ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2025.2.025

Progressive Tree Expansion and Vertex-Based Pruning in UCT Strategies for the Stochastic Canadian Traveller Problem

Petr Šoustek[1,✉], Radomil Matoušek[1,✉], Sebastián Filip[2], Jiří Dvořák[3]

[1]Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic
[2]Mavenir Systems, Inc., Richardson, Texas, USA
[3]Independent Researcher in Mathematical Informatics, Brno, Czech Republic
xmsoust02@vut.cz✉, rmatousek@vut.cz✉; ORCID: 0000-0002-3142-0900

Abstract

The Stochastic Canadian Traveller Problem (SCTP) models adaptive routing on a known graph in which the availability of an edge is revealed only after the traveller reaches an incident vertex, while edge-specific blocking probabilities are known in advance. This paper investigates simulation-based strategies for minimizing the expected travel cost. After summarizing the Optimistic Policy, Hindsight Optimization, Optimistic Rollout, and established UCT-based approaches, the paper formalizes and evaluates two original heuristic variants of Optimistic UCT, UCTO2 and UCTP, which had previously been proposed and implemented in a master's thesis. UCTO2 replaces complete depth-oriented rollout trajectories with progressive tree expansion and batch rollout evaluation of selected nodes. UCTP augments UCTO2 with a pruning mechanism that retains the more promising of alternative tree nodes associated with the same graph vertex. The methods are evaluated on two families of randomly generated graphs with 50 vertices, using ten graphs per family and 500 stochastic realizations per graph. In the reported experiments, UCTP achieved lower mean travel costs than the Optimistic Policy and UCTO while requiring substantially less computation time than the other evaluated UCT variants. HOP and ORO nevertheless attained the lowest overall mean costs. The evidence therefore supports the complete UCTP configuration as a favourable finite-budget alternative within the UCT family, not as a generally superior SCTP policy.

Keywords: *Canadian Traveller Problem, stochastic routing, online path planning, belief-state search, Monte Carlo tree search, Upper Confidence Trees, vertex-based pruning.*

Received: 12 July 2025
Accepted: 24 July 2025
Online: 30 December 2025
Published: 30 December 2025

1 Introduction

The Canadian Traveller Problem (CTP) is an online path-planning problem in which an agent must travel from a source vertex to a target vertex in a known network while the traversability of some edges is initially unknown. The status of an uncertain edge is revealed only when the agent reaches one of its incident vertices. Papadimitriou and Yannakakis introduced the problem as a shortest-path problem with incomplete map information [11]. Since then, the CTP has become a standard model for adaptive navigation under uncertainty, including route planning in disrupted transport networks and navigation in partially known environments [7, 10].

The classical CTP is usually studied in an adversarial setting and evaluated by competitive analysis. The problem remains theoretically active: recent work established tight competitive results for unit-weighted outerplanar graphs and demonstrated that even structurally restricted CTP instances retain nontrivial online complexity [3]. In the stochastic CTP (SCTP), considered in this paper, every uncertain edge has a known blocking probability and the objective is to minimize expected travel cost. The realized edge states remain fixed during a run, while the agent's knowledge grows as new vertices are visited.

An SCTP decision cannot in general be based only on the current graph vertex. It must also reflect which edges are already known to be traversable, which are known to be blocked, and which remain unobserved. Papadimitriou and Yannakakis showed that deciding whether a strategy can guarantee a prescribed worst-case competitive ratio is PSPACE-complete and that optimizing the expected competitive ratio is #P-hard [11]. Fried et al. subsequently established PSPACE-completeness for stochastic CTP, including the independent-edge variant considered here [8]. Together with the exponential growth of the belief-state space, these results motivate simulation-based approximations that estimate candidate-action costs without explicitly enumerating a complete policy.

Eyerich et al. [6] compared several policies for the SCTP, including the optimistic policy (OMT), Hind-

sight Optimization (HOP), Optimistic Rollout (ORO), Blind UCT (UCTB), and Optimistic UCT (UCTO). UCT methods apply the exploration–exploitation principle of Upper Confidence Trees [9] to sampled belief-state trajectories. UCTO guides the otherwise uninformed UCT search by optimistic estimates and virtual prior rollouts. However, repeated depth-oriented simulations and the growth of alternative branches can still make the decision procedure computationally expensive.

Against this background, two original heuristic modifications of Optimistic UCT, UCTO2 and UCTP, initially proposed and implemented in [7], are formalized and evaluated. UCTO2 replaces the depth-oriented construction of a complete simulated route in each tree iteration by progressive tree expansion and rollout-based evaluation of selected nodes. UCTP augments UCTO2 with a pruning rule that retains the more promising of alternative tree nodes associated with the same graph vertex and transfers previously obtained estimates where possible. The journal contribution is not a claim of first introduction of these algorithms; rather, it provides a unified belief-state formulation, a more explicit algorithmic reconstruction, and a critical interpretation of the vertex-based pruning rule. Because equal graph vertices do not necessarily represent equal belief states, that rule is treated as heuristic state aggregation rather than an exact equivalence reduction.

The proposed strategies are evaluated on two families of randomly generated 50-vertex graphs, with ten graphs in each family and 500 stochastic realizations per graph. Within this experimental setting, UCTP achieves lower mean travel costs than OMT and UCTO and requires substantially less computation time than the other evaluated UCT variants. HOP and ORO nevertheless obtain the lowest aggregate mean costs. The results therefore support the complete UCTP configuration as a practical finite-budget alternative within the UCT family, while neither isolating the effect of pruning nor establishing general superiority. The conclusions remain conditional on the tested graph generators, parameter settings, and heuristic state aggregation.

The remainder of the paper is organized as follows. Section 2 formalizes the SCTP and delineates its relation to other CTP variants. Section 3 describes the baseline policies. Section 4 presents UCTO2 and UCTP. Section 5 reports the experimental methodology and results, and Section 6 concludes the paper.

