Robust Multiagent Combinatorial Path Finding Technical Appendix

Paper #9093

Proofs

This section proves the completeness of K-best-EGTSP and the completeness and optimality of Robust CBSS.

1 Completeness of the K-best-EGTSP Algorithm (Alg. 2)

Let $\mathcal{S}$ denote the set of all feasible tours to a given *E-GTSP* instance. A set of tours $\mathcal{S}_K \subseteq \mathcal{S}$ is a *k-best* set of tours if (1) $|\mathcal{S}_K| = K$, and (2) every tour $S' \in \mathcal{S} \setminus \mathcal{S}_K$ has a cost that is larger than or equal to the cost of every tour in $\mathcal{S}_K$.

Theorem 1. *Given a complete and optimal E-GTSP solver, K-best-EGTSP is guaranteed to return a k best set of tours if such exists for any $k > 0$.*

Proof. Assume, for contradiction, that the *K-best-EGTSP* algorithm is not complete. Then there exists a case where the set $\mathcal{S}_K$ returned by *K-best-EGTSP* includes a tour s'_j such that there exists another tour s_j not in $\mathcal{S}_K$, with $\text{cost}(s_j) < \text{cost}(s'_j)$.

Case 1 ($K = 1$): The initial call to `rEGTSP` is made with $I_c^0 = \emptyset$ and $O_c^0 = \emptyset$, effectively invoking the base *E-GTSP* solver on G'. By assumption, the *E-GTSP* solver is complete and optimal, and thus it must return the optimal tour. If $\text{cost}(s_j) < \text{cost}(s'_j)$, the solver would return s_j over s'_j, contradicting optimality.

Case 2 ($K > 1$): $s'_j = tour^k$ for some $k > 1$. If both s_j and s'_j are present in the priority queue $OPEN_{\text{kBEST}}$ at the moment s'_j is selected, the algorithm's best-first selection rule requires choosing s_j before s'_j, since $\text{cost}(s_j) < \text{cost}(s'_j)$. Selecting s'_j while s_j is available with a lower cost contradicts the priority queue's ordering. If s_j is not present in the queue when s'_j is selected, the algorithm will eventually generate s_j in a later iteration, since it is a feasible tour. By adding further constraints, the algorithm reaches s_j with a lower cost than s'_j, implying that the additional constraints led to a better tour. However, `rEGTSP` solves constrained instances by temporarily adjusting edge costs—specifically reducing the costs of inclusion-constrained edges to enforce their selection—and afterward restores these edges to their original costs, ensuring that no constraint can produce a strictly better tour than what the unconstrained solver would have found. Therefore, the solver should have found s_j earlier without the extra constraints, contradicting its optimality. □

2 Completeness and Optimality of *RobustCbss* (Alg. 1)

Theorem 2. *Given that the K-best-EGTSP algorithm is complete, RobustCbss is solution complete and optimal. That is, if an instance is solvable, then RobustCbss is guaranteed to return an optimal solution.*

Proof. **We first prove that *RobustCbss* will return a solution if such exists and then prove that this solution is optimal**.

Assume, for contradiction, that *RobustCbss* is not complete. Then there exists a solvable problem instance for which *RobustCbss* fails to return a solution. Let π be a feasible robust solution for this instance, and let $\gamma(\pi)$ denote the corresponding goal sequence allocation.

RobustCbss maintains a set $\mathcal{T}$ of CT search trees, where each tree corresponds to a unique goal allocation sequence, and systematically explores them to find feasible solutions.

Case 1: *RobustCbss* never creates a CT tree for $\gamma(\pi)$. However, *RobustCbss* systematically enumerates all goal sequence allocations by invoking the K-Best-Sequencing method, which in turn calls *K-best-EGTSP*. Since *K-best-EGTSP* is complete, $\gamma(\pi)$ must eventually be returned, ensuring that a CT tree is created for it. Otherwise, this would contradict the completeness of the K-Best-EGTSP solver.

Case 2: A CT tree T corresponding to $\gamma(\pi)$ is created, but *RobustCbss* never finds π within it. Since π is a feasible robust solution for $\gamma(\pi)$, it must correspond to some CT node in T. *RobustCbss* systematically expands all nodes in T (using CBS) until a robust solution is found or the tree is fully explored. Therefore, *RobustCbss* must eventually reach and verify π during this process, contradicting the assumption that π is never found.

Next, we prove that π, the solution returned by *RobustCbss*, is optimal.

