2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET) | 979-8-3503-3111-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/HONET59747.2023.10374669

Turn Constrained Shortest Path

Amogh Allani
Florida Atlantic University
Boca Raton, USA
Email: aallani2019@fau.edu

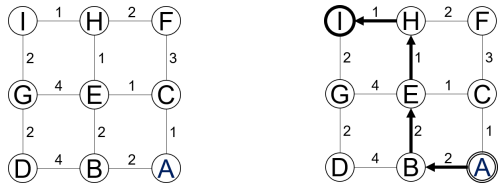
KwangSoo Yang
Florida Atlantic University
Boca Raton, USA
Email: yangk@fau.edu

***Abstract*—Given a transportation network, a source node s, a destination node t, and the number of maximum possible turnings b, the Turn-Constrained Shortest Path (TCSP) problem is to find the route that minimizes the travel distance and meets the turn-constraint. The TCSP problem is important for societal applications such as shipping and logistics, emergency route planning, and traffic management services. We propose novel approaches for TCSP to meet the turn-constraint while minimizing the travel distance for the vehicle route. Experiments using real-world datasets demonstrated that the proposed algorithms can minimize the travel distance and meet the turn-constraint; furthermore, it has comparable solution quality to the unconstrained shortest path and significantly reduces the computational cost.**

***Keywords*—constrained shortest route, routing service, emergency management**

I. Introduction

Given a transportation network, a source node s, a destination node t, and the number of maximum possible turnings b, the Turn-Constrained Shortest Path (TCSP) problem is to find the route that minimizes the travel distance and meets the turn-constraint. Figure 1a shows an example input of TCSP consisting of 9 nodes and 12 edges. Every edge is associated with a distance as indicated by the number displayed over the edge. Let A be the source node and I be the destination node. Assume that the route has no more than one left-turn (i.e., $b = 1$) and minimizes the travel distance. Figure 1b shows an example output of TCSP where the route can minimize the travel distance and meet the turn-constraint (i.e., $A \rightarrow B \rightarrow E \rightarrow H \rightarrow I$).



(a) Input (b) Output

Fig. 1: Example of the Input and Output of TCSP

A. Application Domain

The TCSP problem is important for societal applications such as shipping and logistics, emergency route planning, and traffic management services. Consider the issue of decreasing fuel consumption along a route. In the scope of North America and China, where right-turn on red is broadly permitted, left-turns disadvantage drivers. To take left-turns at intersections, drivers spend more time and fuel idling at signals. The TCSP problem addresses this issue by creating paths that reduce time at intersections. As left-turns are penalized, the route will contain more right-turns, which do not require a green signal. In densely populated road networks (e.g., Los Angeles, Toronto, or Beijing), abundant four-way intersections make the applications of TCSP even more advantageous. For these reasons, TCSP is a fuel-efficient, economical routing option for commuters and businesses.

Importantly, TCSP serves as a safer option in emergency vehicle routing and ensuring driver security. Ambulances and fire trucks must often clear traffic to quickly reach left-turns at an intersection. Since left-turns require a green light, reducing left-turns minimizes the time emergency vehicles need to clear dense traffic. Critically, taking a left-turn can also expose a vehicle to oncoming traffic at intersections without a protected left light. As a result, left-turns have a significantly higher crash factor than right-turns among all drivers [3]. Thus, right-turns encouraged within a TSCP are inherently safer than left-turns. Accordingly, TCSP reduces the need for emergency vehicles to run high-risk left-turns on red lights along their paths. Decreasing response time and further risks is vital to delivering time-critical assistance to an emergency: TSCP accomplishes this task safely and effectively.

For similar reasons, TCSP can assist higher-risk drivers (i.e., drivers affected by age-related disease or novice drivers) with navigating more complex routes. TCSP greatly simplifies routes by reducing the variation between turns. A driver with a memory-related disorder might have greater ease in remembering a TCSP. The scalability of constraints of TCSP could also help simplify routes. In planning routes with detours, for example, TCSP could modify its penalty so that the detour will discourage routes requiring U-turns or three-point turns. Reducing the need for complicated maneuvers minimizes risks for drivers more prone to committing errors. Additionally, in areas with more roundabouts or traffic circles, the penalty could fall on turns taken at the more complex four-way intersections. Given that turns at intersections have a higher crash factor than roundabouts, this would reduce driver errors and accidents.