2 Problem Formulation and Scope

2.1 Stochastic Canadian Traveller Problem

Definition 1 (SCTP instance)**.** An SCTP instance is a five-tuple

$$I = \langle G, p, c, s, t \rangle, \tag{1}$$

where $G = (V, E)$ is a finite connected undirected graph, $s, t \in V$ are the source and target vertices,

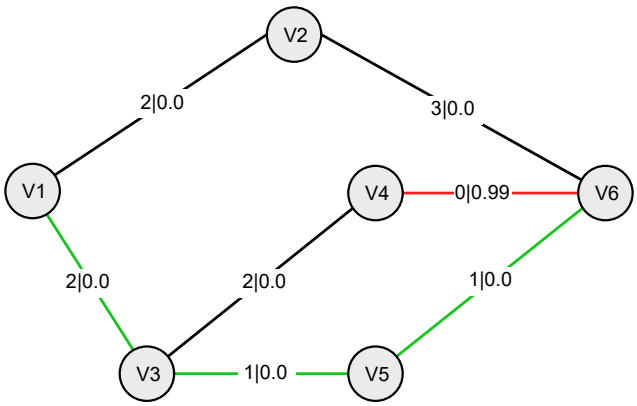

Figure 1: Illustrative SCTP instance and one realized weather. Each edge is labelled $c(e) \,|\, p(e)$. In the displayed realization, the red edge is blocked and the green edges form a shortest traversable route from $V1$ to $V6$. The agent does not know the complete realization in advance.

$c : E \to \mathbb{R}_{\geq 0}$ assigns a traversal cost to every edge, and $p : E \to [0, 1]$ assigns a blocking probability to every edge. Edges with $p(e) = 0$ are guaranteed to be traversable; an edge with $p(e) = 1$ can equivalently be removed from the graph.

Following the standard stochastic formulation [6], the state of each edge is sampled independently before a run and remains unchanged during that run. A *weather* is a subset $W \subseteq E$ containing exactly the traversable edges. Under the independence assumption,

$$\Pr(W) = \prod_{e \in W} \big(1 - p(e)\big) \prod_{e \in E \setminus W} p(e). \tag{2}$$

A weather W is called *good* if s and t are connected in the realized graph (V, W); otherwise it is *bad*. The edge realization is hidden from the agent at the beginning of the run. Whenever the agent reaches a vertex v, the states of all edges incident to v are revealed.

Figure 1 illustrates the distinction between fixed instance data and a realized weather. Edge labels contain the traversal cost and blocking probability, whereas the colouring depicts one complete realization and its clairvoyant shortest path. The full colouring is not available to the agent at the beginning of a run.

Definition 2 (Belief state and policy)**.** A belief state is represented by

$$b = \big(v, E^+, E^-, E^?\big), \tag{3}$$

where $v \in V$ is the current vertex and $(E^+, E^-, E^?)$ is a partition of E into edges known to be traversable, edges known to be blocked, and edges whose states are still unknown. A policy π maps every reachable nonterminal belief state to a movement along an edge known to be traversable.

Throughout Sections 3 and 4, the term *state* denotes the full belief state b unless a graph-vertex label, such

as v or $\ell(u)$, is explicitly stated. This convention separates the information state used by a policy from the vertex label used by the implemented pruning heuristic.

The agent's knowledge grows monotonically because edge states are static. Nevertheless, the number of possible belief states is bounded by

$$|V|\,3^{|E|}, \tag{4}$$

which explains the rapid growth of exact policy representations. Let $C(I, W, \pi)$ denote the total traversal cost incurred by policy π under weather W, and let $\mathcal{W}_g$ denote the set of good weathers. We assume $\Pr(W \in \mathcal{W}_g) > 0$, i.e., a traversable realization occurs with positive probability. As in [6], policy quality is evaluated conditionally on good weather, because every policy has infinite completion cost when no s–t path exists. The expected cost is

$$J(I, \pi) = \sum_{W \in \mathcal{W}_g} \Pr(W \mid W \in \mathcal{W}_g)\, C(I, W, \pi), \tag{5}$$

and an optimal policy satisfies

$$\pi^* \in \arg\min_{\pi} J(I, \pi). \tag{6}$$

Because $\mathcal{W}_g$ can contain exponentially many realizations, the value in (5) is typically estimated by sampling.

Equation (3) is also important for interpreting the proposed pruning strategy. Two search-tree nodes associated with the same graph vertex are not necessarily equivalent: they may encode different sets of observations and therefore admit different future decisions. Consequently, the UCTP rule described in Section 4 performs heuristic aggregation by graph vertex; it is not an exact merging of equivalent belief states.

2.2 Relation to Other CTP Variants

The original CTP also admits an adversarial interpretation, in which blocking probabilities are not used and a hidden set of blocked edges is selected against the online strategy. Performance is then commonly assessed by the competitive ratio between the distance travelled online and the shortest path available to an agent with complete information [2, 11]. In the k-CTP, at most k edges may be blocked [2, 15, 16]. Recent structural results for this model include tight bounds on unit-weighted outerplanar graphs [3].

Recoverable variants allow a blocked edge to be reopened after paying a specified recovery cost or waiting time [13, 14]. Bar-Noy and Schieber also formulated the stochastic recoverable CTP (SRCTP), which combines edge-state probabilities with recovery costs [2]. These models differ fundamentally from the present setting, where a blocked edge remains unavailable throughout the run.

A separate modelling axis concerns dependence among edge states. Dependent SCTP permits correlated blockages represented by a joint distribution, whereas the model in Definition 1 assumes independent Bernoulli edge states. Fried et al. analysed both dependent and independent stochastic variants and proved PSPACE-completeness for stochastic CTP [8].

Further extensions include repeated traversal by successive agents [4], communication among multiple travellers [18], remote sensing of nonincident edges [5], obstacle neutralization [1, 17], and multi-agent online O–D routing with limited communication [12].

This paper is restricted to the single-agent, single-source/single-target, nonrecoverable SCTP with independent, static edge states and local observation at incident vertices. The proposed methods therefore optimize expected traversal cost under the model defined by (1)–(5); they do not directly address recovery actions, correlated edge states, remote sensing, or inter-agent communication.

3 Baseline Policies for the SCTP

This section defines the baseline policies used to motivate and evaluate the proposed UCT modifications. The presentation follows Eyerich et al. [6] and uses the belief-state notation introduced in Section 2. The essential distinction among the methods is the information assumed during cost estimation: OMT ignores blocking probabilities, HOP averages clairvoyant shortest-path costs, ORO simulates OMT without clairvoyance, and UCT learns a tree policy from mutually dependent rollouts.

3.1 Common Decision Framework

Let $b = (v, E^+, E^-, E^?)$ be a nonterminal belief state. A policy π is represented by a cost-to-go estimate $\widehat{C}_\pi(b)$. For an admissible movement a that incurs immediate cost $c(a)$ and leads to a successor belief state b_a, the corresponding one-step estimate is

$$Q_\pi(b, a) = c(a) + \widehat{C}_\pi(b_a). \tag{7}$$

The policy selects an action from

$$\pi(b) \in \arg\min_{a \in A(b)} Q_\pi(b, a), \tag{8}$$

where $A(b)$ contains the currently feasible movements.

