Spring 2023 CS 687 Capstone Project

On the Benefits of a Proposed Method of Quantifying Complexity Induction for the Purpose of Generating Feedback Loop Parameters for Assistance When Pursuing Optimized Heuristic Solutions to 2D Traveling SalesPerson Type Problems Using a Nested Triangles Derived Convex Hulls Solver Adhering to Both Shortest Delta Distance and Inner-to-Outer Layer Consumption Criteria

Adrian Wilkinson
Advisor: Dr. Ali Khamesipour
MS in Computer Science
School of Technology & Computing (STC)
City University of Seattle (CityU)
wilkinsonadrian@cityuniversity.edu,
khamesipourali@cityu.edu

ABSTRACT

Solving 2D recombinant challenges as quickly and as accurately as possible is well documented as a worthy goal. This work investigates the possible benefits of assimilating a specific controlled complexity induction approach inside of a feedback loop for purpose of assisting in solving these type problems. The well known TSP challenge was selected as the medium in which to perform this investigation. A readily controllable complexity induction method was developed. This control method was then aligned with a deterministic 2D TSP problem solution method: more specifically, with a nested triangles derived convex hulls solver employing both shortest-distance merge and inner-to-outer layer consumption criteria. Results show that when the subject complexity induction control method was woven into the aforementioned deterministic nested triangles solution method, the resultant reduced rigidity, learning based merge decisions do provide improved solutions. Additionally, there was clear evidence of healing ability regarding non-validating cross-over features associated with the purely deterministic derived solutions.

Keywords: Feedback, loop, TSP, convex, hull, nested, triangle, polygon, tetrahedron, combinatorial, recombinant, complexity, induction, minimum, criterion, algorithm, consumption, dissolution, AUC

1. INTRODUCTION

"The classical traveler problem has a machine tool or person that starts from one point, visits n-1 other points once, and then returns to the starting point.", Ref 1. Potential and realized applications are many and have been well documented, Ref. 2 One such application assists kidney exchange programs to overcome constraints and combinatorial complexity to efficiently arrive at optimal solutions when determining which exchanges will take place, Ref. 3.

The use of the convex hull has a long history. Cronin (Ref. 4) wrote:

"In 1957 Barachet (*Ref. 5*) proved that there exists an optimal tour which preserves the relative order of the points on the convex hull."

Cronin also wrote, regarding convex hulls:

"The purpose of computing the onion is to gain control of the search space by attempting to insert the cities uniformly into the hull, to limit generation of "greedy" perturbations. A greedy perturbation occurs when by mere virtue of having probed sufficiently far into the hull, a perturbed hull segment continues to absorb cities which rightfully belong to another perturbation."

Cronin also considered the nested hull traversal from both the "Outside-in" and the "Inside-out" perspective.

Many other researchers have investigated TSP solution using convex hulls, Ref. 6 through Ref. 10.

It was speculated that a hobbyist-constructed simplistic 2D TSP problem solving script could be improved by incorporating a more complex approach that injects flexibility to the instantiated, infinitely rigid shortest delta distance merge decision-making process. The goal was to produce a better algorithm. Control of complexity induction was used to accomplish the wanted de-rigidisation. The algorithm was suitably updated. The process of obtaining and reviewing results morphed into a feedback system. The goal of the research was then gradually altered to the title of this paper.

Problem Statement
The benefits of using a complexity induction based feedback loop as an assist when solving recombinant problems has not been exhaustively studied.

The goal of this paper is to evaluate usage of controlled complexity induction for next merge decision-making, specifically for solving recombinant challenges The inclusion of a feedback approach, whether manual or automated, is considered as essential for reasonable success. The recombinant challenge is provided by t2D STP (Travelling Sales Person) problem datasets, Ref. 12 through Ref. 17.

An underlying deterministic method was needed. This hobbyist's previously scripted nested triangles algorithm was selected. This code provided encouraging results when supplied with 2D datasets that could be defined as nested polygons (whether triangles and/or higher order).

Motivation
Addressed by the preceding.

Value to the Student
This topic will provide the author with vessel to display propensity for both abstract and critical thinking. This topic will also provide the author with an opportunity to document the ability to reduce concepts and postulations to an applicable, useful computer algorithm. This topic will provide the author with an opportunity to increase skills with and knowledge of the python programming language. This topic will provide the author to advance technical report writing skills.

The above skills comprise a significant subset of a Master's Degree in Computer Science holder's tool box. Displaying these skills in advance of an interview might be beneficial.

Alignment to Program Outcomes
[Briefly describe how this topic or program is aligned to the MS CS program outcomes in a paragraph or two.
1. Construct a strong foundation of ethical knowledge in computer science (Yes)
2. Explain how to apply in-depth knowledge to one or more areas of interest in computer science. (Yes)
3. Defend the findings of the programming or research project (Yes)
4. Explain an application of a broad set of principles, tools, and techniques in computer science (Yes, particularly from perspective of algorithm construction)
5. Defend a research problem in computer science. (Yes)

2. BACKGROUND

Early work: Scope
A solver was constructed for intent of calculating shortest path (TSP definition) for any planar set of data that lends itself to visualization as a series of nested triangles with either zero, one or two remnant nested innermost points. This activity was undertaken as a follow on of postulation, later reinforced with literature search research, including Ref. 18, that for any 2D dataset, the data set that may be grouped into a series of convex triangles (defined per above) would require the greatest number of merge "decisions" during shortest path calculation when compared against any other 2D data set, of same number of data points, which coincidentally may not be sorted into nested

triangles, assuming the inner-to-outer layer dissolution approach was used. Obviously, the latter data set, which lends itself to regrouping into convex polygons, but absolutely not into nested triangles only, is much more common in our environment, which encompasses both human-generated datasets and the naturally occurring (whether biological, geological or similarly derived) datasets.

It is cautiously postulated in this paper that the simplest state of all of the infinite states that might describe any 2D data set is the very rare occurrence in which the data set can be fully defensibly described as nested triangles only (with two, one or zero leftover interior-most points). Occam's razor and abductive reasoning aside, a certain irony is claimed to be therein present: the simplest state requires the most number of calculations to arrive at the (nested convex hull) solution.

The above, together with a desire to practice newly acquired python scripting skills, was enough to inspire a hobbyist to write an algorithm.

Early work: Literature

The use of nested convex hulls is to solve TSP problems is not new. Figure 1 is extracted from Ref. 8.

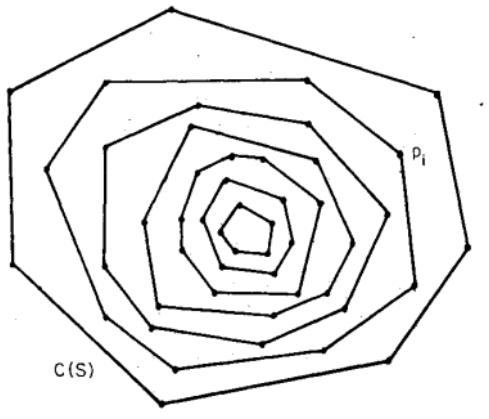


Fig. 1. Convex layers of point set.

Figure 1 Convex Layers of Chazelle point set (Ref. 8).

Work has been accomplished regarding optimized merge criteria (References 4 and 6 through 11).

Cronin's studies of nested hull "Outside-in" vs "Inside-out" traversals research indicated the "Outside-in" and the "Inside-out" approaches are equally valid and can produce the same path.

The author of this paper might caveat Cronin's finding as being dependent upon merge criteria. However, neither this authors nor (speaking most politely) Cronin's assertation is of much consequence to this paper. Reference to previous work is provided herein not just because credit is due but also because it reminds this author to keep in mind how much further the referenced previous works might of proceeded if computer power and programming tools of today had been available.

The work of Cronin and others will be thankfully highlighted again, under the brightest light the Author knows how to shine, in the Conclusions and Future Work section of this paper.

Early work: Motivation

Motivation included, as mentioned above, curiosity and desire to advance and solidify newly learned python programming skills. Additionally, and of greater relevance, prior to build there did exist a consideration that a successful nested triangles solver using inner-to-outer consumption approach might have a degree of universal application to nested polygon TSP solutions: more clearly, there was hope of applicability to the TSP solution to other types of 2D data sets (which, as already stated, exist with unfathomably times greater frequency).

Figures 2a and 2b below are provided for visual assistance. Obviously the plotted dataset in Figure 2a is an "engineered" data set (was not randomly generated). Figure 2b displays the "initial-nested-convex-hull" state for the provided data points with a total of two (2) "left over" innermost points.

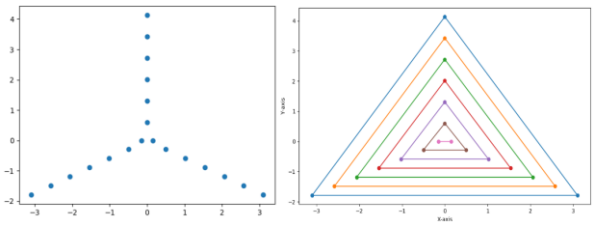


Figure 2 Engineered data set for input to nested convex hull processor plus convex hull nested triangles
Figure 2a: Scatter plot (Engineered)
Figure 2b: Convex hulls of the nest triangles.

Early work: 2D Code, Nested Triangles

The solution steps of the nested triangles solver basically involves merging of inner layer points into the next outer layer, based on steady and consistent shortest distance merge criteria, until all the points that composed the initial inner layer have been consumed by the next outer layer. Then the definition of inner layer is reset and the process begins anew and repeats until a singular traverse path (TSP valid or no) is defined.

Figures 3a and 3b depict the first two sequential steps in the path construct process.

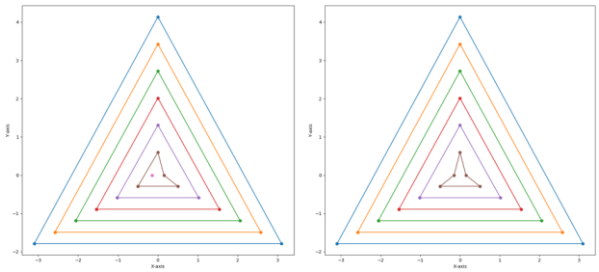
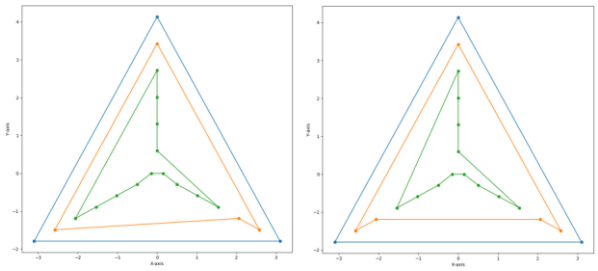


Figure 3a & 3b Engineered data set, Nested convex triangles, first two (2) merges.

Figures 4a and 4b depict two later sequential steps in the path construct process.



Figures 4a & 4b Engineered data set, partial solution (two (2) consecutive merges)

Figures 5a and 5b depict two unique final solutions. Figure 5a depicts the solution for rigid "shortest distance" merge criteria. Figure 5b depicts the solution obtained by early experimentation of merge decision-making flexibility using an included "angle criteria". The rigid (Figure 5a) solution provides shortest (and likely optimized) path.

It is speculated that this is a unique anomaly and possibly not unique to all nested triangle data sets but perhaps only to (nearly balanced, highly symmetrical) engineered datasets of this type. Later work, included herein, shows the opposite: the introduction of flexibility into merge decision-making can provide for shorter paths over the highly rigid "shortest delta distance" criteria.