B. Problem Definition

In our problem formulation, a transportation network is represented as a directed graph of nodes and edges. Every node represents a spatial location in geographic space (i.e., road

intersections) and every edge represents a connection between two nodes and has a distance. The $TCSP(N, E, D, b, s, t)$ problem can be formalized as follows:

Input:

- A transportation network with
- a set of nodes N and a set of edges E,
- a set of positive real distances of edges $D : E \rightarrow R_0^+$
- a source node s and a destination node t, and
- the number of maximum possible turnings b

Output: Turn-Constrained Shortest Path (TCSP)

Objective:

- Minimize the travel distance from s to t.

Constraints:

- Turn-Constraint: The number of turns in the path does not exceed b.

C. Related Work

Shortest Path Problems are extensively used in many fields, such as transportation, logistics, and vehicle routing planning. Given a source node s and a destination node t, the goal of the shortest path problems is to find the route that minimizes the travel cost (e.g., distance, time, etc). Notable examples of shortest path algorithms include Dijkstra's algorithm and A* algorithm [5], [11]. However, these approaches do not always produce the best route in real-world applications due to network constraints (e.g., traffic signals, turn restrictions, gas consumption, etc.) in a transportation network. Furthermore, traditional approaches estimate the turning-cost (i.e., delay and driver turning time) at a intersection and compute the shortest path using Dijkstra's algorithm and A* algorithm [14], [4], [9], [2], [7], [13], [1], [10], [6]. We refer to this problem as the soft turn-constrained shortest path problem. The soft turn-constrained shortest path problem, however, cannot strictly restrict the number of left-turns along the shortest path, resulting in many possible left-turns. More seriously, the turning-cost cannot be correctly estimated due to dynamic time-varying traffic environment (e.g., traffic signal, congestion, and accidents). By contrast, this paper proposes a novel approach for producing a Turn-Constrained Shortest Path (TCSP) that can minimize the travel distance from s to t while satisfying the hard constraint on the number of turning points (i.e., b).

D. Outline

The rest of the paper is organized as follows: Section II describes our proposed approach. Section III describes the experimental design and presents the experimental observations and results. Section IV concludes the paper.

II. Proposed Approach

In this section, we describe our novel approaches to the TCSP problem. Our approach consists of three main components: 1) turn-encoded node, 2) anti-monotone property and distance pruning rule, and 3) admissible heuristic cost function. We theoretically evaluate the proposed approach through a cost model.

A. Turn-Encoded Shortest Path algorithm

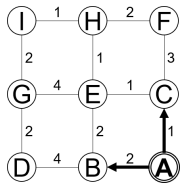
The Turn-Encoded Shortest Path (TESP) algorithm starts with distance estimates from the source and iteratively creates turn-encoded nodes that meet the turn-constraint. The first core idea in TESP is to create turn-encoded nodes and prune the search space that violates the turn-constraint. The turn-encoded node n maintains the number of turns in the path from the source node to the node n.

$$\text{turn-encoded node} = n(i), \tag{1}$$

where n is a node-id and i is the number of turns in the path from the source node s to the node n.

Every turn-encoded node $n(i)$ estimates the upper-bound of the turn-constrained shortest-path distance from the source node s to the node n. It also maintains a pointer to its predecessor to create a simple path.

The Turn-Encoded Shortest Path (TESP) method uses the generalized Dijkstra's algorithm that iteratively updates the estimated distance in an open-node set and selects the open-node with the lowest estimated distance as the next closed-node. It is important to note that each turn-encoded node (i.e., $n(i)$) is unique in both open and closed-node sets, but the node-id can be duplicated.



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
B(0)	2	A(0)
C(0)	1	A(0)

Closed Turn-Encoded Nodes

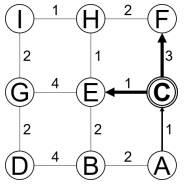
Node	Distance	predecessors
A(0)	0	

(b) Open and Closed Nodes

Fig. 2: TESP: Iteration 1

Consider the input example in Figure 1a. Let A be the source node and let I be the destination node. Assume that the shortest path should have no more than one left-turn (i.e., $b \leq 1$). To simplify the example, let us also assume that every edge is bidirectional with the same distance in both directions.