Eyerich et al. [6] formulate a successor more generally as a belief state reached by a shortest movement sequence in the known traversable subgraph, ending either at the target or at a vertex where new edge information is obtained. Equation (8) also covers an edge-by-edge implementation, because intermediate movements that reveal no new information do not require a new stochastic decision.

For HOP and ORO, a separate estimate is obtained for every successor under consideration. Thus, N rollouts are performed per evaluated successor. UCT instead performs N total rollouts from the current belief state and obtains estimates for its successors as a by-product of one shared search tree.

3.2 Optimistic Policy

The optimistic policy (OMT) applies the free-space assumption: every edge that is not known to be blocked is temporarily treated as traversable [6]. For belief state b, define the optimistic graph

$$G_{\mathrm{OMT}}(b) = \left(V, E^{+} \cup E^{?}\right). \tag{9}$$

The optimistic cost estimate is the shortest-path distance

$$C_{\mathrm{OMT}}(b) = d_{G_{\mathrm{OMT}}(b)}(v, t). \tag{10}$$

In the implementation considered by Eyerich et al., the distance is computed by Dijkstra's algorithm. The agent follows an optimistic shortest path until it reaches the target or discovers that an intended edge is blocked. In the latter case, the belief state is updated and a new optimistic shortest path is computed.

OMT is deterministic and computationally inexpensive, but it does not use the probabilities $p(e)$. A path remains attractive even when it contains edges with high blocking probabilities. OMT is therefore used as a baseline rather than as a stochastic optimizer.

3.3 Hindsight Optimization

Hindsight Optimization (HOP) estimates a belief-state cost by sampling complete edge realizations consistent with the current knowledge [6]. Let $W_1, \ldots, W_N$ be independently sampled weathers consistent with b, and let

$$\mathcal{S}_b = \{n \in \{1, \ldots, N\} : W_n \text{ admits a path from } v \text{ to } t\} \tag{11}$$

be the indices of successful samples. For each $n \in \mathcal{S}_b$, HOP computes the shortest-path distance in the fully revealed traversable graph (V, W_n). Its estimate is

$$\widehat{C}^{N}_{\mathrm{HOP}}(b) = \frac{1}{|\mathcal{S}_b|} \sum_{n \in \mathcal{S}_b} d_{(V, W_n)}(v, t). \tag{12}$$

If no rollout is successful, the estimate is treated as infinite.

A HOP rollout is therefore not a simulation of the agent's partial-information behaviour. It is a weather sample followed by a clairvoyant shortest-path computation. HOP is sometimes described as *averaging over clairvoyance*: a different optimal route may be selected for every sampled weather. This makes HOP less optimistic than OMT, but it can still underestimate the cost achievable by any single adaptive policy. Increasing N stabilizes the Monte Carlo estimate, yet it does not remove this structural bias.

3.4 Optimistic Rollout

Optimistic Rollout (ORO) uses the same independent weather sampling and the same averaging over successful samples as HOP, but changes the cost assigned to each sample [6]. For a successful weather W_n, ORO simulates an actual OMT run. The simulated agent only uses information that would be observable at its current vertex. It follows an optimistic shortest path until reaching the target or encountering an edge that is blocked in W_n; after a blockage is observed, the optimistic graph is updated and the shortest path is recomputed. Let $C_{\mathrm{OMT}}(b; W_n)$ denote the resulting travelled distance. Then

$$\widehat{C}^{N}_{\mathrm{ORO}}(b) = \frac{1}{|\mathcal{S}_b|} \sum_{n \in \mathcal{S}_b} C_{\mathrm{OMT}}(b; W_n). \tag{13}$$

Unlike HOP, ORO does not grant the simulated agent advance knowledge of the sampled weather. Its rollouts are genuine partial-information runs, but every run follows the fixed OMT policy. ORO can consequently inherit and amplify weaknesses of OMT: an action may receive a high estimate because OMT behaves poorly after that action, even when a better continuation policy exists. The N ORO rollouts are mutually independent.

3.5 UCT-Based Policies

UCT [9] builds a search tree over belief sequences. Let

$$\sigma = \langle b_0, b_1, \ldots, b_i \rangle \tag{14}$$

be a partial rollout beginning at the queried belief state $b_0 = b$. After the first k rollouts, let $R_k(\sigma)$ be the number of rollouts whose belief sequence starts with σ, and let $\overline{C}_k(\sigma)$ be their average residual cost from the final belief state of σ to the target.

Suppose that an unfinished sequence ρ ends in belief state b_i and that $\rho_j = \langle \rho, b'_j \rangle$ is obtained by appending successor b'_j. UCT selects the successor maximizing

$$U_k(\rho_j) = B_k \sqrt{\frac{\log R_k(\rho)}{R_k(\rho_j)}} - c(\rho, \rho_j) - \overline{C}_k(\rho_j), \tag{15}$$

where $c(\rho, \rho_j)$ is the movement cost required to extend ρ to ρ_j, and $B_k > 0$ controls the exploration–exploitation balance. Because the problem is formulated as cost minimization, promising successors have a small movement cost and a small estimated residual cost, while the positive square-root term favours less frequently sampled successors. If $R_k(\rho_j) = 0$, the value in (15) is defined as $+\infty$, ensuring that every available successor is tried before repeated exploitation.

Each rollout samples a weather consistent with the root belief state and then generates a complete partial-information run. Unlike ORO, later rollout choices depend on statistics collected during earlier rollouts. After a rollout reaches the target, the residual cost observed after every visited prefix is added to the corresponding tree statistics. Failed rollouts are excluded from the average, consistently with the good-weather conditioning in Section 2. When the rollout or time budget is exhausted, the successor of the root with the smallest estimated movement-plus-residual cost is selected.

Figure 2 gives a schematic view of the first three rollout iterations and the corresponding tree growth in the archival implementation. Its main visual value is that it makes the early exploration of sibling alternatives explicit: after the first rollout through vertex 2, the second iteration opens the alternative through vertex 3 before deeper refinement continues. The symbols R^k and C^k_{sum} denote the visit count and cumulative backed-up rollout return, respectively. The diagram follows a full-return backup convention: the complete root-to-target rollout estimate is added to every visited node. The residual-cost notation introduced above is mathematically different, but for children of a common parent it induces the same ranking as long as the immediate movement cost is not counted twice; the two numerical conventions should not be compared across different tree depths. In the third rollout, the annotation "Cannot add state → estimate calculated with GS" marks a prefix that cannot be extended under the no-repeated-state rule. The remaining cost is therefore estimated by the archived Greedy Strategy (GS), which is the optimistic free-space policy denoted OMT in this paper. Node numbers are graph-vertex labels used for visual clarity; the formal decision state remains the belief state defined in (3).