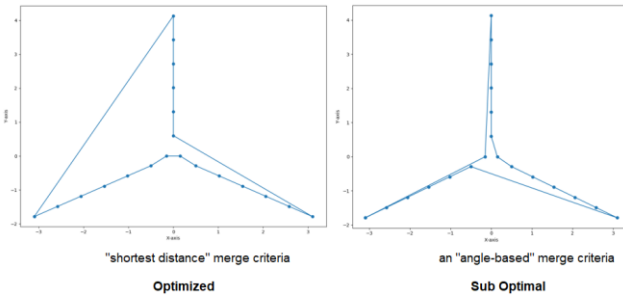


Figure 5a & 5b Engineered Data set, two solutions based on input parameters. The plot on the left (5a) is the shorter path.

Early work: 2D Code, Nested Triangles, Findings

Subsequently , the nested triangle solver was used for more complex nested 2D polygon datasets. For each data set the nested convex hull polygon system was created, the solver was applied and a solution was derived. Generally speaking, encouraging output was generated.

The following (and preceding) Figures are provided for purpose of qualitative evaluation only. Results are not compared to the available prior work for the public domain data sets: final solution paths are not included herein. Such is not the intent of this portion of the paper. The plots do provide convincing representation for supporting the use of the 2D nested triangles solver as a potentially worthy "core" solver for 2D datasets.

P01 (ref 12) data set scatter plot:

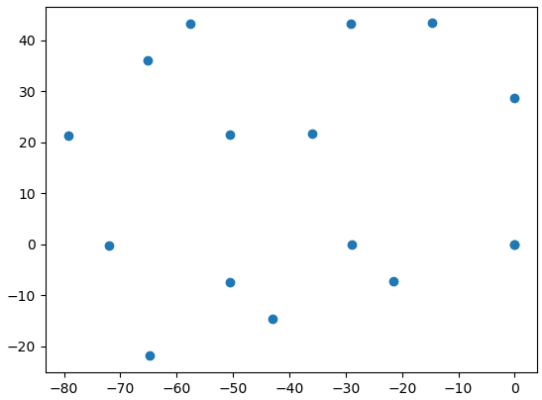


Figure 6 Data set for input to nested convex hull processor, P01, ref 12

P01 data set (ref 12) defined as nested convex hulls:

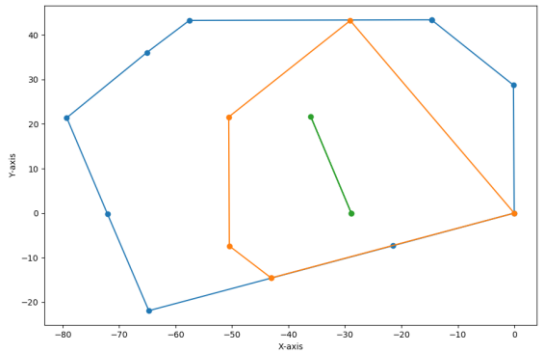


Figure 7 Data set P01 "processed" into nested convex hulls, note hull coincidence for four (4) data points

P01 data set solution obtained using rigid nested triangles solver:

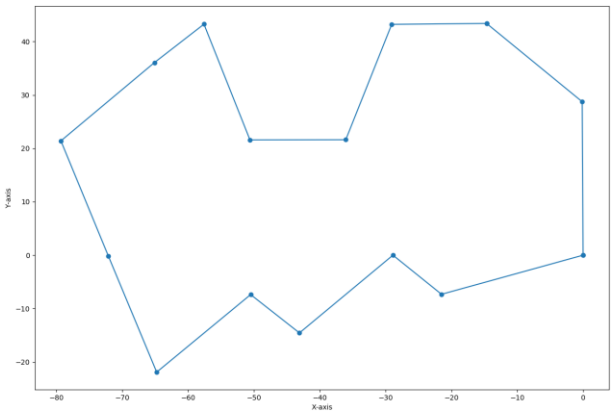


Figure 8 Data set P01 solution, ref 12

Western Sahara data set (ref 13) defined as nested convex hulls, plus solution obtained using rigid nested triangles solver:

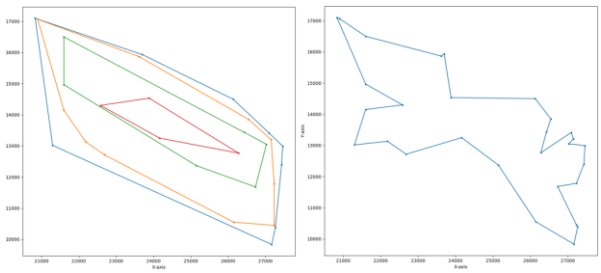


Figure 9 Data set for input to nested hull processor, Western Sahara, ref 13

ATT48 data set (ref 14) scatter plot:

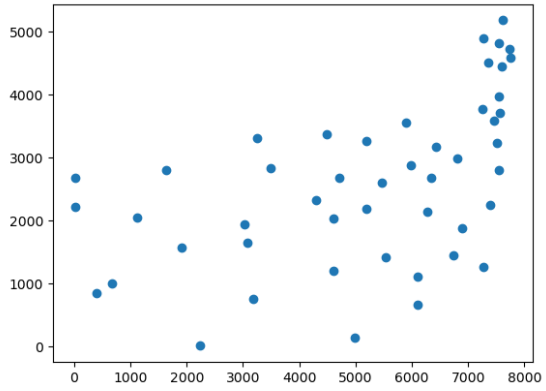


Figure 10 Data set for input to nested convex hull processor, ATT48, ref 14

ATT48 data set (ref 14) defined as convex hulls and final solution obtained using rigid nested triangles solver:

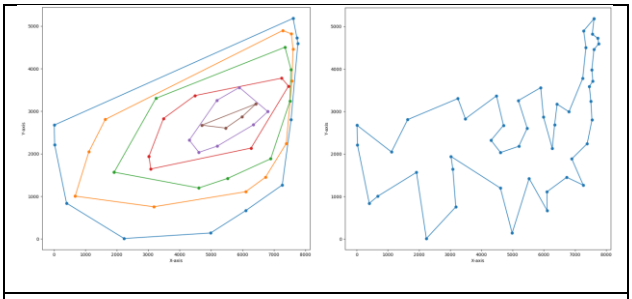


Figure 11 Data set ATT48 "processed" into nested convex hulls for input to Tri solver, plus solution path, ref 14

Qatar data set (ref 15) scatter plot:

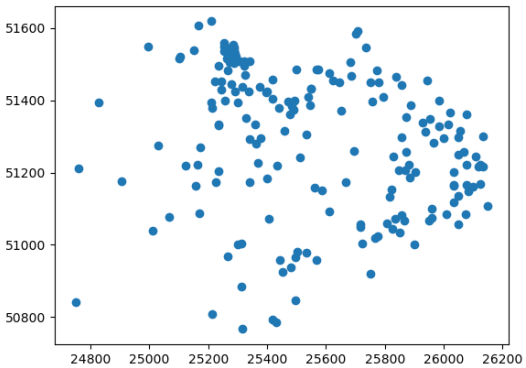


Figure 12 Data set for input to nested convex hull processor, QA194, ref 15

Qatar data set (ref 15) defined as convex hulls and final solution obtained using rigid nested triangles solver (Note: Solution is not TSP valid):

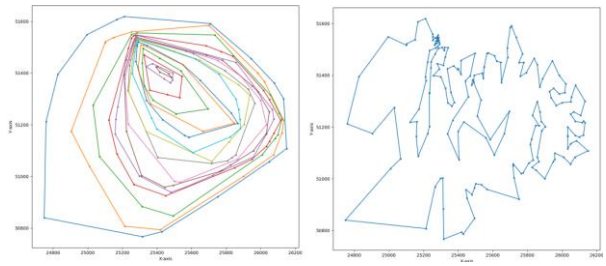


Figure 13 Data set QA194 "processed" into nested convex hulls for input to Tri solver, plus solution path, ref 15

XQF131 data set (ref 16) final solution obtained using rigid nested triangles solver (solution is not TSP valid):

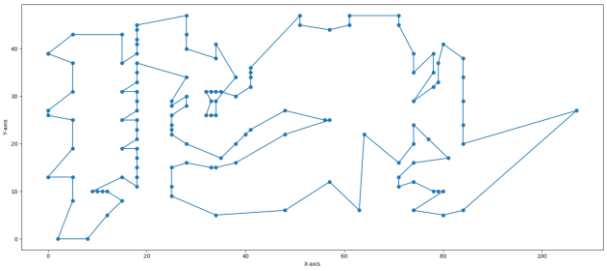


Figure 14 Final TSP solution, XQF131, Triangle solver, merge criteria = d2mc, ref 16

XQF final (valid and optimized) solution per ref 16 (solution not obtained by this Authors code):

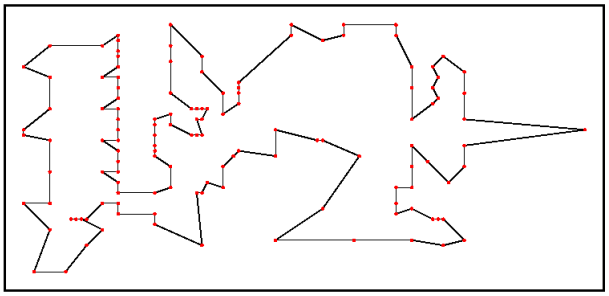


Figure 15 Final TSP solution, XQF131, FSU (credit copyright), ref 16

Again, the above work, using public domain datasets with known solutions, is presented for purpose of qualitatively conveying capability and robustness of the 2D nested triangles solver, mostly using inflexible shortest delta distance merge criteria. Intentionally, no quantitative comparisons are made to the public domain solutions. The XQF131 (ref 16) data set will not be further addressed because it likely requires additional merge criteria beyond the scope of this paper.

It is appropriate to attempt improvement of the 2D nested triangles solver as it does indeed falter with either non-optimal solutions and/or "crossovers" when processing larger and/or more complex datasets. The nested 2D triangles solver was verified as a reasonable choice for investigating possible benefits of complexity induction quantification and control.

Follow-on Work: Scope

The concept of a feedback system for the 2D solver was introduced. A feedback system was designed and placed into the 2D nested triangles solver algorithm.

Follow-on Work: Motivation

A means for increasing the accuracy of the 2D nested triangles solver, regardless of types (number of nodes) of nested polygons, was sought. The need for parameters for driving the feedback loop became evident, as well as a discontinue or STOP Loop condition.

Follow-on Work: Concept of Introduction of Complexity Induction into System: Early

Merge decisions that conveniently meet simple and rigid criteria allowed for development of a working nested triangles inner-to-outer layer TSP solving algorithm. The algorithm works unexpectedly well for more complex polygon nested hulls. However, it was postulated that the inflexible minimum delta distance merge criteria used by said algorithm for solving larger 2D datasets can introduce unwanted, unnecessary, and deleterious complications into the system. Also, when valid solutions are achieved the solutions are often sub-optimal.

The significance of aforementioned complications might be diminished when solution progress reaches a step which requires resolution of the earlier introduced complication. However, it was postulated that it is far more likely the decided upon resolution will introduce further complexity into the system. The thinking then was that complication(s) introduced early into the system as a consequence of misguided merge decisions should be minimized. This theory was conceived without justification beyond abstract reasoning. This theory of applying minimized complexity induction criteria to each marge decision was later revised.

Follow-on Work: Concept of Introduction of Complexity Induction into System: Later

The following discussion dwells completely in the abstract realm, with no supporting mathematical treatise.

If one views or defines any given 2D data set as having three (3) and only three distinct lowest complexity states:

a) scatter plot state
b) convex hull state
c) shortest Hamiltonian path state

then one might surmise that any in between "reorder" states (steps) are accompanied by introduction of complexity into the system.