We first add $A(0)$ to the closed-node set and relax its adjacent nodes (i.e., B and C) (see Figure 2a). Both paths $A \rightarrow B$ and $A \rightarrow C$ have no left-turn; therefore, TESP adds $B(0)$ and $C(0)$ to the open-node set (see Figure 2b).



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
B(0)	2	A(0)
E(1)	2	C(0)-A(0)
F(0)	4	C(0)-A(0)

Closed Turn-Encoded Nodes

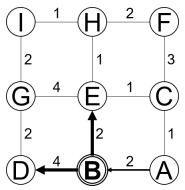
Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)

(b) Open and Closed Nodes

Fig. 3: TESP: Iteration 2

Next, we select the open-node with the lowest estimated distance (i.e., $C(0)$) and relax its adjacent nodes (i.e., E and F) (see Figure 3a). The path $A \rightarrow C \rightarrow F$ has no left-turn

and we add $F(0)$ into the open-node set. However, since node E has one left-turn along the path $A \rightarrow C \rightarrow E$, we increase the number of turns by one (i.e., $E(1)$). Figure 3b shows that TESP creates two open-nodes (i.e., E(1) and F(0)).



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
E(1)	2	C(0)-A(0)
F(0)	4	C(0)-A(0)
D(0)	**6**	**B(0)-A(0)**
E(0)	**4**	**B(0)-A(0)**

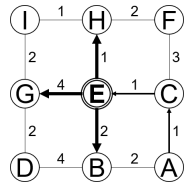
Closed Turn-Encoded Nodes

Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)
B(0)	**2**	**A(0)**

(b) Open and Closed Nodes

Fig. 4: TESP: Iteration 3

After that, we add $B(0)$ to the closed-node set and relax its adjacent nodes (D and E) (see Figure 4a). Since both paths $A \rightarrow B \rightarrow D$ and $A \rightarrow B \rightarrow E$ have no left-turn, we create two turn-encoded nodes (i.e., $D(0)$ and $E(0)$) into the open-node set (see Figure 4b).



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
F(0)	4	C(0)-A(0)
D(0)	6	B(0)-A(0)
E(0)	4	B(0)-A(0)
G(1)	**6**	**E(1)-C(0)-A(0)**
H(1)	**3**	**E(1)-C(0)-A(0)**

Closed Turn-Encoded Nodes

Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)
B(0)	2	A(0)
E(1)	**2**	**C(0)-A(0)**

(b) Open and Close Nodes

Fig. 5: TESP: Iteration 4

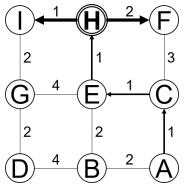
The second core idea of TESP is to remove all infeasible turn-encoded nodes in open-node set according to the anti-monotone property.

Definition 1 (**Anti-monotone Property**). *The turn-encoded node has the anti-monotone property if the node $n(t)$ violates the turn-constraint. If the turn-encoded node $n(t)$ violates the turn-constraint, then all of its successors violate the turn-constraint.*

The relaxation of the closed-node produces a path that stays constant or increases the number of turns. Therefore, we can remove infeasible turn-encoded nodes in the relaxation process. In the fourth iteration, TESP adds $E(1)$ to the closed-node set and relaxes three adjacent nodes (i.e., B, G, and H) (see Figure 5a). Since the path $A \rightarrow C \rightarrow E \rightarrow B$ violates the turn-constraint, we do not add $B(2)$ in the open-node set according to the anti-monotone property. In this example, we create two open-nodes (i.e., $G(1)$ and $H(1)$) (see Figure 5b).

The third core idea of TESP is that we prune the search space using the following distance pruning rule.