The bias parameter should scale with the magnitude of the problem costs. Eyerich et al. estimate B_k for rollout $k+1$ by the average cost of the preceding k rollouts; its value is irrelevant for the first rollout. The parameter settings actually used in the present experiments are reported separately in Section 5.

Blind UCT (UCTB). UCTB uses (15) without any CTP-specific guidance. In particular, initially unvisited successors are not ordered by a goal-directed heuristic. UCTB is asymptotically well founded, but its early rollouts can resemble random walks and may require a prohibitively large sample budget before useful estimates emerge [6].

Optimistic UCT (UCTO). UCTO supplements UCTB with the optimistic cost in (10) [6]. It introduces two forms of guidance:

1. if several successors of a partial rollout are unvisited, ties are broken in favour of the successor with the smaller C_{OMT} value;
2. every successor is initialized as if it had already received M virtual rollouts, each with residual cost C_{OMT}.

If $S_k(\rho_j)$ denotes the sum of residual costs from actual rollouts through ρ_j, the virtual-rollout initialization can be written as

$$\begin{aligned} R_k^{(M)}(\rho_j) &= R_k(\rho_j) + M, \\ \overline{C}_k^{(M)}(\rho_j) &= \frac{S_k(\rho_j) + MC_{\text{OMT}}(b'_j)}{R_k(\rho_j) + M}. \end{aligned} \tag{16}$$

The modified quantities replace $R_k(\rho_j)$ and $\overline{C}_k(\rho_j)$ in the selection rule. The original experiments used $M = 20$ and divided the adaptive bias estimate by ten to favour exploitation [6]. The virtual prior influences early search but becomes negligible as the number of actual visits increases; therefore, it does not alter the limiting UCT policy.

3.6 Asymptotic Distinctions

The limiting cost functions established by Eyerich et al. clarify why the algorithms should not be treated as interchangeable. For every SCTP instance and belief state,

$$C_{\text{OMT}}(b) \le C^{\infty}_{\text{HOP}}(b) \le C^{\infty}_{\text{UCT}}(b) = C^*(b) \le C^{\infty}_{\text{ORO}}(b), \tag{17}$$

where the UCT result applies to both UCTB and UCTO [6]. These inequalities concern limiting *cost estimates*, not finite-budget empirical policy rankings. OMT and HOP can remain overly optimistic even with unlimited sampling, whereas ORO may remain overly pessimistic because its continuation policy is fixed to OMT. In contrast, UCT converges to the optimal cost function under the assumptions of the reference analysis, although UCTB can converge too slowly to be useful in practice. UCTO preserves the same limit while using OMT-based priors to improve finite-budget search.

This distinction motivates the methods in Section 4: UCTO2 and UCTP retain optimistic guidance but modify how the search tree is expanded and reduced in order to obtain useful decisions within a limited computational budget.

4 Proposed UCT Variants

UCTO2 and UCTP are the two original strategy designs first proposed and implemented in [7]. The present section reconstructs them in a unified notation, makes their finite-budget mechanisms explicit, and distinguishes the belief state of the decision process from the graph-vertex labels used by the pruning rule. Both strategies modify the finite-budget behaviour of UCTO and are executed anew whenever the agent must choose its next movement in the realized SCTP instance. The root represents the agent's current decision state, while a tree node u stores a graph-vertex label $\ell(u) \in V$, its parent, an evaluation flag, a visit count $R(u)$, and a cumulative propagated cost $S(u)$. Its empirical mean is

$$\overline{C}(u) = \frac{S(u)}{R(u)} \tag{18}$$

whenever $R(u) > 0$. In contrast to the belief-state tree described in Section 3, the implemented variants identify and compare nodes primarily through graph vertices and tree paths. This distinction is essential for the interpretation of UCTP.

The original work refers to this construction as breadth-first UCT. It is not a conventional FIFO breadth-first search: unevaluated siblings are considered early, but repeated UCT selection gradually expands promising branches to greater depths. A deci-

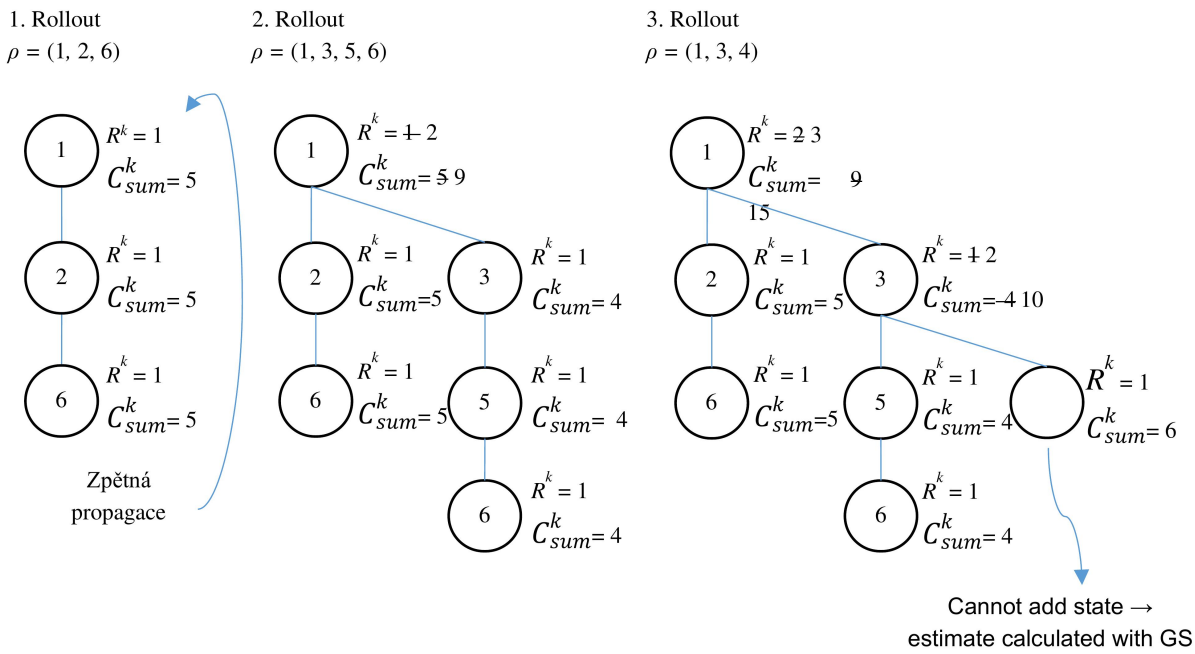

Figure 2: Growth and update of UCT statistics during the first three rollout iterations for the example in Fig. 1. After the first rollout explores $1 \to 2 \to 6$, the next iteration opens the sibling branch through vertex 3 before deeper refinement. The quantities R^k and C^k_{sum} use the archival full-return convention. In the third rollout, no additional non-repeated state can be appended after vertex 4, so the remaining cost is estimated by GS (OMT in the terminology of this paper). Adapted from [7].

sion process is stopped either after N tree iterations or when a prescribed decision-time limit t_{dec} is reached. Each newly selected node is evaluated by N_r stochastic rollouts.