An axiom is proposed: Once a scatter plot has been "redefined" as its convex hull(s) then any and all complexity accumulated during the "reorder" process is eliminated and lowest state is once again achieved. Another axiom is proposed: Once (a) convex hull(s) has (have) been redefined as shortest Hamiltonian path state then any and all complexity accumulated during the "reorder" process is eliminated and lowest state is again achieved. Consequently each and every step in the "reorder" process can now be considered as complexity inducing.

Complexity induction happens during every move (merge). Complexity induction is necessary and inevitable if re-ordering is to occur. The questions then arises: Is it worthwhile to attempt to quantify and manipulate complexity induction.

Somewhat traditionally, one might be inclined to impose minimized complexity induction for each merge step when deriving shortest path. Results contained herein do, to some extent, support this approach.

However, the above proffered axiom for shedding most accumulated complexity upon "reorder" to a TSP valid path, or indeed complete shedding thereof upon "reorder" to the absolute shortest valid path, forces or at least invites rethink. An understood and controlled complexity induction approach might be able to envision and identify "for the greater good" moments. Then, perhaps, a higher complexity induction merge would be permitted to prevail over localized minimized complexity induction. The proper balance of local and global prioritization, and timing thereof, might, if all is properly and sufficiently surrendered to the machine, be discovered by the machine and shorter valid paths might be found.

(As an aside and if not already implicit, and apologetically dwelling for one more moment in the abstract, should the solution be TSP non-valid then the system retains its total induced complexity after path closure and is in a highly complex, arguably low desirability state.)

It is hoped that the above scripting describes the scope of the expectations and responsibilities that might be placed on the proposed (or any) complexity induction feedback system.

The Ref. 11 paper was, loosely, a confidence assist resource for this Author during development of the above described complexity induction theory.

Follow-on Work: Concept of Introduction of Complexity Induction into System: Quantification

The concept of complexity induction can free an algorithm from the clutches of shortest merge criteria. This has been proven and is later documented herein. The concept of the 2D dataset system eventually, upon success, shedding most or all accumulated complexity induction (depending upon degree of success) allows the algorithm to make significant sacrifice(s) in exchange for offsetting later gain(s). From a complexity induction perspective accumulation is of minimal consequence if success results in shedding thereof.

This theory update pushing controlled complexity beyond simple minimized induction criteria for each and every merge step furthers the need for learning and for the facilitating feedback loop. A truly learn capable machine would be needed to successfully explore the concept. Firstly though, the need has become apparent to quantity complexity induction from the discrete perspective of actualized merges.

Follow-on Work: Concept of Introduction of Complexity Induction into System: Quantification: "Switchbacks" / "Folds"

One such "complexity introducer" is proposed as any merge decision which introduces a "switchback" or a "fold" into the system. The following figures are assistive in understanding "switchbacks".

Figure 16 below displays a merge scenario that introduces little or near immeasurable complexity induction into the system.

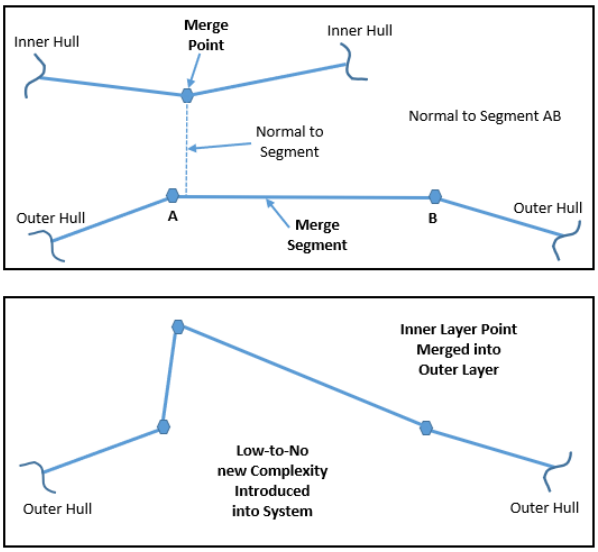


Figure 16 Merge decision does introduces very little complexity:
(a) Depiction of "before merge"
(b) Depiction of "after merge"

Figure 17 below displays a merge scenario that introduces quantifiable complexity induction into the system. Figure 17 depicts the formation of "switchbacks" or "folds" into the system.

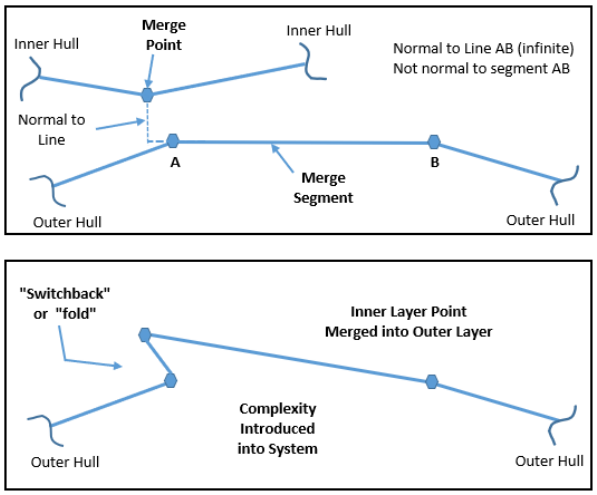


Figure 17 Merge decision introduces notable complexity, a "fold-Over" or a "switchback":
(a) Depiction of "before merge"
(b) Depiction of "after merge"

Follow-on Work: Concept of Introduction of Complexity Induction into System: Complexity Induction as a Function of Region

Merge decisions may readily be grouped into four categories based on physical regions. This categorization is assistive in complexity induction quantification (and code writing).

Regions A, B and C

The following figures depict Region A, B and C merge decisions.

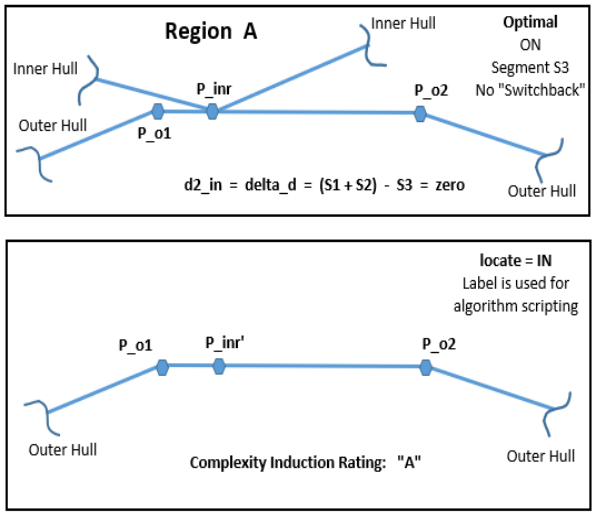


Figure 18 Complexity Induction Region A

Region B merge decisions:

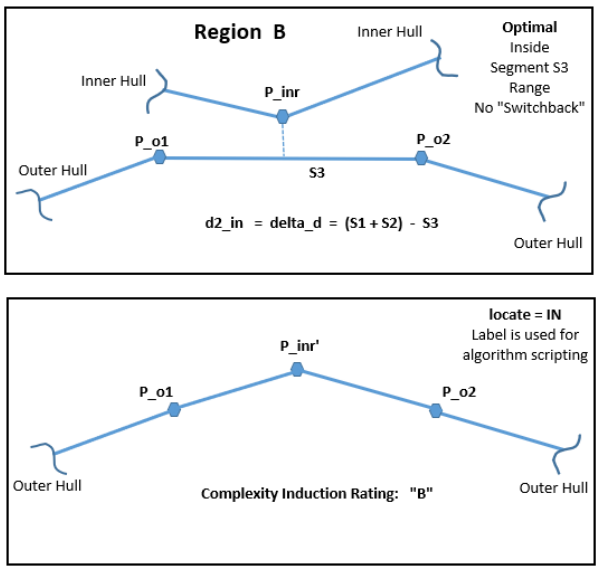


Figure 19 Complexity Induction Region B

Region C merge decisions:

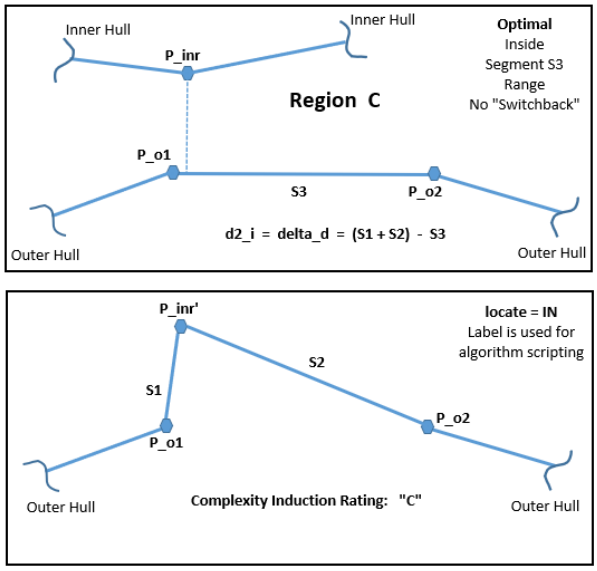


Figure 20 Complexity Induction Region C

Region D

The following two figures depict Region D merge decisions. Region D merge decisions, such as the one depicted in Figure 21, induct (incur) significant complexity into the system.

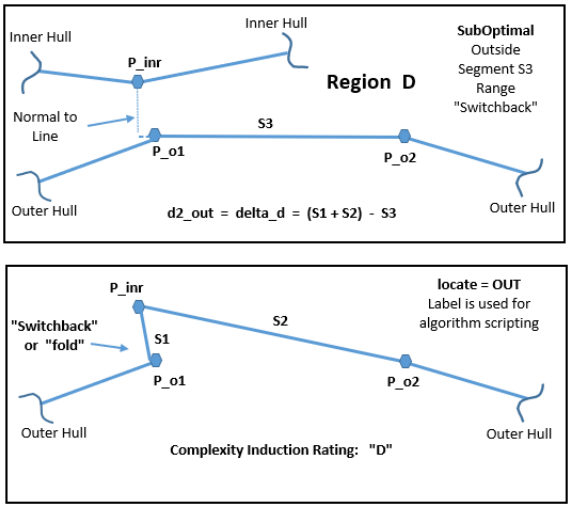


Figure 21 Complexity Induction Region D(i)

A system incurs even greater complexity induction for the following Region D merge:

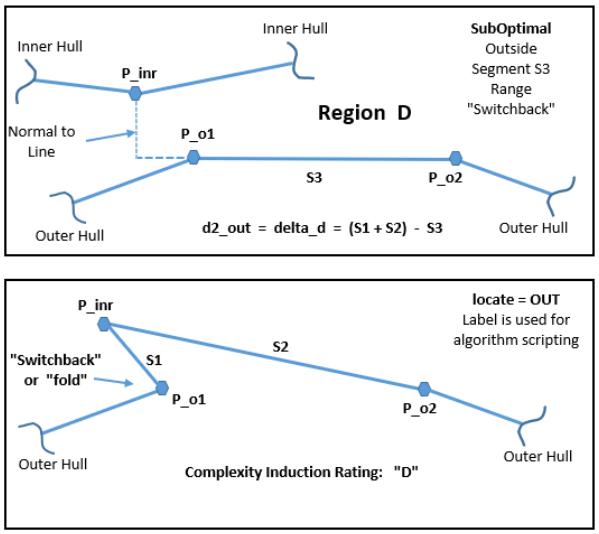


Figure 22 Complexity Induction Region D(ii)

Follow-on Work: Complexity Induction Factor (ci_f), Concept

A more sophisticated merge decision criteria beyond change in distance (delta_d) is needed to control switchback creation. A complexity Induction factor is needed. A simplistic function or plot of complexity induction factor vs a simple defining dimension is needed.

