instance. A set of tours $\mathcal{S}_K \subseteq \mathcal{S}$ is a k -best set if (1) $|\mathcal{S}_K| = K$, and (2) every tour $S' \in \mathcal{S} \setminus \mathcal{S}_K$ has cost at least as large as the most expensive tour in \mathcal{S}_K .

Theorem 4. *Given a complete and optimal E-GTSP solver, K -best-EGTSP is guaranteed to return a k -best set of tours for any $k > 0$.*

Proof. Intuitively, K -best-EGTSP enumerates tours by systematically branching on edges of previously generated tours. Because the underlying E-GTSP solver is complete and optimal, each new branch eventually explores all feasible tours in non-decreasing order of cost.

Assume, for contradiction, that K -best-EGTSP fails to return a valid k -best set. Then there exists a returned tour s'_j such that a cheaper tour s_j is missing, i.e., $\text{cost}(s_j) < \text{cost}(s'_j)$.

Case 1 ($K = 1$): The first call to rEGTSP is made with $I_c^0 = O_c^0 = \emptyset$, i.e., directly invoking the base E-GTSP solver. By assumption, the solver returns the optimal tour. If a cheaper tour s_j existed, it would have been selected over s'_j , a contradiction.

Case 2 ($K > 1$): Suppose s'_j is the k -th selected tour. If s_j was already in the priority queue when s'_j was chosen, the best-first ordering would require expanding s_j first, contradicting the selection of s'_j . If s_j was not yet generated when s'_j was selected, then it must eventually be produced in a later iteration, as K -best-EGTSP systematically explores all feasible tours by branching on previously generated ones with added inclusion/exclusion constraints. Crucially, rEGTSP solves these constrained subproblems by temporarily modifying edge costs: edges required by inclusion constraints receive a large negative offset to enforce their selection, while all other edges either retain their original costs or are penalized. After a tour is found, its cost is re-evaluated using the original edge costs, ensuring that no constrained subproblem can yield a tour with a strictly lower true cost than what could have been found in the unconstrained problem. Therefore, if s_j were truly cheaper than s'_j , it would necessarily have been discovered earlier during the enumeration, which contradicts the assumption.

Thus, no such s_j can exist, proving the algorithm's completeness. \square

Correctness of Step 3: Decomposing the E-GTSP Tour into a Multi-Agent Goal Sequence Allocation

In Step 3, we decompose the E-GTSP tour into per-agent goal sequences. We prove that this decomposition preserves feasibility and cost, yielding an equivalent optimal allocation.

Notation 2. Let $\text{Alloc}(\text{tour})$ denote the multi-agent goal sequence allocation obtained by decomposing a given E-GTSP tour using Step 3.

Lemma 5. *For any E-GTSP tour, there exists a corresponding multi-agent goal sequence allocation obtained via the Step 3 decomposition.*

Proof. Let tour be an E-GTSP tour, represented as an ordered sequence of vertices $\{s_0, v_{g_{j,o}}, \dots, s_i, \dots, s_n, \dots\}$, where each s_i is the initial configuration of agent i , and

each $v_{g_{j,o}}$ is an orientation-specific copy of goal j approached from orientation o . Since each initial agent vertex forms a singleton cluster in the E-GTSP formulation, the tour must include exactly n such agent vertices. To construct $\text{Alloc}(\text{tour})$, we partition the tour into n disjoint segments by cutting at each agent vertex. For each agent i , let $sq(i)$ denote the sequence of goal vertices appearing between s_i and the next agent vertex. Each $sq(i)$ is assigned to agent i , preserving the visitation order. Finally, we map each orientation-specific goal vertex $v_{g_{j,o}} \in sq(i)$ to its original goal v_{g_j} , discarding orientation information. This produces a valid multi-agent goal sequence allocation $\text{Alloc}(\text{tour})$. \square

Lemma 6. *The cost of the multi-agent goal sequence allocation is equal to the cost of the original E-GTSP tour.*