Definition 2 (**Distance Pruning Rule**). *Let O be the open-node set and let C be the closed-node set. Then the distance pruning rule removes the turn-encoded node $n(j)$ if $n(i) \in O$ or $n(i) \in C$, $i \leq j$, and $distance(n(i)) \leq distance(n(j))$.*



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
F(0)	4	C(0)-A(0)
D(0)	6	B(0)-A(0)
E(0)	4	B(0)-A(0)
G(1)	6	E(1)-C(0)-A(0)

Closed Turn-Encoded Nodes

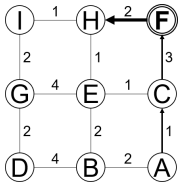
Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)
B(0)	2	A(0)
E(1)	2	C(0)-A(0)
H(1)	**3**	**E(1)-C(0)-A(0)**

(b) Open and Closed Nodes

Fig. 6: TESP: Iteration 5

In the fifth iteration, TESP adds $H(1)$ to the closed-node set and tries to relax its two adjacent nodes (i.e., I and F) (see Figure 6a). The path $A \rightarrow C \rightarrow E \rightarrow F$ produces $F(1)$. Since the distance of $F(0)$ (i.e., 4) in the open-node set is less than $F(1)$ (i.e., 5), we can remove $F(1)$ according the distance pruning rule. We also remove $I(2)$ because the path $A \rightarrow C \rightarrow E \rightarrow H \rightarrow I$ violates the turn-constraint. In this example, $H(1)$ produces no successor (i.e., open-node).

This process terminates in $O(n \cdot b)$ iterations, where n is the number of nodes and b is the turn-constraint. Let k be the number of turn-points on the unconstrained shortest path. Then the number of iterations becomes $O(n \cdot k)$ if $k < b$.



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
D(0)	6	B(0)-A(0)
E(0)	4	B(0)-A(0)
G(1)	6	E(1)-C(0)-A(0)

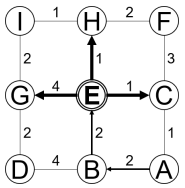
Closed Turn-Encoded Nodes

Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)
B(0)	2	A(0)
E(1)	2	C(0)-A(0)
H(1)	3	E(1)-C(0)-A(0)
F(0)	**4**	**C(0)-A(0)**

(b) Open and Closed Nodes

Fig. 7: TESP: Iteration 6

In this example, TESP requires three more iterations to find a turn-constrained shortest path (TCSP) (see Figures 7-9). Figure 7a shows that $F(0)$ has one successor (i.e., H) along the path $A \rightarrow C \rightarrow F \rightarrow H$. We can remove it because TESP has already closed the turn-encoded node $H(1)$ (see 7b).



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
D(0)	6	B(0)-A(0)
G(1)	6	E(1)-C(0)-A(0)
H(0)	**5**	**E(0)-B(0)-A(0)**

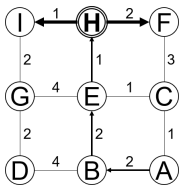
Closed Turn-Encoded Nodes

Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)
B(0)	2	A(0)
E(1)	2	C(0)-A(0)
H(1)	3	E(1)-C(0)-A(0)
F(0)	4	C(0)-A(0)
E(0)	**4**	**B(0)-A(0)**

(b) Open and Closed Nodes

Fig. 8: TESP: Iteration 7

Figure 8a shows that $E(0)$ has three successors (i.e., C, G, and H). We can remove $C(0)$ because TESP has already closed the turn-encoded node $C(0)$. We can also remove $G(1)$ according to the distance pruning rule. We add $H(0)$ to the open-node set along the path $A \rightarrow B \rightarrow E \rightarrow H$ (see Figure 8b).



(a) Graph

Open Turn-Encoded Nodes

Node	Distance	predecessors
D(0)	6	B(0)-A(0)
G(1)	6	E(1)-C(0)-A(0)
I(1)	**6**	**H(0)-E(0)-B(0)-A(0)**

Closed Turn-Encoded Nodes

Node	Distance	predecessors
A(0)	0	
C(0)	1	A(0)
B(0)	2	A(0)
E(1)	2	C(0)-A(0)
H(1)	3	E(1)-C(0)-A(0)
F(0)	4	C(0)-A(0)
E(0)	4	B(0)-A(0)
H(0)	**5**	**E(0)-B(0)-A(0)**

(b) Open and Closed Nodes

Fig. 9: TESP: Iteration 8

Lastly, Figure 9a shows that $H(0)$ has two successors (i.e., F and I). The distance of $F(0)$ (i.e., 4) in the closed-node set is less than the distance of the path $A \rightarrow B \rightarrow E \rightarrow H \rightarrow F$ (i.e., 7). Therefore, we can remove it. The path $A \rightarrow B \rightarrow E \rightarrow H \rightarrow I$ produces the turn-encoded node $I(1)$. Since the distance of all turn-encoded nodes in the open-node set is equal to the distance of $I(1)$ (i.e., 6), the TESP algorithm returns a TCSP (i.e., $A \rightarrow B \rightarrow E \rightarrow H \rightarrow I$) (see Figure 9b).