It is forecast that the desired function curve will have the following approximate overall shape:

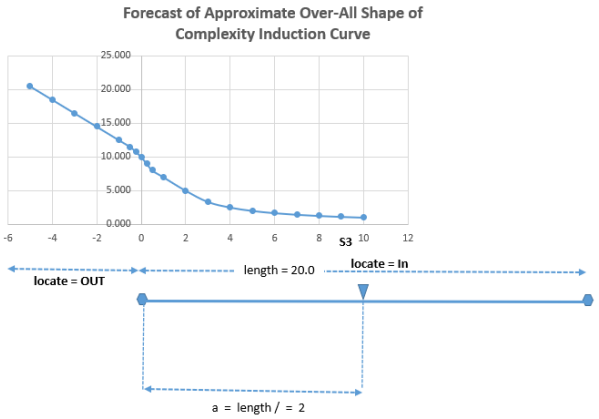


Figure 23 Factor f, induction complexity, forecast curve

Follow-on Work: Complexity Induction Factor (f_ci), Derivation

The actual derivation of the plot (function) is of lesser importance than the overall shape of the plot. This is because the loop feedback parameters, not yet discussed, will provide the needed additional manipulations. However, the chances of a "good" run/solution increase the closer the complexity induction curve is to "best" curve. Also, any over-reliance on the not yet discussed feedback parameters to provide the necessary compensation is likely not complimentary.

Figure 24 explains derived parameters for Regions A, B and C.

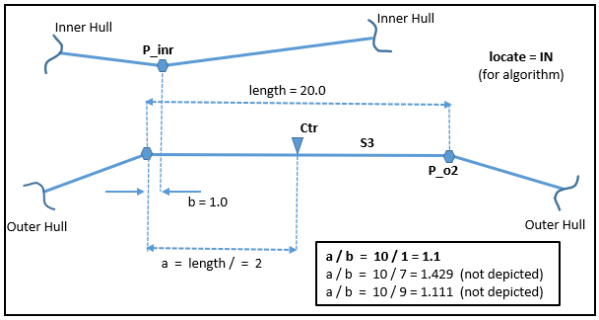


Figure 24 Factor f, (f_ci) Induction Complexity Curve, Regions A, B and C

Figure 25 explains derived parameters for RegionS.

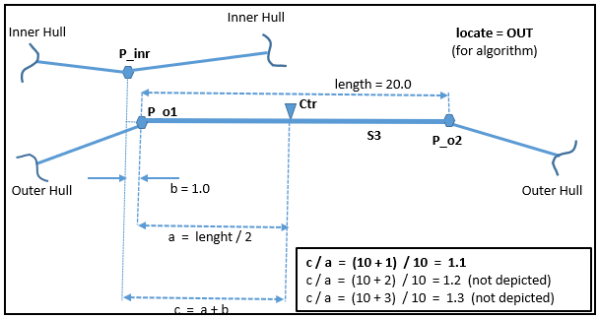


Figure 25 Factor f, (f_ci) Induction Complexity Curve, Region D

Figure 26 depicts projected plot of Regions B, C and D induction complexity curve.

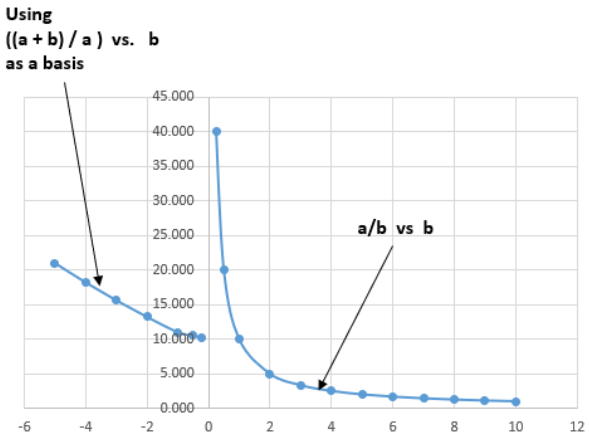


Figure 26 Factor f, (f_ci) Induction Complexity Curve, Preliminary

Figure 27 depicts a proposed mathematical definition of Regions B, C and D induction complexity curve.

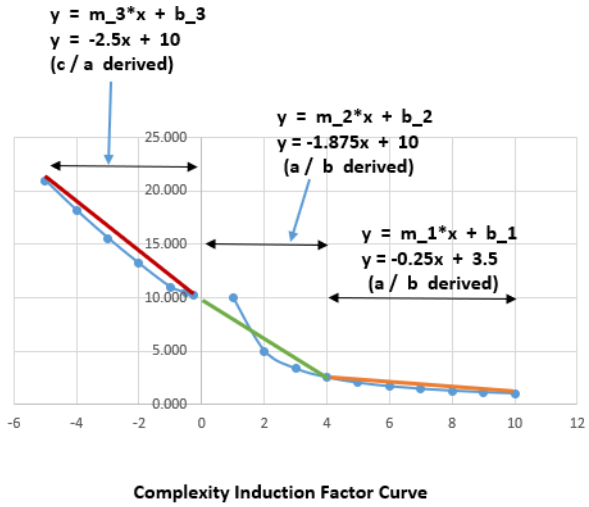


Figure 27 Factor f, (f_ci) Induction Complexity Curve, Final

The equations shown in Figure 505 provided a consistent basis for the following exploratory work. These equations will not be used beyond the scope of this section of this paper: namely, "Follow-On Work".

Follow-on Work: Complexity Induction Factor (f_ci) Application [times delta_d (d2)]
A table was populated using inner layer point location (x, y), delta_distance (d2) and using d2 times f_ci. The f_ci factor was derived using the f_ci curve. Notable plots and findings are shown below.

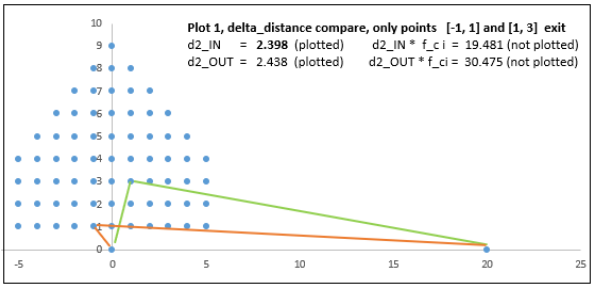


Figure 28 Delta_d (d2_IN, d2_OUT)
Finding: The f_ci factor ensures "IN" locate merge decision is made.

Additional findings follow:

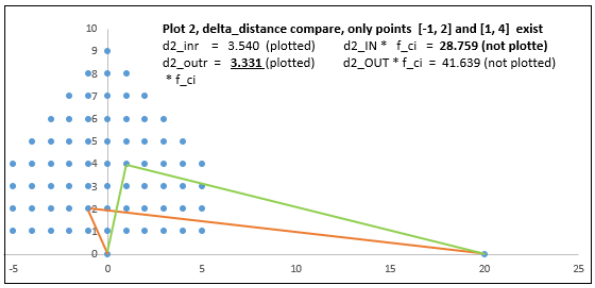


Figure 29 Delta_d (d2_IN, d2_OUT)
Finding: The f_ci factor ensures "IN" locate merge decision is made. The "OUT" merge provides very minimal distance advantage and is complexity induction expensive. Note that a simple delta_distance criteria (d2 alone) would of selected the "OUT merge and created a "switchback" with a suboptimal (6.27%) distance traveled benefit.

Additional findings follow:

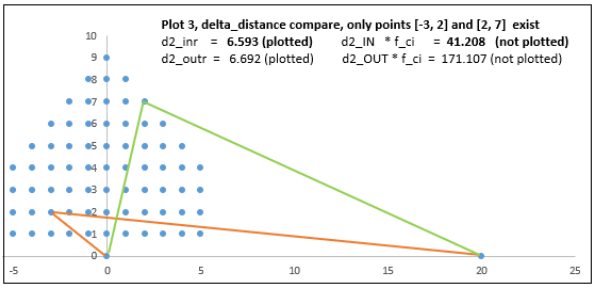


Figure 30 Delta_d (d2_IN, d2_OUT)
Finding: The f_ci factor ensures "IN" locate merge decision is made.

Additional findings follow:

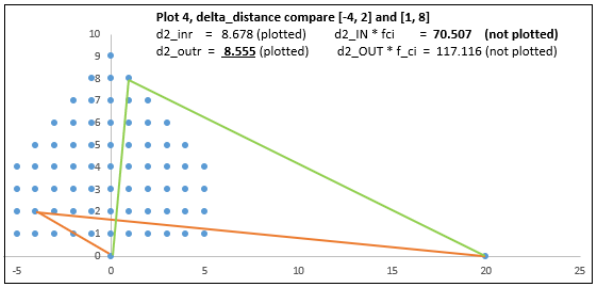


Figure 31 Delta_d (d2_IN, d2_OUT)
Finding: The f_ci factor ensures "IN" locate merge decision is made. The "OUT" merge provides very minimal distance advantage and is complexity induction expensive. Again, note that the simplistic delta_distance criteria (d2 alone) would of selected the "OUT merge and created a "switchback" with near-zero distance traveled gain.

Follow-on Work: Angle Severity Complexity Induction Factor (asci_f), Concept

The concept of "switchback" angle severity is herein introduced. This factor provides for another means of quantifying and controlling complexity induction.

Figures 32 and 33 are assistive.

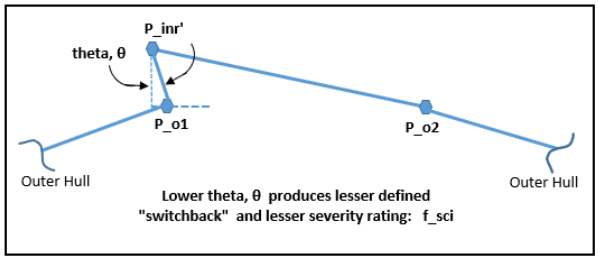


Figure 32 Lower theta, θ, produces lesser complexity "switchback" or "fold" and lesser severity rating: f_sci

A higher severity "switchback" or "fold", consequent of a higher theta angle, is shown in Figure 33.

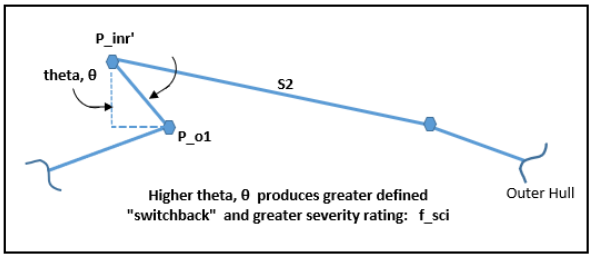


Figure 33 Higher theta, θ, produces greater complexity "switchback" or "fold" and a correspondingly greater severity rating: f_sci

A severity rating system was developed based on angle theta for Region D merge proposals. For various reasons angle severity was not used in this paper and so derivation of the system is not explained. The author considered not including the theta angle merge criteria topic in this paper. However, Figure 5b (6 nested triangles) and Figure 14 (ref. 16) traverse paths were both created using variants of the theta angle approach.

3. CURRENT WORK

Current Work: AUC (concept of area under the curve as a useful observation/monitoring Tool)

Figure 34 below shows the originally intended means of plotting data for each solution. It was hoped that such plots might lend themselves well to enhancing learn rate of the feedback loop.

It is a plot of accumulative delta distance vs accumulative complexity for randomly generated x and y data pairs. Fictitious data was used to populate an associated table (not shown herein). It is theorized that the AUC (area under the curve) might be representative of qualitative goodness of the solution path.