Proof. As described in Lemma 5, the allocation is constructed by partitioning the E-GTSP tour into agent-specific segments. Segmentation occurs at Type-2 edges, which are auxiliary and have zero cost. Removing these edges does not affect the total cost. Additionally, mapping orientation-specific goal vertices $v_{g_{j,o}}$ to their original vertices v_{g_j} does not alter the cost: the edge weights between goals in the allocation are inherited from the E-GTSP traversal, which already accounts for orientation-dependent movement. Therefore, the total cost remains unchanged. \square

Theorem 7. *Let tour be an optimal tour to the E-GTSP. Then $\text{Alloc}(\text{tour})$ is an optimal multi-agent goal sequence allocation.*

Proof. By Lemma 5, Step 3 produces a valid allocation from any E-GTSP tour. By Lemma 6, this decomposition preserves the total cost. Thus, if tour is optimal for the E-GTSP, then $\text{Alloc}(\text{tour})$ is also optimal for the corresponding multi-agent goal sequence allocation problem. \square

Completeness of RCBssEff (Based on the Robust CBSS Framework in Alg. 1)

In this section, we prove that RCBssEff is *complete*: if a feasible robust solution exists, the algorithm is guaranteed to return one.

Theorem 8. *Given the correctness and completeness of Steps 1–3 (which ensure the correctness and completeness of the K -best-Sequencing procedure), RCBssEff is solution complete.*

Proof. Assume, for contradiction, that RCBssEff fails to return a solution for some solvable instance. Let π be a feasible robust solution, and let $\gamma(\pi)$ denote its corresponding goal sequence allocation.

RCBssEff maintains a set \mathcal{T} of Constraint Trees (CTs), each representing a unique goal allocation sequence, and systematically explores them using CBS to find feasible solutions.

Case 1: RCBssEff never creates a CT tree for $\gamma(\pi)$. However, RCBssEff enumerates all goal sequence allocations using the K -Best-Sequencing method, which invokes K -best-EGTSP. Since K -best-EGTSP is complete, $\gamma(\pi)$ must even-

tually be generated, ensuring that a CT tree for it is created — a contradiction.

Case 2: A CT tree T for $\gamma(\pi)$ exists, but *RCbssEff* never finds π within it. Since π is a feasible robust solution consistent with $\gamma(\pi)$, it must correspond to some CT node in T . *RCbssEff* systematically expands all nodes in each CT until a robust solution is found or the tree is fully explored. Thus, π must eventually be reached and verified, contradicting the assumption.

Therefore, *RCbssEff* is complete. \square

Optimality of *RCbssEff* (Based on the Robust CBSS Framework in Alg. 1)

Next, we prove that the solution returned by *RCbssEff* is *optimal* with respect to the defined cost function.

Theorem 9. *Given the correctness and completeness of Steps 1–3 (which ensure the correctness and completeness of the K-best-Sequencing procedure), the solution returned by *RCbssEff* is optimal.*

Proof. Let Z denote the CT node corresponding to the solution π returned by *RCbssEff*. By construction, π consists of tours where each agent visits all its assigned goals. As Z was expanded, every other CT node $Z' \in \text{OPEN}$ satisfied $\text{cost}(Z') \geq \text{cost}(Z)$.

Assume, for contradiction, that there exists another solution π' with $\text{cost}(\pi') < \text{cost}(\pi)$. Let $\gamma(\pi')$ denote the goal allocation sequence of π' .

Case 1: A CT tree for $\gamma(\pi')$ already exists. Then there must be an unexpanded node Z_T in this tree with $\text{cost}(Z_T) \leq \text{cost}(\pi')$. But since *RCbssEff* selected Z for expansion, we must have $\text{cost}(Z) \leq \text{cost}(Z_T)$, which contradicts $\text{cost}(\pi') < \text{cost}(\pi)$.

Case 2: No CT tree for $\gamma(\pi')$ has been generated. Then $\text{cost}(Z) \leq \text{cost}(\gamma(\pi'))$, where $\text{cost}(\gamma(\pi'))$ is the minimal cost of its allocation. Since $\text{cost}(\gamma(\pi')) \leq \text{cost}(\pi')$, it follows that $\text{cost}(\pi) \leq \text{cost}(\pi')$, again contradicting the assumption.

Thus, no cheaper solution exists, and *RCbssEff* returns an optimal solution. \square