Algorithm 1 TESP Algorithm (Pseudo-code)

Inputs:
- A transportation network G with a set of nodes N and a set of edges E,
- a set of positive real distances of edges $D : E \rightarrow R_0^+$
- a source node s and a destination node t, and
- the number of maximum possible turnings b

Outputs: Turn-Constrained Shortest Path

Steps:

1: Create a open-node set O and a closed-node set C.
2: Create a priority queue Q.
3: $O = Q \cup s(0)$
4: **while** Q is empty **do**
5: $u(i) = EXTRACT{-}MIN(Q)$
6: **if** u(i) is the destination **then**
7: break
8: **end if**
9: $C = Q \cup u(i)$
10: **for each** $v \in Adjacent(u)$ **do**
11: Create turn-encoded node $v(j))$
12: Remove $v(j)$ if $j > b$
13: Remove $v(j)$ if $v(k) \in O$, $k \leq j$, and $Dist(v(k)) \leq Dist(v(j))$
14: $Relax(u(i), v(j))$
15: **end for**
16: **end while**
17: Identify the turn-constrained shortest path (TCSP).
18: **return** TCSP

Algorithm 1 presents the pseudo-code for a generalized version of TESP. First, it creates open-node set, closed-node set, and priority queue (Lines $1{-}2$). It then adds the source node (i.e., $s(0)$) into the priority queue (Line 3) and iteratively expands the search space (Lines $4{-}17$). At each iteration, TESP selects the closed-node with the least distance (Line 5) and removes all infeasible adjacent nodes according to the anti-monotone property and distance pruning rule (Lines $12{-}13$). It then relaxes the remaining adjacent nodes (Line 14) and continues until the priority queue is empty (Line 4) or the destination node is found (Line $6{-}7$). Finally, it returns a TCSP (Lines $17{-}18$).

B. Turn-Encoded Shortest Path with A algorithm*

A large-sized network may have many feasible turn-constrained paths. One of the best ways to further reduce the search space for TESP is to utilize the underestimated heuristic function based on the A* algorithm [8], [11]. The A* algorithm uses the evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the travel distance from the source node s to the node n and $h(n)$ is the estimated distance of the turn-constrained shortest path from the node n to the destination node t. It is well known that A* is cost-optimal when $h(n)$ is an admissible heuristic function [11].

Definition 3 (**Admissible Heuristic**). *The turn-constrained shortest-path distance $h(n, t, \hat{b})$ is admissible for all nodes n on the turn-constrained shortest path of $TCSP(N, E, D, b, s, t)$ if $b \leq \hat{b}$.*

The cost for choosing the next closed-node for TESP can be defined using the following evaluation function:

$$f(n, b) = g(n) + h(n, \hat{b}), \tag{2}$$

where n is a node-id, b and $\hat{b}$ are a turn-constraint, and $b \leq \hat{b}$.

The Turn-Encoded Shortest Path with A* algorithm (TESP-A*) uses the admissible heuristic function that can stretch the search space toward the destination node and reduce the number of iterations of the TESP algorithm. It precomputes and stores the values of the admissible heuristic function between nodes and the destination node. Given a source s and a destination node t, the space complexity of the heuristic function (i.e., $h(n, \hat{b})$) is $O(n)$, where n is the number of nodes and $\hat{b}$ is the turn-constraint.

TESP-A* can be viewed as the generalized version of the bi-directional search algorithm. Let $turns(g(n))$ be the number of turns for $g(n)$ and let $turns(\hat{b})$ be the number of turns for $h(n, \hat{b})$. Assume that $turns(g(n)) = i$ and $turns(\hat{b}) = j$. Then the cost of $f(n, b) = g(n)+h(n, \hat{b})$ is optimal if $i+j \leq b$ and $b \leq \hat{b}$. If $f(n, b)$ is the optimal cost of TCSP, then TESP-A* is the same as the bi-directional search algorithm. We can use this condition as the early stopping criterion for TESP-A*.