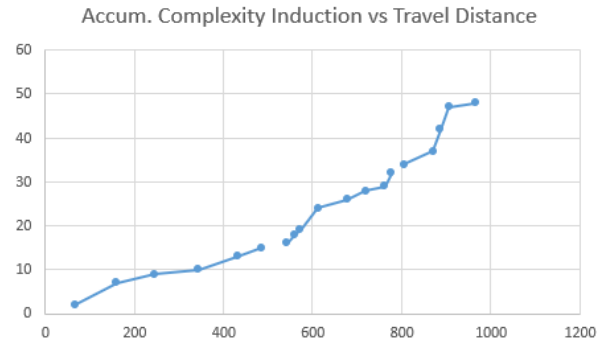


Figure 34 Accumulative Complexity Induction vs Travel Distance.

Code snippet follows:

```
model = np.polyfit(x, y, 1)

model

array([ 18.38942308, 158.49595142])

predict = np.poly1d(model)

kilomMark = 20
predict(kilomMark)

526.2844129554655

from sklearn.metrics import r2_score
r2_score(y, predict(x))

0.9241777204309907
```

Figure 35 Snippet, linear regression Code.

The linear regression plot follows:

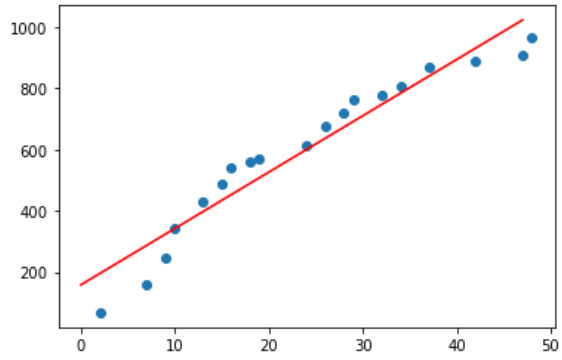


Figure 36 Linear regression plot.

First order resultants include total path length, total complexity induction and total delta_L accumulations. Various plots might be subsequently generated such as travel distance vs accumulating complexity induction. Second order affects such as area under the best fit line (AUC) might be extracted.

There might be hopeful expectation of decrease in second order affect (e.g., AUC) as the algorithm feedback system allows for selection of lesser complexity induction and better sequenced merges and also as the total travel distance lessens.

Figure 37 below is provided for visual assistance.

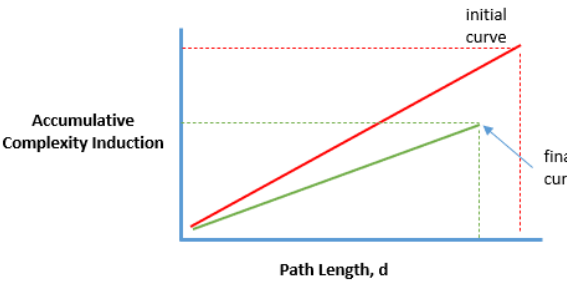


Figure 37 Improved area under the cruve plot.

However, it was later discovered that the underlying onion layer method precludes ready alignment between the "x" and the "y" axis. Neither axis is relatable directly to visited path points.

For convex hull solutions both the x-axis can only be relative to order of merge decision. For example, plot point 22 indicates accumulative complexity induction (y-axis) and accumulative delta_L (x-axis) upon actualization of merge event number 22. The Ref. 14 (att48) data set is composed of 48 cities while 96 merge decisions are required to arrive at the convex hulls layer solution. The Author was unable to accomplish suitable alignment. Perhaps the Reader will see a means.

The germane preceding plots were not removed from this report for three reasons. One, the Reader might provide insight. Two, they convey evolution of thought. Three, because for other TSP solution methods (not nested convex hulls based), the correlation with path build might be more easily established.

Current Work: Minimized Complexity Induction Criteria Derived Feedback Parameters

The preceding work proves what, in retrospect, is rather obvious and well known: modification of influential factors results in change. Of less certainty is the effect on including these control factors into subject solution logic. The preceding work led to a move forward decision for attempting to determine and define a suitable feedback parameter approach for augmentation of the nested triangles derived convex hull solver.

To facilitate this decision-making process the induced complexity cost of each candidate merge must also be known (as well as the easily calculated and traditionally well-established visited point (city) gain).

The nested triangle inner-to-outer layer consumption code, with its unyielding delta distance merge criteria, was revised to incorporate findings of the preceding. In so doing, it was decided that the previous line curves used to explore the effects and possible benefits of manipulation of complexity induction, best be viewed as and relegated to status "conceptual". Those equations were discarded for remainder of this paper's investigation.

More simplistic factors for complexity induction were applied. As indicated in preceding plots the segment was fractionalized into 0.1 segment fractions based upon center point of the segment. The angle parameter, only applicable to Region D merge points, was nullified to unity and to "travel-along-only" status in the code for the purposes of remainder of this study. This was due to concern that Delta distance combined with Region D specificity could create a control that duplicates the Region D angle control. Problematic over-constraint could be

introduced into the system but might not be readily apparent or easily detectible.

Current Work: Qatar, 13 runs, inputs

The code was rewritten to incorporate complexity induction factor(s). All results presented in this paper were generated using manual input changes to parameters. The algorithm was constructed with intent of automating the parameter feedback loop and it is expected that the needed update would be neither overly complex or time consuming. Such an algorithm would, however, require higher power computing capability. Also, verification of path validity, no revisits and no cross-overs, is, at this time a manual (human eye) process.

Figure 38 below shows Regions B, C and D zonal parameters definition.

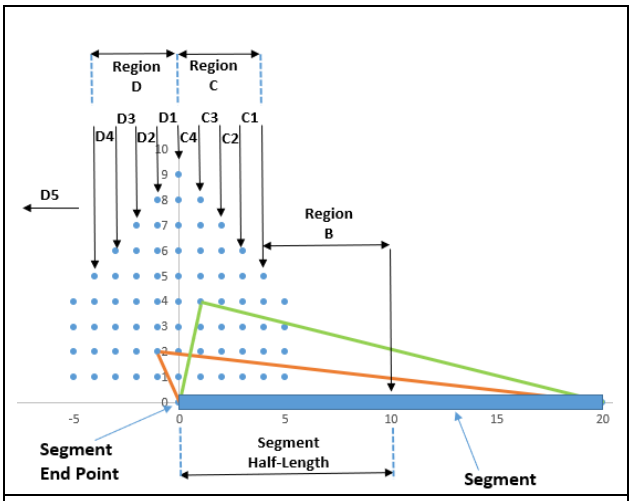


Figure 38 Algorithm input parameters

Regions C4 and D1 are closest to the segment end point. Regions C1 and D4 and D5 are furthest from the segment end point.

Based upon initial use of the new algorithm it was concluded that it might be easier to detect benefits of complexity induction criteria using more complex (and more highly populated) datasets. The ref. 15 (Qatar) data set was run.

Table 1 below provides correlation of input parameters to the thirteen (13) total Qatar (ref 15) runs.

RUN	A	B	REGION C				REGION D				
			C1	C2	C3	C4	D1	D2	D3	D4	D5
1	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	0.00	1.00	1.00	1.00	1.00	1.00	0.90	0.90	0.90	0.90	0.90
3	0.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00	1.00	0.90	0.90
4	0.00	1.00	1.00	1.00	1.00	1.00	0.90	0.90	1.00	0.90	0.90
5	0.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00	1.00	1.00	0.90
6	0.00	1.00	1.00	1.00	1.00	1.00	0.90	0.90	1.00	1.00	0.90
11	0.00	1.00	1.10	1.10	1.10	1.10	0.90	0.90	0.90	0.90	0.90
7	0.00	1.00	1.00	1.00	1.00	1.00	0.85	0.85	0.85	0.85	0.85
8	0.00	1.00	1.00	1.00	1.00	1.00	0.80	0.80	1.00	0.80	0.80
9	0.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00	1.00	1.00	1.00
10	0.00	1.00	1.00	1.00	1.00	1.00	0.95	0.95	0.95	0.95	0.95
12	0.00	1.00	1.15	1.15	1.15	1.15	0.90	0.90	0.90	0.90	0.90
13	0.00	1.00	1.10	1.10	1.10	1.10	0.95	0.95	0.95	0.95	0.95

fl 1 Grey cells signify no variance of inputs from either zero or unity.

Constrain region D parameters to unity or less.
Constrain Region B and C parameters to unity or greater.
Otherwise there will be complications with AUC calculations.

Table 1 Input parameters, Runs 1 – 13, Ref 15 dataset (Qatar).

The following lesson was learned regarding AUC calculation and is clumsily summarized as follows: Any A, B, C, D (or angle) factor that might serve to induce (incentivize or obviate) complexity induction, namely in the form of "switchbacks" or "folds", ought be assigned accordingly. For this reason it is best that factors for Regions B and C be 1.0 or greater. Similarly, the factors for Region D ought be unity or less.

The above can be blended in satisfactorily with the previously presented complexity induction theory, but such discussion is non-essential. Now is an opportune moment to differentiate between algorithm input parameters (see Figure 38 and Table 1) and complexity induction factors. The difference is simple while the explanation is somewhat lengthy.

Otherwise and additionally, use of input parameters less than unity can and likely will create some confusion when these parameters are used for AUC (area under the curve) calculations. For AUC calculations contained herein the Region D factors are first inverted before AUC calculation inclusion while the Regions B and C factors are used in unmodified state (not inverted). (Similar considerations ought be applied to the angle factor of Region D should the occasion arrive that it is made participant.) This methodology serves two goals. One, to be able to readily increase or reduce tendency for Region D merges. To assure that all Region B, C, and D merges cause suitable complexity induction.

Figure 39 is simply an assistive script screen-capture of the code:

```
            f_asci = 1.0
#   ***************************************************
            if (Where == "IN_A"):       # in Region IN_A
                merge = rA
#   ***************************************************
            elif (Where == "IN_B"):     # in Region IN_B
                merge = delta_L * rB
                r_fac = rB
#   ***************************************************
            elif (Where == "IN_C1"):    # in Region IN_C
                merge = delta_L * rC1
                r_fac = rC1
#   ***************************************************
            elif (Where == "IN_C2"):    # in Region IN_C
                merge = delta_L * rC2
                r_fac = rC2
#   ***************************************************
            elif (Where == "IN_C3"):    # in Region IN_C
                merge = delta_L * rC3
                r_fac = rC3
#   ***************************************************
            elif (Where == "IN_C4"):    # in Region IN_C
                merge = delta_L * rC4
                r_fac = rC4
#   ***************************************************
            elif (Where == "OUT_D1"):   # in Region IN_D1
                merge = delta_L * rD1 * f_asci
                r_fac = rD1
#   ***************************************************
            elif (Where == "OUT_D2"):   # in Region IN_D2
                merge = delta_L * rD2 * f_asci
                r_fac = rD2
#   ***************************************************
            elif (Where == "OUT_D3"):   # in Region IN_D3
                merge = delta_L * rD3 * f_asci
                r_fac = rD3
#   ***************************************************
            elif (Where == "OUT_D4"):   # in Region IN_D4
                merge = delta_L * rD4 * f_asci
                r_fac = rD4
#   ***************************************************
            else:     # Where == "OUT_D5" in Region IN_D5
                merge = delta_L * rD5 * f_asci
                r_fac = rD5
#   ***************************************************
```

Figure 39 Algorithm snippet
Run 1
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 1.00
No Cross-Overs, Valid TSP Solution
Path=21.339 units
Complexity Induction = 57, Delta_L = 5.79,
AUC = 169.95

Figure 40 is an assistive script screen-capture of the code for parameters input. (Run 2 parameters are depicted).

```
        else:
            # INSERT FEEDBACK LOGIC here
            rA  = 1.00
            rB  = 1.00
            rC1 = 1.00
            rC2 = 1.00
            rC3 = 1.00
            rC4 = 1.00
            rD1 = 0.90
            rD2 = 0.90
            rD3 = 0.90
            rD4 = 0.90
            rD5 = 0.90
            rD_angl = 65.0
```

Figure 40 Algorithm Region zonal input parameters

Figure 41 below is an assistive script screen-capture of the logic code for determination of complexity induction factor. Note the inversion of the "r_fac" for Region D merges. An "r_fac" of less than or equal to unity is conducive to (incentivizes) Region D merges. The inverse is the associated complexity induction, which is always equal to or greater than unity.

```
#   ********** r_Comp_Induct **********
        #r_Comp_Induct = 1 / r_fac
        if (Where == "OUT_D1"):
            r_Comp_Induct = 1 / r_fac
        elif (Where == "OUT_D2"):
            r_Comp_Induct = 1 / r_fac
        elif (Where == "OUT_D3"):
            r_Comp_Induct = 1 / r_fac
        elif (Where == "OUT_D4"):
            r_Comp_Induct = 1 / r_fac
        elif (Where == "OUT_D5"):
            r_Comp_Induct = 1 / r_fac
        else:
            r_Comp_Induct = r_fac
#   ***********************************
```

Figure 41 Algorithm Comp Induct factor

Current Work: 6 Triangles, 2 runs, Results
As previously stated, the code was rewritten (updated) and a total of thirteen (13) runs were made on the ref. 15 (Qatar) data set. To assist and to acclimate the Reader for review of those results, the smaller dataset runs of six (6) nested triangles are depicted first.

