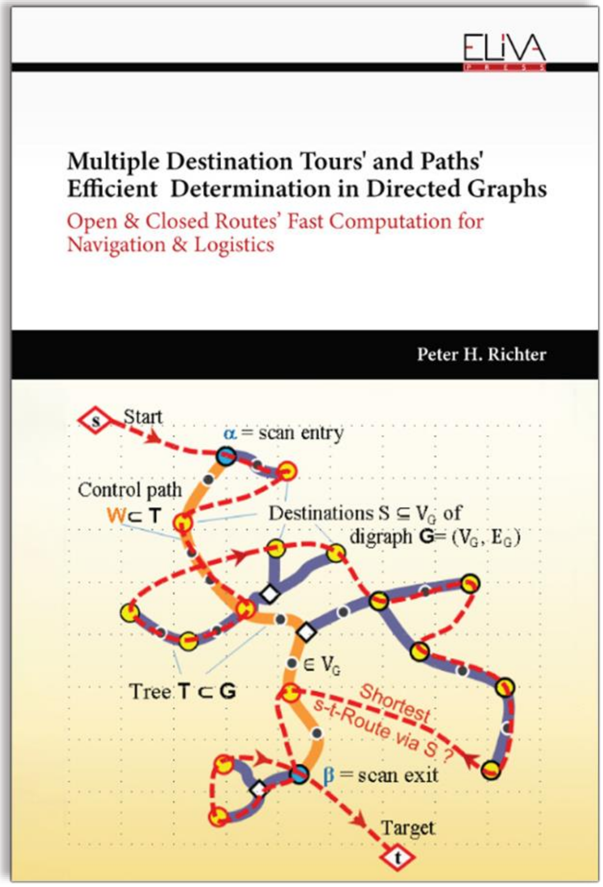
Multiple Destination Tours' and Paths' Efficient Determination in Directed Graphs

Open & Closed Routes' Fast Computation for Navigation and Logistics

Peter H. Richter

ISBN 978-1-63648-284-2, ELIVA PRESS, 2021



The *Asymmetric Steiner Traveling Salesman Path Problem* (*ASTSPP*) has been unattended in the past despite its high practical importance for real-time navigation in digital traffic nets! We are given a digraph $\vec{G}$ with asymmetric arc weights, start point **s**, target point **t**, and a subset $S \subseteq V(\vec{G})$. The objective is to find a shortest route from **s** to **t** in $\vec{G}$ visiting all destinations S at least once. The proposed deterministic solution approach relies on meta-heuristics

(a) *Advanced Scan of Spanning Trees (ASST)* applied to approx.. Steiner trees $\mathbf{T} \subset \vec{G}$, $S \subset V_T$,

(b) *Tree Structure Adaption* (*TSA)* that overcomes "flaws" of tree **T** hampering good results,

(c) *Confined Complete Enumeration (CCE)* that rearranges the sequence of the last $n \leq 6$ spots of S each time a new successor $x \in S$ has been found,

(d) *Compact Priority Queue (CPQ)* essentially reducing run-time effort.

The correspondingly implemented algorithm $\mathfrak{A}$ meets today's demands for navigation and logistics that require to allow …

graphs G or digraphs $\vec{G}$ being not necessarily complete ones	➔ $G \subset K_n$ or $\vec{G} \subset \vec{K}_n$,
the use of asymmetric arc costs $\vec{\lambda}$: $E_{\vec{G}} \rightarrow \mathbb{R}_{\geq 0}$	➔ "Asymmetric" problems,
routes not required to visit all V_G but a subset $S \subseteq V_G$	➔ "Steiner" problems,
routes not required to visit vertices $S \subseteq V_G$ exactly once	➔ not Hamiltonian routes,
graphs not committed to observe the triangle inequality	➔ graph metric not required,
open or closed trips by one algorithm with the same efficiency	➔ pari passu **s**= **t** or **s**≠ **t**,
consideration of crossings' road signs / traffic commandments	➔ edge-queuing SPT algorithms
drawing advantage from planar graphs for algorithmic efficiency	➔ traffic maps are planar.

Algorithm $\mathfrak{A}$ …

- tackles digraphs as well as undirected graphs ($\vec{\lambda}$(x, y)= $\vec{\lambda}$(y, x)= λ(x, y)),
- can process the cases S= V_G as well as S ⊂ V_G,
- can process the cases **s**= **t** as well as **s** ≠ **t**.

That means, $\mathfrak{A}$ meets all problems **X** of List 1. That are those, not persisting on Hamiltonian structures.