Definition 4 (**Tight Admissible Heuristic**). *The turn-constrained shortest-path distance $h(n, t, b - i)$ is admissible if b is the turn-constraint and $turns(g(n)) = i$*

We can materialize the turn-constrained shortest-path distance $h(n, t, b - i)$ for all $i \leq b$ [12]. The space complexity of the tight admissible heuristic functions becomes $O(n \cdot b)$. However, the computation of $h(n, t, \hat{b})$ for each $\hat{b} \leq b$ is challenging for a large-sized transportation network. In our proposed approach, we remove the turn-constraint (i.e., $\hat{b} = \infty$) and use the shortest-path distance as the admissible heuristic for TESP-A*.

C. Asymptotic analysis of the proposed approaches

We developed a cost model for the proposed approaches.

1) TESP: Let n be the number of nodes, let m be the number of edges, and let b be the number of turns (i.e., turn-constraint). TESP requires at most $O(n \cdot b)$ iterations. The relaxation examines at most $m \cdot b$ edges. TESP creates a turn-encoded path at a cost of $O(n \cdot b \cdot \log(n \cdot b) + m \cdot b)$. Assume that $m = O(n)$. Then the computational complexity of TESP is $O(n \cdot b \cdot \log(n \cdot b))$.

2) TESP-A:* The main difference between TESP and TESP-A* is the admissible heuristic function and the early stopping criterion. The cost model of TESP-A* is the same as that of TESP. The computational complexity becomes $O(n \cdot \log n)$ if we use tight admissible heuristic function. This is because only one turn-encoded node is required for each node to evaluate the tight admissible heuristic function.

III. Experimental Evaluation

We conducted experiments to evaluate the performance of our proposed approaches (i.e., TESP and TESP-A*). The overall goal was to show the performance improvements to create a Turn-Constrained Shortest Path (TCSP). We wanted to answer five questions: (1) What is the effect of the size of the network (i.e., number of nodes (n))? (2) What is the effect of the path-length (i.e., l)? (3) What is the effect of the turn-constraint (i.e., b)? (4) Does the solution of the proposed approaches meet the turn-constraint? (5) Are the proposed approaches scalable for large-sized transportation networks?

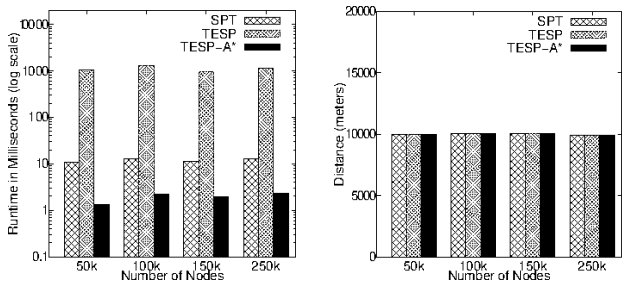
A. Experimental Layout

We tested three different approaches: 1) Turn-Encoded Shortest Path (TESP), 2) Turn-Encoded Shortest Path with A* (TESP-A*), and 3) Dijkstra's Shortest Path Algorithm. It is worth noting that the shortest-path distance can serve as a lower bound of the turn-constrained shortest-path distance. However, the shortest path cannot preserve the turn-constraint. For the transportation network, we chose ten different areas in the Florida map from OpenStreetMap and created directed graphs. In our experiment, we use the shortest-path distance (i.e., $\hat{b} = \infty$) as the admissible heuristic for TESP-A* (see Equation 2). We precomputed the shortest paths and materialized the values of the admissible heuristic function for the TESP-A* approach. For simplicity, we consider only the left-turn-constraint in our analysis. The algorithms were implemented in Java 1.8 with a 32 GB memory run-time environment. All experiments were performed on an Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz machine running Ubuntu 16.04.6 LTS with 32GB of RAM.

B. Experimental Results and Analysis

We experimentally evaluated the proposed algorithms by comparing the impact on performance of (1) size of the transportation network (i.e., number of nodes (n)), (2) path-length (i.e., l), and (3) the turn-constraint (i.e., b). Performance measurements were execution time, the length of the path (i.e., travel distance) from the source node to the destination node, and the number of violations of the turn-constraint.