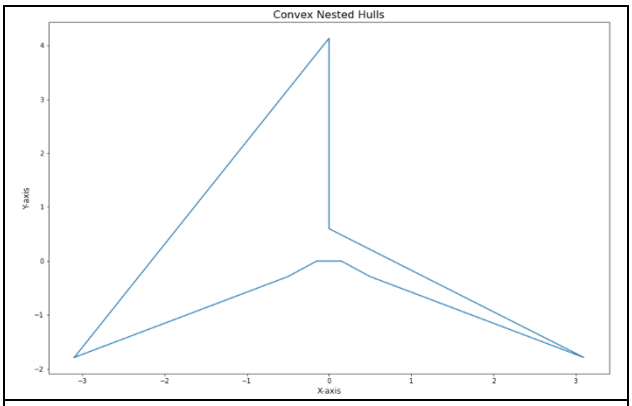


Figure 42 Six Nested Triangles, plus 2 datum

Figure 45a Run 1
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 1.00
No Cross-Overs, Valid TSP Solution
Path=21.339 units
Complexity Unduction = 57, Delta_L = 5.79,
AUC = 169.95

Figure 45b Run 2
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90
No Cross-Overs, Valid TSP Solution
Path = 21.339 units
Complexity Induction = 61, Delta_L = 5.79, AUC = 181.62

Note: The Figure is the same as Figure 5a.
It was not readily possible to replicate the triangle solution path shown in Figure 5b (angle parameter based) by using the distance fraction based "A", "B", "C" and "D" parameters.
The plot did not change until all "D" parameters were set to 0.25 or lower.

Current Work: Additional AUC Plots
6 Nested Triangles

Figure 43 plots the 57 data points representing each merge decision (57 total) made to create the Figure 5a and Figure 42 traverse path (plots are all the same).

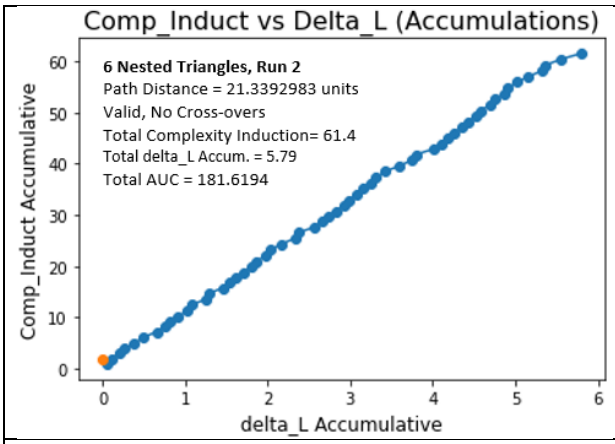


Figure 43 6 Nested Triangles, Run 2
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90
No Cross-Overs, Valid TSP Solution
Path = 21.339 units
Complexity Induction = 61, Delta_L = 5.79
AUC = 181.62
Check: ½ * 6 * 60 = 180

A total of 57 merge decisions / merge events were required for path completion while total number of points is 20.

Current Work: att48, 2 runs, Results
As previously stated, the code was rewritten (updated) and a total of thirteen (13) runs were made on the ref. 15 (Qatar) data set. To assist and to acclimate the Reader for review of those results, the smaller dataset runs of att48 (ref 14) are depicted next.

Run 1 (D = 1.0) results are shown in Figure 44.

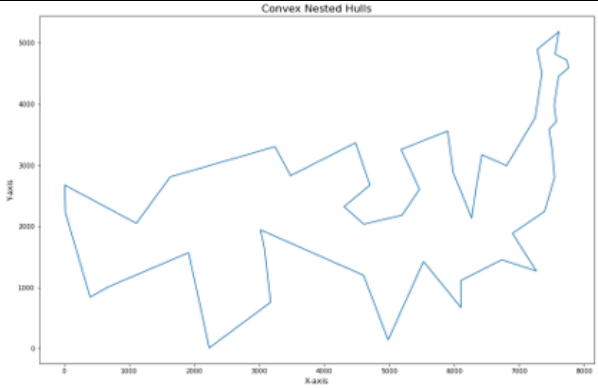


Figure 44 Dataset att48, Run 1
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 1.00
No Cross-Overs, Valid TSP Solution
Path = 34,273 units
Note: This is the same plot as Figure 11

Run 2 (D=0.90) results are shown in Figure 45.

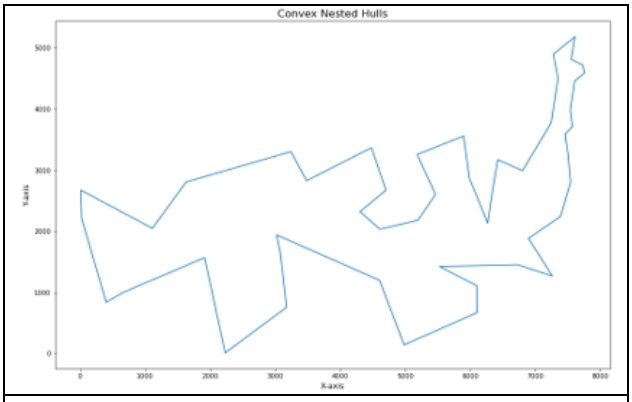


Figure 45 Dataset att48, Run 2
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5=0.90
No Cross-Overs, Valid TSP Solution
Path = 34,304 units

A shorter path results for the rigid "shortest delta length" solution (run 1, all parameters = 1.00). The very close 31 units of separation provide some hope that a shorter path solution is obtainable by a more thorough study of effects of manipulation of the input parameters. No further effort was dedicated to this specific campaign by the researcher.

Current Work: Additional AUC Plots Att48 (ref 103)

Figure 46 plots the 96 data points representing each merge decision (96 total) made to create the Figure 47 traverse path.

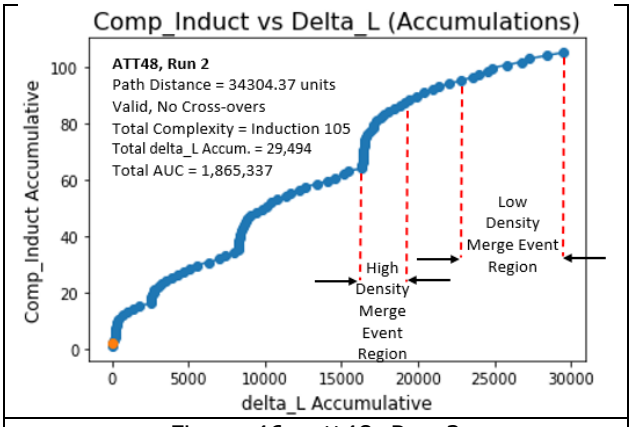


Figure 46 att48, Run 2
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5=0.90
No Crossovers, Valid TSP Solution
Path = 34,304 units
Complexity Induction = 105, Delta_L = 29,494
AUC = 1,865,337
Check: ½ * 29,494 * 105 = 1,548,435

A total of 96 merge decisions / merge events were required for path completion.

The above Figure 46 plot is similar in nature to the following Qatar (ref 15) and Canada (ref 17) plots. Note the densification of data point merge decisions (merge events) on the high slope portions of the curve compared to the lesser slope (plateau-like) portions. This densification will not be readily visuall detectibe on the corresponding Qatar (ref 15) and Canada (ref 17) plots.

Current Work: Qatar, 13 runs, Results

As previously stated, the code was rewritten (updated) and a total of thirteen (13) runs were made on the ref. 15 (Qatar) data set. Results are provided in Table 2.

RUN	CROSS OVERS	PATH LENGTH (units)	Total Complexity Induction	Total delta_L	AUC Valid ?	AUC AREA (units)	Path inmage shown
1	yes	10504.99436	1178.0000	35492.26811	no fl 1	23,800,320 (n/a) fl 2	I fl 1
2	no	**10399.37873**	1284.7778	35515.5012	**yes**	25,807,737	II
3	no	**10399.37873**	1279.7778	**35367.5321**	**yes**	**25,626,958**	II
4	no	**10399.37873**	1284.5556	35444.9404	**yes**	25,775,596	II
5	no	10399.95878	**1279.2222**	35383.97624	**yes**	25,638,346	III
6	no	10399.95878	1284.5556	35465.82192	**yes**	25,794,328	III
11	no	10431.58462	1288.8444	35515.50123	**yes**	26,011,792	IV
7	yes	10395.71500	1346.1765	35517.10448	no	27,038,095	
8	multi	10705.52103	1410.7500	36371.16668	no	29,084,629	*not*
9	yes	10504.99400	1178.7778	35266.59796	no	23,574,089	*included*
10	yes	10509.84200	1227.9474	35064.88536	no	24,472,819	
12	yes	10448.03865	1,289.8222	35632.99277	no	26,061,857	*not*
13	yes	10438.42558	1,232.6842	35345.56330	no	24,745,946	*included*

fl 1 Only shown since all parameters = 1.00.
(shortest delta distance merge criteria only.)
fl 2 Disregard all AUC values for runs with one (1) or more cross-overs. Runs with one or more cross-overs can produce lesser AUC values than runs with no cross-overs.
fl 3 It is likely that for a data set of this smaller size that same path distance equivalates to same actual path.
However, the differences in accumulations data indicated slightly merge decision logic was used.
fl 4 Essentially the non-validity of Run 1 and Runs 7 through 13 also invalidates all associated data.