4.1 UCTO2

UCTO2 retains the optimistic guidance of UCTO but changes the unit of simulation. Standard UCTO constructs one complete simulated route in each rollout. UCTO2 instead performs one tree iteration at a time, evaluates a selected node by a batch of N_r rollouts, and reuses that estimate in subsequent tree traversal. One decision consists of selection, expansion, rollout-based evaluation, and backward propagation.

Selection

Starting at the root, the algorithm repeatedly selects an evaluated child until it reaches an unevaluated node, a leaf, or a node labelled by the target vertex. For an evaluated child u of node q, the selection score is

$$U_i(u) = B_i \sqrt{\frac{\log \widetilde{R}_i(q)}{\widetilde{R}_i(u)}} - \widetilde{C}_i(u), \tag{19}$$

and the child with the largest score is selected. The quantities $\widetilde{R}_i$ and $\widetilde{C}_i$ include the M optimistic virtual rollouts defined in (16). An unevaluated child is selected before already evaluated alternatives; when several unevaluated children are available, the optimistic estimate C_{OMT} is used as the tie-breaking rule.

The exploration bias is scaled by the average root cost obtained in the preceding iterations:

$$B_i = \alpha\, \overline{C}_{i-1}(\text{root}). \tag{20}$$

The implemented UCTO2 and UCTP configurations use

$$\alpha = 10. \tag{21}$$

For the first iteration, the value of B_i is immaterial because an unevaluated alternative is selected. The larger multiplier strengthens exploration relative to the reference UCTO setting. The experimental UCTO comparator uses a different multiplier, reported explicitly in Section 5.

Expansion

If the selected node has already been evaluated and is not terminal, all admissible graph neighbours are inserted as children and one new child is selected for evaluation. Immediate return to the predecessor vertex is normally suppressed because it would reproduce a recently examined branch. The predecessor is reconsidered after the agent observes a new blockage, since the newly acquired information can make backtracking rational.

The implementation also retains a cached estimate for the predecessor vertex when that alternative received at least ten visits in the preceding decision process. The cached value corresponds to the second-best root alternative of that process. The threshold of ten

visits is empirical and was introduced to reduce repeated computation; it is not part of the UCT convergence theory.

Rollout-Based Node Evaluation

Let u be the selected node, let $q = \mathrm{par}(u)$, and let $e_u = (\ell(q), \ell(u))$ be the proposed edge. The algorithm generates N_r independent weathers consistent with the information currently available to the agent. For rollout n, define

$$Y_n(u) = \begin{cases} c(e_u) + d_{(V,W_n)}(\ell(u), t), & e_u \in W_n, \\ C_{\mathrm{OMT}}(q; W_n), & e_u \notin W_n. \end{cases} \tag{22}$$

In the first case, the proposed edge is traversable and the remaining distance is evaluated optimistically by the shortest path in the sampled realization. In the second case, the proposed movement cannot be executed and an OMT run is simulated from the parent node under the same weather.

A rollout is successful if the corresponding continuation reaches the target. Let $\mathcal{A}(u) \subseteq \{1, \ldots, N_r\}$ contain the successful rollout indices. The one-time estimate assigned to u is

$$\widehat{C}_r(u) = \frac{1}{|\mathcal{A}(u)|} \sum_{n \in \mathcal{A}(u)} Y_n(u). \tag{23}$$

If N_t and N_b denote the numbers of retained rollouts in which e_u is traversable and blocked, respectively, then

$$|\mathcal{A}(u)| = N_t + N_b \leq N_r. \tag{24}$$

Samples without a successful continuation are omitted, consistently with the good-weather conditioning adopted in Section 2. If $\mathcal{A}(u)$ is empty, the node receives an infinite estimate. The estimator in (23) is deliberately asymmetric: a traversable proposed edge is followed by a clairvoyant shortest-path estimate, whereas a blocked proposed edge triggers a partial-information OMT simulation from the parent.

Backward Propagation

Suppose that the selected node is reached through the tree path

$$P = \langle u_0, u_1, \ldots, u_d \rangle, \tag{25}$$

where u_0 is the root and $u_d = u$. The propagated values are defined by

$$Z_d = \widehat{C}_r(u_d), \tag{26}$$

$$Z_j = Z_{j+1} + c(\ell(u_{j-1}), \ell(u_j)), \quad j = d-1, \ldots, 1, \tag{27}$$

$$Z_0 = Z_1. \tag{28}$$

Thus, the selected-node estimate already contains the proposed final tree edge; when moving towards the root, the cost of the edge entering each ancestor is added. The root receives the same propagated value as its direct child. For every node u_j on P, the statistics are updated as

$$S(u_j) \leftarrow S(u_j) + Z_j, \qquad R(u_j) \leftarrow R(u_j) + 1. \tag{29}$$

After the iteration or time budget has been exhausted, the root child with the smallest estimated movement-plus-completion cost is selected as the agent's next action.

Algorithm 1 UCTO2 decision procedure

Require: current decision state, budgets N or t_{dec}, rollout batch N_r, virtual prior M

```
initialize a root node for the current position
while the iteration and time budgets are not exhausted do
    u ← root
    while u is evaluated, nonterminal, and has children do
        u ← child maximizing (19)
    end while
    if u is evaluated and nonterminal then
        expand u and select a new child
    end if
    if u is nonterminal then
        estimate Ĉ_r(u) by (22)–(23)
        propagate the estimate by (26)–(29)
    end if
end while
return the best estimated child of the root
```

Algorithm 1 summarizes one UCTO2 decision process.

4.2 UCTP

UCTP applies the UCTO2 decision procedure and adds pruning by graph-vertex label. Let $\mathcal{M}$ be a registry of evaluated nonterminal nodes. For each graph vertex v, the registry stores at most one evaluated representative u satisfying $\ell(u) = v$, together with its parent and rollout estimate. Target-labelled nodes are exempt from pruning.

