The Generalized Traveling Salesman Path Problem GTSPP relayed to the Asymmetric Steiner Traveling Salesman Path Problem ASTSPP - An Impactful Construction Algorithm

Peter H. Richter / O&S Consultancy Berlin / peterhrichter@online.de

Abstract

The GTSPP looks in asymmetric directed graphs, whose nodes are partially segmented into several clusters (services, classes), for a shortest walk from a given start ***s*** to a given target ***t*** such that each cluster is visited at least once. The proposed construction method arises from the problem's decomposition into (a) *Search for a Densest Set S of Service Locations* (shortly *Service Location Cloud Problem* SLCP) containing at least one location each clusters such that the sum of the distances among these locations as well as to ***s*** and ***t*** is minimal and (b) the *Asymmetric Steiner Traveling Salesman Path Problem* ASTSPP that looks for a shortest ***s***-***t***-walk through all these nodes S. A novel SLCP algorithm determines the service location cloud S and a novel ASTSPP algorithm builds and scans an approximate Steiner tree T(S) to get a shortest walk from ***s*** to ***t*** through S using a new meta-heuristic *Advanced Scan of Spanning Trees* ASST.

During this scan a new meta-heuristic *Tree Structure Adaption* TSA determines in the background proposals to change afterwards T(S) to an "improved" tree T′(S) whose scan again most probably shortens the walk. As the tests on large high density digraphs reveal the proposed heuristic gets near-optimal results, i.e. with a sample standard deviation $\boldsymbol{q}_{\max}$= 2,5% for moderate instances in real-time.

Summary

Tackling the Generalized Traveling Salesman Path Problem GTSPP, the proposed construction algorithm $\widehat{\mathfrak{A}}$ arises from the problem decomposing into the *Service Location Cloud Problem* SLCP and the *Asymmetric Steiner Traveling Salesman Path Problem* ASTSPP. SCLP algorithm $\widehat{\mathfrak{A}}_{\mathbf{P1}}$ determines a densest set of service locations Ω integrating all clusters such that the sum of the distances among its locations as well as to $\boldsymbol{s}$ and $\boldsymbol{t}$ is minimal. It uses a novel metaheuristic ***Cloud***() that determines a convenient set Ω of service locations. ASTSPP algorithm $\widehat{\mathfrak{A}}_{\mathbf{P2}}$ builds a Steiner tree T(Ω) and scans T(Ω) from $\boldsymbol{s}$ to $\boldsymbol{t}$ using the very efficient meta-heuristics *Advanced Scan of Spanning Trees* ASST and *Confined Complete Enumeration* CCE to get a least cost tour $\boldsymbol{\pi}$:{0, 1, ..., |Ω|+1}→ Ω ∪ {$\boldsymbol{s}$, $\boldsymbol{t}$}. During its scan, $\widehat{\mathfrak{A}}_{\mathbf{P2}}$ looks with meta-heuristic *Tree Structure Adaption* TSA for a proposal $\wp$ to change T(Ω) $\overset{\wp}{\rightarrow}$ T′(Ω) for a new tree T′(Ω) whose scanning by ASST afterwards generally provides a shorter tour π′ compared to π build before. Comparing to the optima it was

necessary to confine the test to $\boldsymbol{n}_{\mathfrak{C}}$ = 6 clusters and 10 locations each cluster. Algorithm $\mathfrak{A}$ receives a Sample Standard Deviation SSD $\boldsymbol{q}_{\max}$= 2,5% needing about 150 ms, Diagram 1. For larger problem sizes $\boldsymbol{n}_{\mathfrak{C}}$ = 15 and 15 locations each cluster, optimization feature TSA jeopardizes the real-time ability (nominal time ≤ 2.000 ms), Diagram 2, wherefore the deactivation of TSA is an option to meet the requirements of time critical apps. $\mathfrak{A}$ yields its best results if at least 45% quota of heap $\mathfrak{H}$ are processed with a time effort of 7 s. Still, real-time ability can be granted if $\mathfrak{A}$ avoids TSA so far the user accepts a result deterioration of 1,8% but decreasing nominal time to 1,3 s.

Algorithm $\mathfrak{A}$ runs with $\boldsymbol{O}(|\Phi|\cdot n^3)$. It covers the cases $\Phi \subset \mathrm{V}_{\mathbf{G}}$ and $\Phi = \mathrm{V}_{\mathbf{G}}$, $\boldsymbol{s}$= $\boldsymbol{t}$ and as $\boldsymbol{s} \neq \boldsymbol{t}$, and digraphs and undirected graphs and the cost function hasn't to rely on a metric property. It allows the service locations of all clusters $\mathfrak{C}$ be disseminated over and mingled in V_{G} by any possible distribution. Algorithm $\mathfrak{A}$ is especially convenient for adopting parallel computing.

Today's apps for navigation and logistics are increasingly required to allow:

1. graphs G or digraphs $\vec{G}$ that are not necessarily complete ones => $G \subset K_n$ or $\vec{G} \subset \vec{K}_n$,
2. the use of asymmetric arc costs $\vec{\lambda}: E_{\vec{G}} \rightarrow \mathbb{R}_{\geq 0}$ => "Asymmetric" problems,
3. routes that enable visiting only a subset $S \subseteq V_G$ => "Steiner" problems,
4. routes where vertices might be visited more than once => Walks",
5. shortest (s-t)-routes through $S \subset V_G$ from a start s to a target t => "s-t paths" or "s-t-walks",
6. graphs not committed to observe the triangle inequality => graph metric not required,
7. considering the crossings' road signs / traffic commandments => edge-queuing SPT algorithms
8. open or closed routes by one algorithm with the same efficiency => pari passu $s = t$ or $s \neq t$,
9. drawing advantage from using planar graphs => traffic maps are planar,
10. overlapping (not necessarily separated) cluster regions => meets general demands.

The new efficient algorithm GTSPP_2 meets these requirements!