Let Z be the corresponding CT node. By construction, π is a set of tours, each agent following a tour that visits all its assigned goals. As Z was expanded, every other CT node $Z' \in$ `OPEN` satisfies $\text{cost}(Z') \geq \text{cost}(Z)$.

Assume, for contradiction, that there exists a solution π' with $\text{cost}(\pi') < \text{cost}(\pi)$. Let $\gamma(\pi)$ and $\gamma(\pi')$ denote the goal sequence allocations corresponding to π and π', respectively.

RobustCbss maintains a set $\mathcal{T}$ of CT search trees, where each tree corresponds to a unique goal allocation sequence. These trees are monotonic: the cost of every CT node is less than or equal to the cost of its descendants.

Case 1: There exists a CT tree $T \in \mathcal{T}$ corresponding to the same goal allocation sequence as π'. Since π' has not yet been found, there must exist an unexpanded node $Z_T \in T$ in `OPEN` with $\text{cost}(Z_T) \leq \text{cost}(\pi')$. But since *RobustCbss* selected Z for expansion, we have $\text{cost}(\pi) = \text{cost}(Z) \leq \text{cost}(Z_T)$, which contradicts the assumption that $\text{cost}(\pi') < \text{cost}(\pi)$.

Case 2: π' is not in any tree generated so far, meaning its goal allocation sequence has not yet been explored. In this case, we know that $\text{cost}(Z) \leq \text{cost}(\gamma(\pi'))$, where $\text{cost}(\gamma(\pi'))$ denotes the minimal cost over the goal allocation of π', and $\text{cost}(\gamma(\pi')) \leq \text{cost}(\pi')$.

This implies $\mathrm{cost}(\pi) \leq \mathrm{cost}(\pi')$, contradicting the assumption that $\mathrm{cost}(\pi') < \mathrm{cost}(\pi)$. □

3 Correctness of the mTSP to *E-GTSP* Transformation

Let mTSP(Sol_e) denote the mTSP solution corresponding to the E-GTSP solution Sol_e.

Lemma 3. *For any solution Sol_e to the E-GTSP, there exists a corresponding mTSP solution mTSP(Sol_e) obtained via a valid reduction.*

Proof. Let Sol_e be a solution to the E-GTSP, represented as an ordered sequence of vertices $\{v_0^1, v_g^{j,o}, \ldots, v_0^i, \ldots, v_0^m, \ldots\}$, where each v_0^i is the initial position 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, Sol_e must contain exactly the m agent vertices. To construct mTSP(Sol_e), we identify all such agent vertices and partition Sol_e into m disjoint segments. For each agent i, let $sq(i)$ denote the sequence of goal vertices that appear between v_0^i and the next agent vertex in the solution. Each $sq(i)$ is assigned to agent i, preserving the order in which the goals appear. Since all vertices in $sq(i)$ belong to V'_g, we map each orientation-specific goal vertex $v_g^{j,o} \in sq(i)$ to its corresponding original goal vertex $v_g^j \in V_g$, discarding the orientation information. The result is a valid mTSP solution mTSP(Sol_e). □

Lemma 4. *The cost of mTSP(Sol_e) is equal to the cost of the original E-GTSP solution Sol_e.*

Proof. As described in Lemma 3, the mTSP solution is obtained by partitioning Sol_e into segments, each corresponding to the goals assigned to a specific agent. Segmentation is performed by traversing Sol_e and initiating a new segment whenever an initial agent vertex v_0^i is encountered. By construction, each such entry occurs via a Type 2 edge, which is auxiliary and has zero cost. Removing these edges to define the segments does not affect the total cost. Furthermore, mapping each orientation-specific goal vertex $v_g^{j,o}$ to its original vertex v_g^j does not alter any tour cost. Although orientation information is removed, the cost between original goal vertices in mTSP(Sol_e) is inherited from the traversal tours in Sol_e, which already account for orientation-dependent movement. Therefore, the total cost remains unchanged. □

Theorem 5. *Let Sol_e be an optimal solution to the E-GTSP. Then mTSP(Sol_e) is an optimal solution to the corresponding mTSP.*

Proof. By Lemma 3, there exists a valid reduction from Sol_e to an mTSP solution mTSP(Sol_e). By Lemma 4, this reduction preserves the total cost of the solution. Since Sol_e is optimal for the E-GTSP and the reduction does not alter feasibility or cost, it follows that mTSP(Sol_e) is also optimal for the corresponding mTSP instance. □