1) Effect of Network Size: The first set of experiments evaluated the effect of the size of the transportation network (i.e., n). We used a Florida map and created ten road networks with $50,000$, $100,000$, $150,000$, and $200,000$ nodes, respectively. We fixed the turn-constraint (i.e., b) to 10 and the path-length (i.e., l) to $10km$. We varied the number of nodes (i.e., n) from $50,000$ to $200,000$. The performance measurements were averaged over 100 test runs for each network. Figure 10a gives the execution time. As can be seen, TESP-A* outperforms TESP. This is because TESP-A* utilizes an admissible heuristic to reduce the search space. We can see that the number of nodes has little effect on algorithm performance because the size of the search space of the three algorithms depends on the path-length. Figure 10b shows that the shortest path approach performs slightly better than TESP and TESP-A*. However, the shortest paths violated the turn-constraint and produced 15.2, 16.5, 18.7, and 17.6 left-turns on average, respectively.

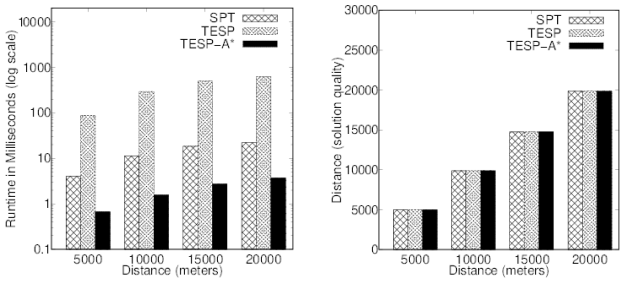


(a) Run Time Comparison (b) Solution Quality

Fig. 10: Effect of number of nodes (b = 10 and l = $10km$)

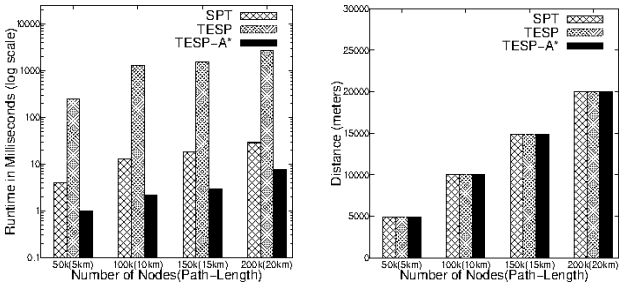
2) Effect of Path-Length: The second set of experiments evaluated the effect of the path-length (i.e., l) on algorithm performance. We fixed the number of nodes (i.e., n) to $100,000$ and fixed the turn-constraint (i.e., b) to 6. We varied the path-length (i.e., l) from $5km$ to $20km$. The performance measurements were averaged over 100 test runs for each path-length. Figure 11a shows that TESP-A* outperforms TESP. As the path-length increases, the performance gap also increases. This is because the search space of the three algorithms increases as the path-length increases. We can see that TESP-A* can efficiently reduce the search space for finding the turn-constrained shortest path. When comparing the solution quality (i.e., travel distance) of the path, the three algorithms are almost identical, even though the shortest path performs slightly better (see Figure 11b). However, the shortest paths violated the turn-constraint and produced 10.5, 12.5, 13.2, and 13.1 left-turns on average, respectively.

3) Effect of Turn-Constraint: The third set of experiments evaluated the effect of the turn-constraint (i.e., b) on algorithm performance. We fixed the number of nodes (i.e., n) to $100,000$ and fixed the path-length (i.e., l) to $10km$. We varied the turn-constraint (i.e., b) from 4 to 10. The performance measurements were averaged over 100 test runs for each turn-constraint. Figure 12a gives the execution time. As can be seen, TESP-A* significantly outperforms TESP. As the value



(a) Run Time Comparison (b) Solution Quality

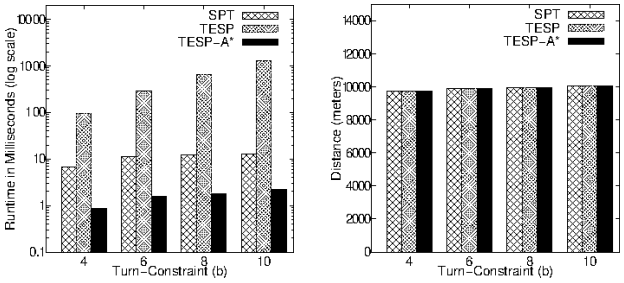
Fig. 11: Effect of path-length (n = $100K$ and b = 6)



(a) Run Time Comparison (b) Solution Quality

Fig. 13: Scalability of TESP-A* (b = 10)

of turn-constraint (i.e., b) increases, so does the performance gap. This is because the search space increases for TESP as the value of the turn-constraint increases. Figure 12b shows that the shortest path approach performs slightly better than TESP and TESP-A* in terms of the travel distance. However, the shortest paths violated the turn-constraint and produced 8.9, 12.5, 14.4, and 16.5 left-turns on average, respectively.