Suppose that a newly evaluated node u and an existing representative u' satisfy

$$\ell(u) = \ell(u') = v, \qquad u \neq u'. \tag{30}$$

The two occurrences can have different parents, different root-to-node paths, and different estimates because the evaluation in (23) depends on the proposed parent–child edge. Let

$$P(u) = \langle u_1, \ldots, u_d = u \rangle \tag{31}$$

denote the path from a direct child of the root to u, excluding the root itself. UCTP assigns the path score

$$H(u) = \frac{1}{d} \sum_{j=1}^{d} \widehat{C}_r(u_j). \tag{32}$$

The occurrence with the smaller path score is retained:

$$u_{\text{keep}} \in \arg\min_{x \in \{u,u'\}} H(x), \tag{33}$$

and the other occurrence is removed from the tree.

If the removed occurrence already has evaluated descendants, they are transferred to the retained representative. Their accumulated information is then propagated through the retained branch to the root so that completed rollout work is not discarded. The registry $\mathcal{M}$ is updated to point to the retained node. Consequently, the maintained tree contains at most one evaluated occurrence of each nonterminal graph-vertex label, although unevaluated duplicate frontier nodes can arise temporarily.

The pruning rule is a heuristic state-aggregation operation. Equality of vertex labels does not imply equality of the belief states in (3), and the path average in (32) is not a Bellman value or an admissible bound. UCTP can therefore remove a branch that would be preferable under a different observation history. No optimality or asymptotic-convergence claim is made for the pruned algorithm. Its intended role is computational: by concentrating evaluations on one representative of a repeatedly encountered vertex, it seeks a more favourable finite-time cost–quality trade-off. The experiments evaluate the complete UCTP configuration, including the modified expansion scheme, the multiplier $\alpha = 10$, and pruning; they do not isolate the causal contribution of pruning alone.

5 Experimental Methodology and Results

This section reconstructs and critically reanalyses the experimental design and numerical results documented in [7]. They compare the proposed UCT variants with OMT, HOP, ORO, and the reference UCTO configuration. A clairvoyant Dijkstra solution is included only as an offline lower bound: it is computed after the complete edge realization is known and is not an implementable SCTP policy. The archived implementation and thesis text sometimes refer to the deterministic free-space baseline as the Greedy Strategy (GS). Because this policy applies the same optimistic assumption as OMT in [6], its archived results are reported here under the OMT terminology for consistency.

5.1 Experimental Design

Graph Families

Two families of graphs, denoted A and B, were generated. Every graph contains 50 vertices. Ten graph instances were generated from each family.

For family A, the generator selected between 2 and 10 neighbours for each vertex. Seventy-five percent of the edges were assigned blocking probabilities in the interval $[0.1, 1]$, while the remaining 25% were guaranteed to be traversable. Edge costs were integer values in $[10, 50]$.

Table 1: Documented strategy configurations.

Strategy	Search budget	Additional parameters
OMT	deterministic	–
HOP	$N = 1000$	–
ORO	$N = 300$	–
UCTO	$t_{\text{dec}} = 10$ s	$M = 20$, $\alpha = 5$
UCTO2	$t_{\text{dec}} = 10$ s	$M = 20$, $N_r = 20$, $\alpha = 10$
UCTP	$t_{\text{dec}} = 10$ s	$M = 20$, $N_r = 20$, $\alpha = 10$

For family B, the corresponding neighbour range was 2 to 4. Seventy-five percent of the edges were assigned blocking probabilities in $[0.1, 0.9]$, and the remaining 25% had zero blocking probability. Every vertex was assigned integer coordinates in $[0, 99]^2$, and the cost of an edge was the Euclidean distance between its endpoints.

Family A is therefore denser and generally permits routes with fewer intermediate vertices, whereas family B is sparser and uses geometry-derived costs. The archived experiment description does not specify the random seeds, the exact edge counts of the generated graphs, or further details of the graph-construction procedure. The results should consequently be interpreted as evidence for these 20 fixed test instances rather than as estimates over a fully specified random-graph distribution.

Strategies and Parameter Settings

Each strategy was executed 500 times on every graph, with newly realized edge states in each run. Table 1 summarizes the documented parameter settings. HOP and ORO used different rollout counts because preliminary calibration indicated that $N_{\text{HOP}} = 1000$ and $N_{\text{ORO}} = 300$ produced broadly similar computation times while remaining in a region of stable mean solution quality [7]. The three UCT-based configurations used a decision-time limit of

$$t_{\text{dec}} = 10 \text{ s} \tag{34}$$

for every stochastic decision. On the test platform, this allowed UCTO to perform approximately 10 000 rollouts per decision.

For UCTO, the exploration bias in (20) was therefore

$$B_i^{\text{UCTO}} = 5\,\overline{C}_{i-1}(\text{root}), \tag{35}$$

whereas UCTO2 and UCTP used the multiplier $\alpha = 10$ given in (21). The comparison consequently concerns complete algorithm configurations. In particular, it does not isolate pruning from the modified expansion scheme, the node estimator, or the different exploration-bias multiplier.

The experiments were executed by the documented Java implementation on a Lenovo IdeaPad G510 equipped with an Intel Core i5-4210M processor at 2.6 GHz and 4 GB of DDR3L-1600 memory. The reported time is the mean wall-clock computation time for one completed source-to-target run, including all decisions

made during that run. It is therefore not identical to the per-decision limit in (34). OMT times were recorded as zero at the available reporting precision.

Reported Statistics

For every graph and strategy, Tables 2 and 3 report the mean path cost over 500 runs together with the reported half-width of a 95% confidence interval. The final cost row is the unweighted arithmetic mean of the ten graph-level means. No confidence interval was reported for this aggregate. The exact confidence-interval estimator and the original per-run samples are not available in the preserved research record; therefore, no post hoc paired tests or claims of statistical significance are made here. Comparisons below refer only to differences between the reported means.

5.2 Results for Graph Family A

For family A, HOP had the lowest reported mean on seven graphs, ORO on two graphs, and UCTO on one graph. Their aggregate means were 189.2 for HOP and 189.5 for ORO, corresponding to reductions of 13.3% and 13.1%, respectively, relative to OMT. UCTP obtained an aggregate mean of 192.2. It was therefore not the best policy overall, but it was the best UCT-based configuration in aggregate and reduced mean cost by 11.9% relative to OMT, by 2.8% relative to UCTO, and by 6.2% relative to UCTO2.