In graph G= (V_G, E_G) or digraph $\vec{G}$=(V_G, $E_{\vec{G}}$) we look for a bijection π describing a) closed path b) s – t – path / c) closed walk c) s – t – walk via S ⊆ VG.			Instance $\mathfrak{I}$=	Round trip?	G directed?	x∈ S to visit	S= V_G?
	TSP	Traveling Salesman Problem	(G, λ)	y	n	1	S= V_G
	ATSP	Asymmetric TSP	($\vec{G}$, $\vec{\lambda}$)		y		
	TSPP	Traveling Salesman **P**ath Problem	(G, λ, s, t)	n	n	1	
X	TSWP	Traveling Salesman **W**alk Problem.				≥ 1	
	ATSPP	Asymmetric TSPP	($\vec{G}$, $\vec{\lambda}$, s, t)		y	1	
X	ATSWP	Asymmetric TSWP				≥ 1	
X	STSP	Steiner Trav. Salesman Problem	(G, λ, S)	y	n	≥ 1	S⊆ V_G
X	ASTSP	Asymmetric STSP	($\vec{G}$, $\vec{\lambda}$, S)		y		
X	STSPP	Steiner Tr. Salesman Path Pr.	(G, λ, S, s, t)	n	n		
X	ASTSPP	Asymmetric STSPP	($\vec{G}$, $\vec{\lambda}$, S, s, t)		y		

Graph	**G**= (V_G, E_G), λ: $E_G \rightarrow \mathbb{R}_{\geq 0}$,
Digraph	$\vec{\mathbf{G}}$= (V_G, $E_{\vec{G}}$), $\vec{\lambda}$: $E_{\vec{G}} \rightarrow \mathbb{R}_{\geq 0}$,
Destinations	**S** ⊆ V_G, start s, target t,
Result:	Bijection $\boldsymbol{\pi}$: {1, 2, …|S|} → S,
X:	Problems efficiently solved by ASTSPP algorithm $\mathfrak{A}$

x∈ S to be visited:
1 – exactly once
≥ 1 – at least once

List 1 The TSP Problem Family – a rough overview

The deterministic $O(n^3)$-algorithm $\mathfrak{A}$ efficiently solves ASTSPP for open and closed tours. It directly uses digital traffic maps G (digraphs) without to twist them to complete ones, what enables that digital traffic map alterations (i.e. turn restrictions, activating / deactivating one-way-streets, considering traffic jam, etc.) can be directly transferred into the domain of $\mathfrak{A}$ in real-time.

The introduction of *Compact Priority Queue (CPQ)* Θ compared to the conventional use of $\mathbb{A} \times \mathbb{B}$ (for choosing a set Θ of pairs (scan-entry α, scan-exit β, |Θ|= |$\mathbb{A}$|= |$\mathbb{B}$|)) leads to an essential runtime reduction without to peril getting near-optimal results. The new optimization features *Advanced Scan of Spanning Trees* ***ASST***, *Confined Complete Enumeration* ***CCE*** and *Tree Structure Adaption* ***TSA*** show an impressive optimization potential. However, Table 1 shows that ***TSA*** seems to lose some of its improvement potential the more graph G is crammed with spots S. The reason is that a dense ramification $|S| / |V_G| \geq 2/3$ of tree **T** increasingly hampers ***TSA*** finding improvement possibilities.

Regarding larger problem sizes 100≤ |S|≤ 1.000, Table 2 shows that the common use of ***CCE*** and ***TSA*** gets 15% length improvement compared to the pure use of ***ASST***. This improvement capability persists also for |S|= 1.000 where the destination denseness is not so high.

Testing real-time ability (nominal time ≤ 2 sec) algorithm $\mathfrak{A}$ running on a 2,70 GHz PC using graph G_2 with |Θ| =3 accomplishes tours as with ***TSA***: |S|= 115 and without ***TSA***: |S|= 1590.

That means that time-critical apps processing higher problem sizes should use ***CCE*** without ***TSA*** to keep real-time ability provided a loss of about 5% solution quality is accepted.

Comparing results with the optimal ones, we used large set of random graphs for a problem size |S|= 14 considering remote distant start and target and near distant ones (**s**= **t**): Algorithm $\mathfrak{A}$ running with ***CCE*** and ***TSA*** (|Θ| = 5) didn'**t** surpass a sample standard deviation q_{max}= 1,86 %! Optimization feature ***CCE*** (δ= 5) contributes evenly over |S| for the most respectable result improvement. However, only ***TSA*** changing $\mathbf{T} \overset{\wp}{\rightarrow} \mathbf{T'}$ w.r.t. proposal $\wp$ enables ***CCE*** providing its noticeable contribution to the total result improvement. Again, Since $\mathfrak{A}$ tackles the cases (S⊂ V_G & S= V_G), (**s** ≠ **t** & **s**= **t**) and (digraphs & undirected graphs) it efficiently solves not only ASTSPP but also TSWP, ATSWP, STSP ASTSP, STSPP.

For a discussion about the TSP's Problem Family and related publications and for a detailed description of Algorithm $\mathfrak{A}$, its meta-heuristics and comprehensive performance analysis (C++ implementation, Microsoft Visual Studio 2012) we refer to the book

"Multiple Destination Tours' and Paths' Efficient Determination in Directed Graphs"
Peter H. Richter
ISBN 978-1-63648-284-2, ELIVA PRESS, August 2021