Table 2 - Qatar dataset (ref 15) Results for 13 runs (manual manipulation of input parameters)

The Type I, II, III and IV path solutions referenced in Table 2 are depicted below:

The Type II path solution referenced in Table 2 is depicted below:

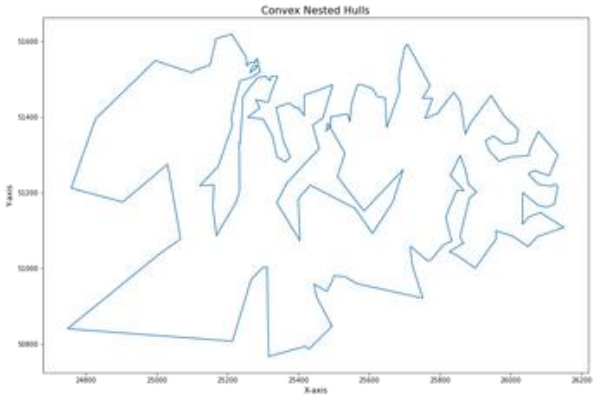


Figure 47 Plot Type I, Run 1
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 1.00
1 Cross-Over, Not Valid TSP Solution
Path = 10,504.99436 units
Note: This plot is the same as Figure 13.

The Type II path solution referenced in Table 2 is depicted below:

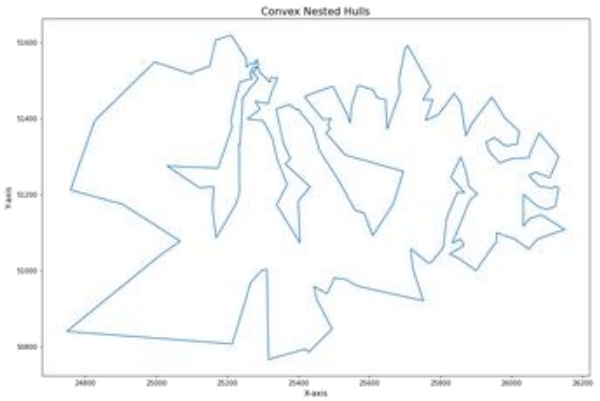


Figure 48 Plot Type II, Runs 2, 3, 4
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= see Table
(D = 0.90 for Run 2)
No Cross-Overs, Valid TSP Solution
Path = 10,399.37873 units

The Type III path solution referenced in Table 2 is depicted below:

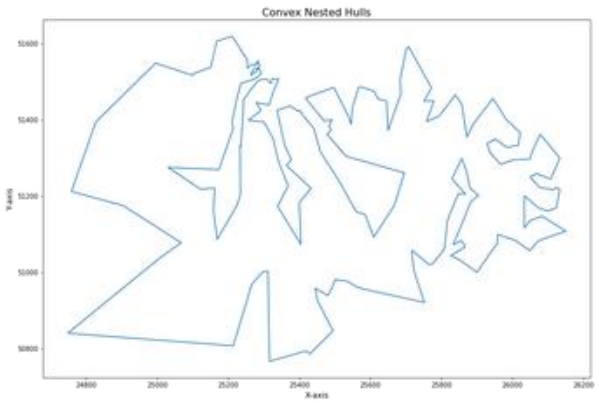


Figure 49 Plot Type III, Runs 5 and 6
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5 = see Table
No Cross-Overs, Valid TSP Solution
Path = 10,399.95878 units

The Type IV path solution referenced in Table 2 is depicted below:

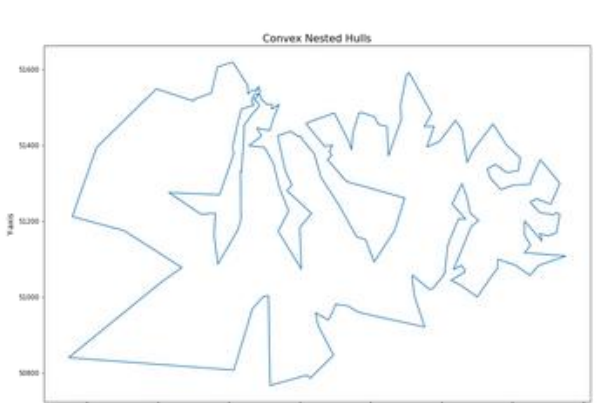


Figure 50 Plot Type IV, Run 11
A=0, B=C1=C2=C3=C4= 1.10
D1=D2=D3=D4=D5 = 0.90
No Cross-Overs, Valid TSP Solution
Path = 10,431.58462 units

Assistive zoom-ins of the Type I, II, III and IV solutions are depicted in Figure 51 below:

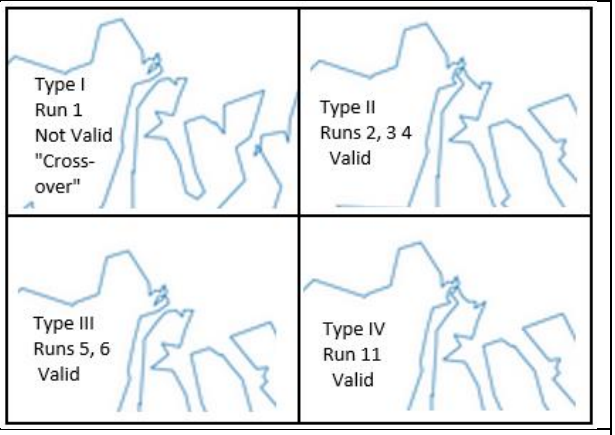


Figure 51 Visual assist zoom-in.

Again, further visual assist zoom-ins of the Type I, II, III and IV solutions are depicted below:

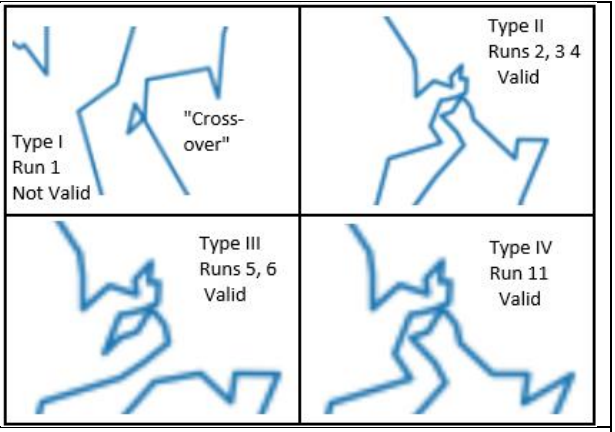


Figure 52 Visual assist increased zoom-in.

Current Work: Additional AUC Plots
Qatar

A total of 1,178 merge decisions / merge events were required for path completion of each Qatar dataset run.

Figure 53 plots the curve representing each merge decision (1178 total) made to create the Figure 50 traverse path.

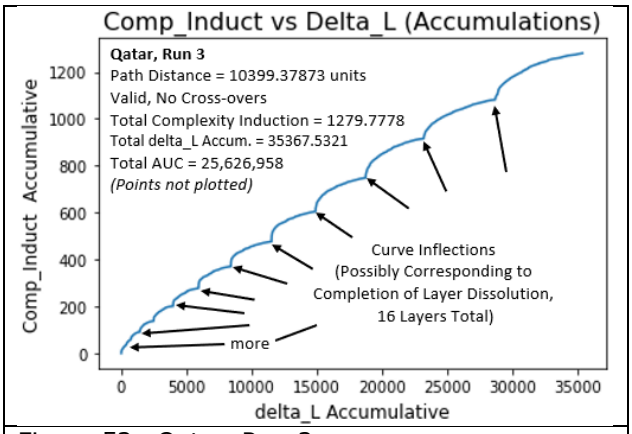


Figure 53 Qatar, Run 3

It is conjectured that the curve inflection points correspond to completion of dissolution of onion layers. The Canada dataset (ref 106) plots of progressive onion layer merging further strengthen this supposition.

Figure 54 plots the 1178 data points representing each merge decision (1178 total) made to create the Figure 50 traverse path.

Note that the densification of data point merge decisions (merge events) on the high slope portions of the curve compared to the lesser slope (plateau-like) portions readily observed in the Figure 46 (att48 dataset) plot are not detectible in this plot. This would hold true even if 2 out of 3 data points were omitted, because of the size of the dataset. The six nested triangles dataset plot and the att48 USA cities dataset plots were shown previously for Reader visual assist and acclimation.

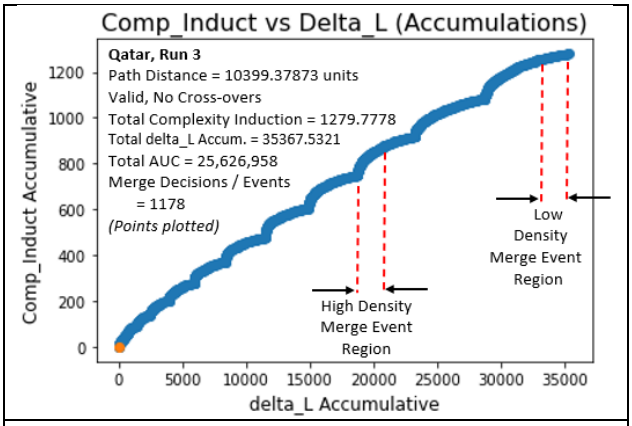


Figure 54 Qatar, Run 3

Check: ½ * 35,367 * 1279.8 = 22,631,343

Although obvious in retrospect, the following is pointed out by the Researcher: When all modification factors are equal to unity then the complexity induction accumulations (y-axis) will always be an integer and will always be equal to the total number of merge decisions made at that time in build create process. The total complexity induction for the Qatar (ref 104) data set would then be baseline represented by the integer 1178 for a run with all modification factors equal to unity.

Consequently, complexity induction for a run with modification factors greater than unity could be characterized, during and at completion of path construct, as a percentage of baseline for build progress. The Author is unsure if this calculated percentage should be considered a first order or a second order data analysis extract. The discussion or fraction might be added to the myriad of plots that complexity induction tracking offers.

Figure 55 below shows a plot of the Qatar complexity induction accumulation total vs. solution total path length for each of the thirteen runs. (Non-valid solutions are also plotted).

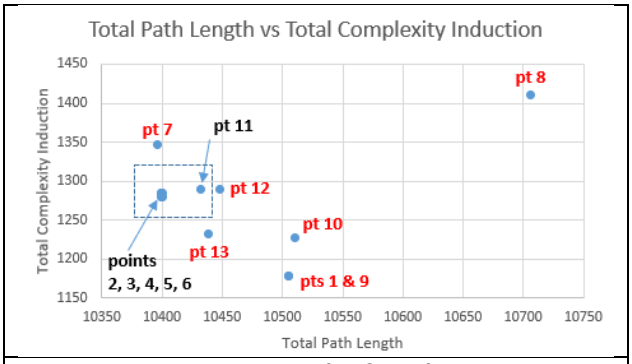


Figure 55 Qatar, Run 2 (Ref 106)

Figure 56 shows a plot of the Qatar complexity induction accumulation total vs. solution total path length for runs 2, 3, 5, 5, 6 and 11 (only).

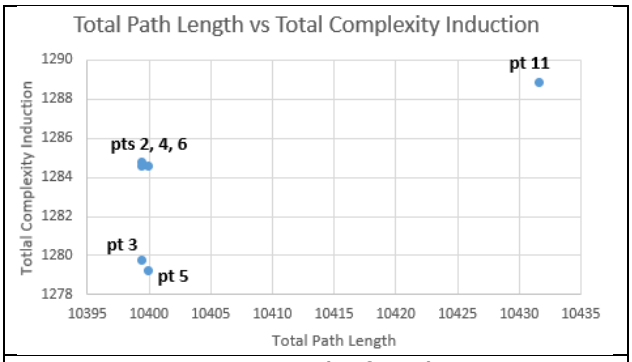


Figure 56 Qatar, Run 2 (Ref 106)

The box which contains the data for runs 2, 3, 5, 5, 6 and 11 might be considered a second order consequence of the associated complexity induction data analyses. Obviously, one needs to be more cautious with making use of second order based trends than with trends associated with first order.

Note that path total distance, complexity induction accumulation, delta_L accumulation and AUC should only be compared amongst valid solutions (with non-valid data deemed non-representative and non-useful).