UCTP also had the lowest mean among the three UCT configurations on five of the ten family-A graphs. Its mean computation time of 23.2 s was approximately 3.1× lower than that of UCTO and 3.2× lower than that of UCTO2. HOP and ORO were nevertheless both faster and slightly better in aggregate under the selected configurations.

5.3 Results for Graph Family B

For family B, ORO had the lowest reported mean on seven graphs and HOP on three. ORO achieved the best aggregate mean, 601.2, which is 11.2% below the OMT mean. HOP followed closely at 603.9, a reduction of 10.8% relative to OMT. UCTP again had the best aggregate result among the UCT configurations: its mean cost of 621.1 was 8.2% below OMT, 6.7% below UCTO, and 5.7% below UCTO2.

The advantage of UCTP within the UCT group was more consistent for the sparse graph family: it had the lowest UCT mean on nine of the ten graphs. Its mean computation time was approximately 5.6× lower than UCTO and 6.1× lower than UCTO2. The implementation can terminate a decision process early when pruning leaves only one child of the root; the stronger time reduction in family B is consistent with this topology-dependent mechanism. This observation is descriptive rather than an isolated causal estimate of the pruning effect.

5.4 Interpretation and Limitations

Three conclusions are supported by the archived results. First, methods that use blocking probabilities improved substantially over OMT in aggregate, although the magnitude depended on the graph family and algorithm configuration. Second, HOP and ORO produced the lowest overall mean costs and also the shortest nontrivial computation times under the selected budgets. The random graph families may not emphasize structures in which the limiting optimism of HOP or the fixed OMT continuation of ORO becomes especially harmful. Third, UCTP improved the empirical cost–time trade-off within the UCT family: it reduced both aggregate mean cost and computation time relative to the implemented UCTO and UCTO2 configurations.

These conclusions have important limits. The experiment was not an ablation study, because UCTP differs from UCTO in several mechanisms in addition to pruning. The rollout budgets of HOP and ORO are not computationally matched to the per-decision time budgets of the UCT strategies. Only ten graphs from each incompletely specified generator were evaluated, and no structured “trap” instances or exact small-instance policies were included. Finally, the preserved archive contains graph instances and aggregate tables but not the Java source code, random seeds, raw per-run outputs, or the exact confidence-interval computation. The present analysis therefore retains the original numerical evidence, corrects its interpretation, and avoids claims of general superiority or statistical significance.

6 Conclusion

This paper presented a rigorous journal-level reconstruction of two original heuristic UCT variants for the Stochastic Canadian Traveller Problem, UCTO2 and UCTP, first proposed in [7]. The SCTP was formulated over belief states that distinguish the current vertex from the accumulated knowledge of traversable, blocked, and unobserved edges. Within this framework, OMT, HOP, ORO, UCTB, and UCTO were differentiated according to the information and continuation policy used in their cost estimates. UCTO2 was then formalized as progressive tree expansion with batch rollout evaluation, and UCTP as its vertex-labelled pruning extension.

The archived experiments do not support a claim that UCTP is the best SCTP policy overall. For graph family A, HOP achieved the lowest aggregate mean cost, 189.2, followed by ORO at 189.5 and UCTP at 192.2. For family B, ORO was best at 601.2, followed by HOP at 603.9 and UCTP at 621.1. UCTP was, however, the best evaluated UCT configuration in aggregate for both families. Relative to UCTO, it reduced mean cost by 2.8% in family A and 6.7% in family B, while its mean computation time was approximately 3.1× and 5.6× lower, respectively. The principal em-

Table 2: Family A: mean path cost ± reported 95% confidence-interval half-width. Bold type marks the lowest mean among implementable policies for each graph. Dijkstra is a clairvoyant offline lower bound.

Graph	Dijkstra	OMT	HOP	ORO	UCTO	UCTO2	UCTP
A50-0	176.3 ± 2	276.7 ± 8	256.1 ± 10	$\mathbf{220.3 \pm 5}$	238.9 ± 5	243.6 ± 7	239.8 ± 9
A50-1	151.7 ± 2	201.8 ± 6	$\mathbf{161.6 \pm 4}$	181.5 ± 5	181.2 ± 5	166.4 ± 4	169.5 ± 4
A50-2	116.9 ± 2	161.8 ± 6	$\mathbf{149.5 \pm 6}$	150.2 ± 5	155.0 ± 6	161.9 ± 7	157.6 ± 6
A50-3	110.1 ± 1	150.3 ± 4	130.7 ± 2	135.8 ± 3	$\mathbf{124.2 \pm 3}$	127.9 ± 3	132.5 ± 3
A50-4	150.8 ± 2	194.5 ± 7	$\mathbf{160.3 \pm 3}$	161.0 ± 3	179.1 ± 5	194.0 ± 6	161.6 ± 3
A50-5	200.7 ± 3	279.0 ± 9	243.4 ± 5	$\mathbf{241.9 \pm 6}$	251.0 ± 5	256.5 ± 6	254.9 ± 5
A50-6	176.2 ± 2	229.8 ± 4	$\mathbf{186.0 \pm 2}$	198.8 ± 3	217.6 ± 4	228.5 ± 4	187.0 ± 2
A50-7	175.4 ± 2	225.4 ± 5	$\mathbf{196.4 \pm 4}$	196.5 ± 4	200.4 ± 5	208.5 ± 5	197.2 ± 4
A50-8	192.6 ± 3	262.9 ± 6	$\mathbf{224.6 \pm 4}$	225.1 ± 4	235.5 ± 5	261.5 ± 5	228.5 ± 5
A50-9	165.6 ± 1	198.7 ± 4	$\mathbf{183.5 \pm 3}$	183.6 ± 3	194.2 ± 4	199.8 ± 3	193.6 ± 3
Mean cost	161.6	218.1	**189.2**	189.5	197.7	204.9	192.2
Mean time [s]	0.0	0.0	5.7	8.0	72.3	75.1	23.2

Table 3: Family B: mean path cost ± reported 95% confidence-interval half-width. Bold type marks the lowest mean among implementable policies for each graph. Dijkstra is a clairvoyant offline lower bound.