(a) Run Time Comparison (b) Solution Quality

Fig. 12: Effect of turn-constraint (n = $100K$ and l = $10km$)

4) Scalability of TESP-A:* The fourth experiment verified the scalability of TESP-A*. We fixed the turn-constraint (i.e., b) to 10 and incrementally increased the number of nodes (i.e., n) from $50,000$ to $200,000$. We also increased the path-length (i.e., l) proportional to the number of nodes (i.e., $5km - 20km$). Figure 13a shows that TESP-A* significantly outperforms TESP. As both the number of nodes and the path-length increase, so does the performance gap. This is because TESP-A* can efficiently prune the search space based on admissible heuristic. Figure 13b shows that the solution of quality of the shortest path is slightly better than other solutions in terms of the travel distance. However, the shortest paths violated the turn-constraint and produced 16, 16.4, 19.3, and 18.8 left-turns on average.

IV. Conclusion

We presented the problem of creating a Turn Constrained Shortest Path (TCSP). In this paper, we introduced a novel Turn-Encoded Shortest Path (TESP) approach for producing a TCSP to meet the turn-constraint while minimizing the travel distance. We also proposed TESP-A* that incorporates the admissible heuristic function to reduce the computational cost of TESP. We presented experiments using the Florida map, which demonstrated that our proposed algorithm had comparable solution quality to the shortest path in terms of travel distance, produced a path with the turn-constraint, and significantly reduced the computational cost.

V. Acknowledgments

This material is based upon work supported by the National Science Foundation CAREER under Grant No. 1844565.

References

[1] Buchhold, V., Wagner, D., Zeitz, T., and Zündorf, M. Customizable contraction hierarchies with turn costs. In *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)* (2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[2] Caldwell, T. On finding minimum routes in a network with turn penalties. *Communications of the ACM 4*, 2 (1961), 107–108.

[3] Choi, E.-H. Crash factors in intersection-related crashes: An on-scene perspective. Tech. rep., 2010.

[4] Clossey, J., Laporte, G., and Soriano, P. Solving arc routing problems with turn penalties. *Journal of the Operational Research Society 52*, 4 (2001), 433–439.

[5] Demetrescu, C., Goldberg, A. V., and Johnson, D. S. *The shortest path problem: Ninth DIMACS implementation challenge*, vol. 74. American Mathematical Soc., 2009.

[6] Geisberger, R., and Vetter, C. Efficient routing in road networks with turn costs. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10* (2011), Springer, pp. 100–111.

[7] Gutiérrez, E., and Medaglia, A. L. Labeling algorithm for the shortest path problem with turn prohibitions with application to large-scale road networks. *Annals of Operations Research 157* (2008), 169–182.

[8] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics 4*, 2 (1968), 100–107.

[9] Kirby, R. F., and Potts, R. B. The minimum route problem for networks with turn penalties and prohibitions. *Transportation Research 3*, 3 (1969), 397–408.

[10] Pallottino, S., and Scutella, M. G. Shortest path algorithms in transportation models: classical and innovative aspects. In *Equilibrium and advanced transportation modelling*. Springer, 1998, pp. 245–281.

[11] Russell, S. J. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[12] Shekhar, S., Fetterer, A., and Goyal, B. Materialization trade-offs in hierarchical shortest path algorithms. In *Advances in Spatial Databases: 5th International Symposium, SSD'97 Berlin, Germany, July 15–18, 1997 Proceedings 5* (1997), Springer, pp. 94–111.

[13] Vanhove, S., and Fack, V. Route planning with turn restrictions: A computational experiment. *Operations Research Letters 40*, 5 (2012), 342–348.

[14] Winter, S. Modeling costs of turns in route planning. *GeoInformatica 6* (2002), 345–361.