Graph	Dijkstra	OMT	HOP	ORO	UCTO	UCTO2	UCTP
B50-0	477.1 ± 6	597.9 ± 12	$\mathbf{550.2 \pm 9}$	559.2 ± 10	595.9 ± 11	604.4 ± 12	565.1 ± 9
B50-1	607.3 ± 6	834.7 ± 15	727.3 ± 12	$\mathbf{706.2 \pm 11}$	812.3 ± 17	754.8 ± 14	746.3 ± 12
B50-2	606.5 ± 6	847.7 ± 19	728.7 ± 12	$\mathbf{712.6 \pm 12}$	775.2 ± 15	812.3 ± 16	744.0 ± 12
B50-3	498.0 ± 6	663.2 ± 14	$\mathbf{568.9 \pm 11}$	579.8 ± 10	756.4 ± 17	648.9 ± 12	584.2 ± 11
B50-4	430.5 ± 4	522.9 ± 12	$\mathbf{477.9 \pm 10}$	492.8 ± 8	491.8 ± 10	508.4 ± 13	484.4 ± 9
B50-5	533.0 ± 5	690.8 ± 14	625.4 ± 11	$\mathbf{619.6 \pm 10}$	670.7 ± 11	646.7 ± 12	660.3 ± 13
B50-6	467.9 ± 6	606.7 ± 15	553.8 ± 12	$\mathbf{548.5 \pm 12}$	604.4 ± 13	642.1 ± 25	585.5 ± 14
B50-7	535.2 ± 7	700.0 ± 15	616.7 ± 13	$\mathbf{613.5 \pm 13}$	676.0 ± 15	667.4 ± 14	647.8 ± 14
B50-8	531.7 ± 6	722.2 ± 16	647.7 ± 10	$\mathbf{643.7 \pm 12}$	671.4 ± 12	678.8 ± 13	654.8 ± 11
B50-9	472.3 ± 4	580.7 ± 10	542.2 ± 8	$\mathbf{536.5 \pm 8}$	604.1 ± 9	620.5 ± 10	538.9 ± 7
Mean cost	516.0	676.7	603.9	**601.2**	665.8	658.4	621.1
Mean time [s]	0.0	0.0	6.8	7.9	108.1	117.3	19.2

pirical conclusion is therefore a more favourable finite-budget cost–time trade-off within the tested UCT family.

This conclusion is conditional. UCTP differs from UCTO not only by pruning, but also by its expansion scheme, rollout-based node estimator, and exploration-bias multiplier. The experiment used different budget definitions for HOP, ORO, and the UCT methods, involved only ten graphs per family, and did not preserve random seeds, raw run-level data, or the exact confidence-interval procedure. Moreover, merging alternatives by graph-vertex label is not equivalent to merging identical belief states and can discard information-dependent branches. No optimality, convergence, isolated pruning effect, or general superiority is therefore claimed.

A definitive follow-up should use an openly available implementation, matched computational budgets, controlled ablations of expansion, bias scaling, and pruning, and paired evaluations on shared stochastic realizations. Small instances with exact belief-state solutions and structured instances designed to expose optimistic or vertex-aggregation failures would provide particularly informative tests. A further methodological direction is to replace vertex-only pruning by a belief-aware or bounded-loss aggregation rule, thereby retaining the computational motivation of UCTP while providing a stronger theoretical basis.

Acknowledgement: This work was supported by the Czech Science Foundation (GACR), grant project No. 24-12474S, "Benchmarking derivative-free global optimization methods," and by grant No. FEKT-S-26-8988, "Advanced Methods in Cybernetics, Robotics, and Artificial Intelligence," financially supported by the Internal Science Fund of Brno University of Technology.

References

[1] Alkaya, A. F., Yildirim, S., and Aksakalli, V. Heuristics for the canadian traveler problem with neutralizations. *Computers & Industrial Engineering 159* (2021), 107488.

[2] Bar-Noy, A., and Schieber, B. The canadian traveller problem. In *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms* (1991), Citeseer, pp. 261–270.

[3] Beaudou, L., Bergé, P., Chernyshev, V., Dailly, A., Gérard, Y., Lagoutte, A., Limouzy, V., and Pastor, L. The canadian traveller problem on outerplanar graphs. In *49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024)* (2024), vol. 306 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 19:1–19:16.

[4] Bnaya, Z., Felner, A., Fried, D., Maksin, O., and Shimony, S. E. Repeated-task canadian traveler problem. *AI Communications 28*, 3 (2015), 453–477.

[5] Bnaya, Z., Felner, A., and Shimony, S. E. Canadian traveler problem with remote sensing. In *IJCAI* (2009), pp. 437–442.

[6] Eyerich, P., Keller, T., and Helmert, M. High-quality policies for the Canadian Traveler's Problem. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (2010), pp. 51–58.

[7] Filip, S. Řešení problému kanadského cestujícího [Solving Canadian Traveller Problem]. Master's thesis, Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Brno, Czech Republic, 2017. Supervisor: Jiří Dvořák. Available: http://hdl.handle.net/11012/67982.

[8] Fried, D., Shimony, S. E., Benbassat, A., and Wenner, C. Complexity of canadian traveler problem variants. *Theoretical Computer Science 487* (2013), 1–16.

[9] Kocsis, L., and Szepesvári, C. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006* (2006), Springer, pp. 282–293.

[10] Nikolova, E., and Karger, D. R. Route planning under uncertainty: The canadian traveller problem. In *AAAI* (2008), pp. 969–974.

[11] Papadimitriou, C. H., and Yannakakis, M. Shortest paths without a map. *Theoretical Computer Science 84*, 1 (1991), 127–150.

[12] Shiri, D., and Salman, F. S. On the online multi-agent o–d *k*-canadian traveler problem. *Journal of Combinatorial Optimization 34*, 2 (2017), 453–461.

[13] Su, B., and Xu, Y. Online recoverable canadian traveller problem. In *Proceedings of the international conference on management science and engineering* (2004), pp. 633–639.

[14] Su, B., Xu, Y., Xiao, P., and Tian, L. A risk-reward competitive analysis for the recoverable canadian traveller problem. In *Combinatorial Optimization and Applications: Second International Conference, COCOA 2008, St. John's, NL, Canada, August 21-24, 2008. Proceedings 2* (2008), Springer, pp. 417–426.

[15] Westphal, S. A note on the k-canadian traveller problem. *Information Processing Letters 106*, 3 (2008), 87–89.

[16] Xu, Y., Hu, M., Su, B., Zhu, B., and Zhu, Z. The canadian traveller problem and its competitive analysis. *Journal of combinatorial optimization 18*, 2 (2009), 195–205.

[17] Yildirim, S., Aksakalli, V., and Alkaya, A. F. Canadian traveler problem with neutralizations. *Expert Systems with Applications 132* (2019), 151–165.

[18] Zhang, H., Xu, Y., and Qin, L. The k-canadian travelers problem with communication. *Journal of Combinatorial Optimization 26* (2013), 251–265.