Current Work: Canada, 1 run, inputs

The Canadian cities dataset (ref 17) was partially run. Obtained results are shown in Figures 57 through 662. A total of 368,497 merge decisions / merge events would be required for path completion.

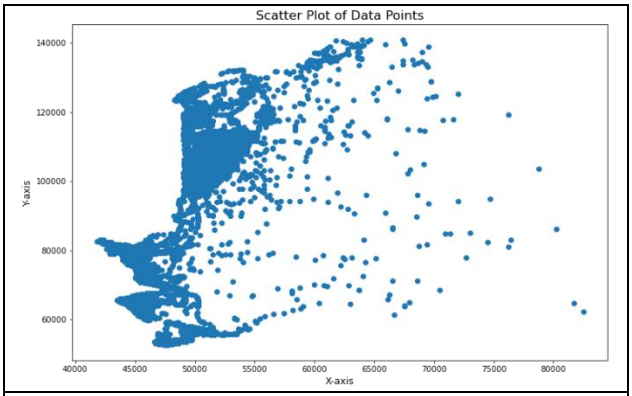


Figure 57 Canada, Ref 17, 4662 cities (4,663 per Ref 106, Researcher somehow misplaced a datum), Scatter Plot

Figure 58 below shows the convex (onion) layers plus 100 merges.

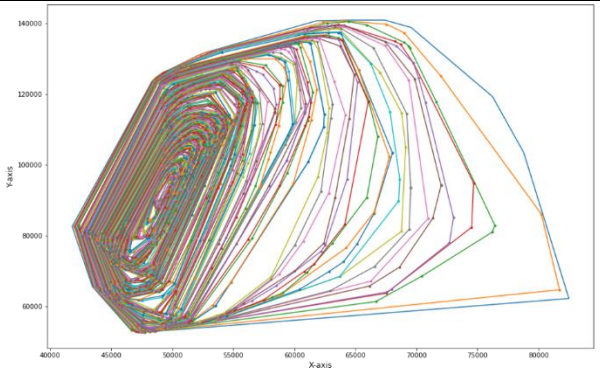


Figure 58 Canada, Convex Hulls (170 total)
Ref 17
4663 cities, Run 2
Convex hulls depicted plus 100 subsequent merges, (hardly visible)).
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90

Figure 59 below plots the 100 data points representing each merge decision made.

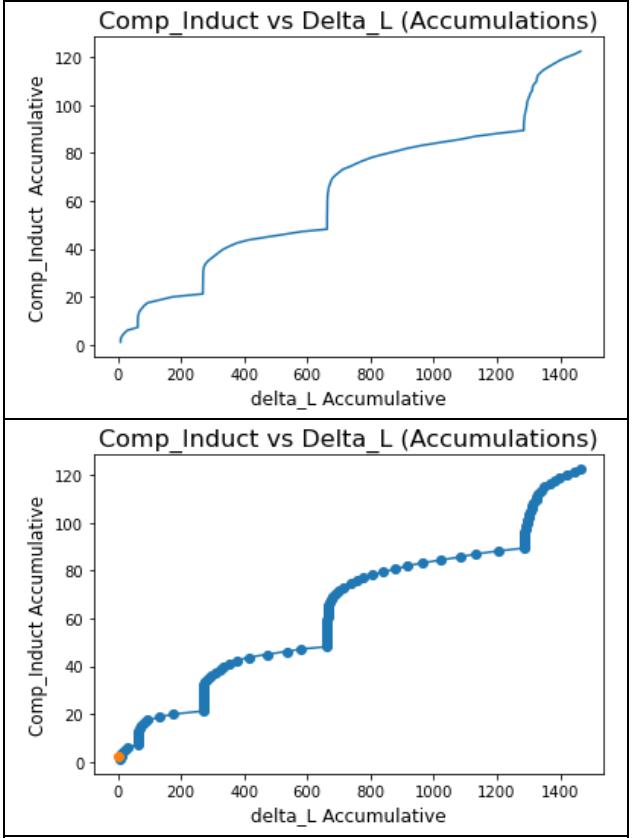


Figure 59 Canada, Ref 17, (100 merges)

4663 cities, Run 2
Convex hulls depicted plus 100 subsequent merges, (hardly visible)).
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90

Figure 60 shows the convex (onion) layers plus 2,000 merges.

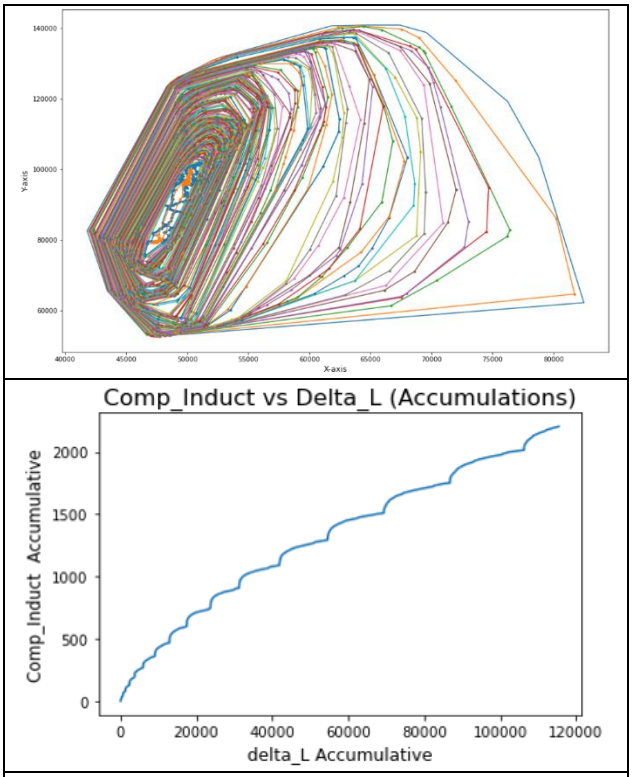
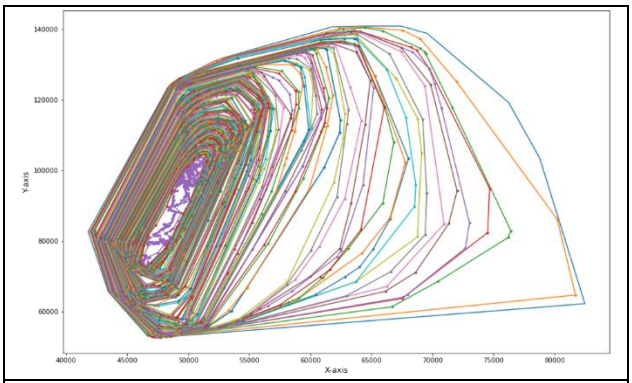


Figure 60 Canada, Ref 17

Figure 60a Canada, Ref 106, 4663 cities, Run 2
Convex hulls depicted plus 2,000 subsequent merges).
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90

Figure 60b Complexity induction Accumulation vs delta_L accumulation (2,000 merges)

Figure 61 shows the convex (onion) layers plus 4,000 merges.



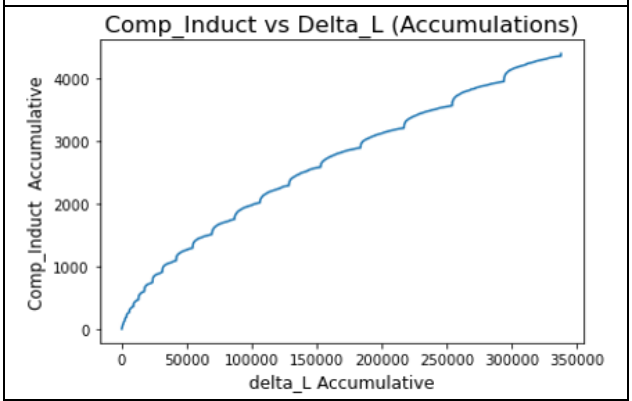


Figure 61 Canada, Ref 17

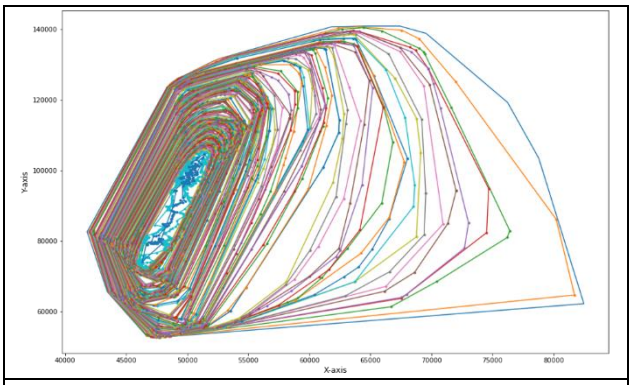
Figure 61a Canada, Ref 106, 4663 cities, Run 2
Convex hulls depicted plus 4,000 subsequent merges).
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90

Figure 61b Complexity induction Accumulation vs delta_L accumulation (4,000 merges)

Complexity Induction Accumulation = 4396

It can now be readily observed that the curve lends itself to credible definition by as few as three (3) linear regression curves (not depicted).

Figure 62 shows the convex (onion) layers plus 6,000 merges.



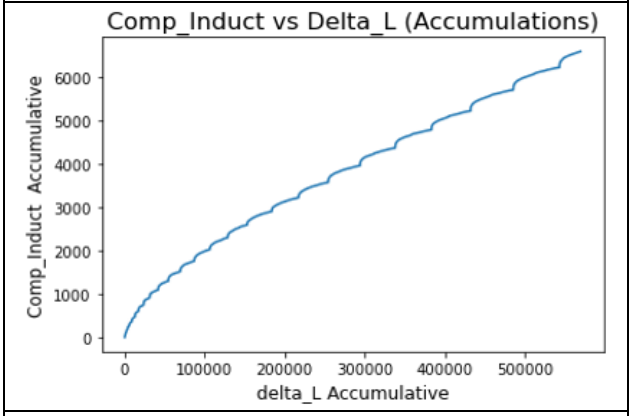


Figure 62 Canada, Ref 17

Figure 62a Canada, Ref 106, 4663 cities, Run 2
Convex hulls depicted plus 6,000 subsequent merges).
A=0, B=C1=C2=C3=C4= 1.00
D1=D2=D3=D4=D5= 0.90

Figure 62b Complexity induction Accumulation vs delta_L accumulation (6000 merges)

Complexity Induction Accumulation = 6582
Delta_L Accumulation = 569,281
AUC = 2,177406,217

Quick Check:
(1/2) * 6,582 * 569,281 = 1,873,503,771

It can now be readily observed that the curve lends itself to credible definition by as little as two (2) linear regression curves (again, not depicted). Obviously the onion layer merging is in early process with some 360,000 more merge decisions/actualizations required to define a path.

4. FINDINGS

Early work: Scope
Runs 2, 3, 4, 5 and 6 provide evidence that the nested convex hull inner-to-outer consumption approach that strictly uses smallest change in distance criteria can be improved upon by including system parameters which are designed to control complexity induction. These runs identified three (3) valid TSP paths. Each of these valid paths yielded a shorter travel distance than the run 1 path (not valid, with one (1) disqualifying cross-over) obtained using shortest delta distance merge criteria (only).

The findings indicate that the locations of the inevitable "switchbacks" can be controlled. Greater control could possibly be attained by increased discretization at proximity of segment end point (regions C and D), although this might have greater applicability to larger size datasets. Also, greater control would likely be attained if a further updated algorithm provided for updating of the Regions C and D parameters during the course of the run.

Additionally, somewhat unexpectedly, there also is evidence that control of complexity induction can "mend" a path and thereby provide for a valid final path where before said validity of the achieved final path was absent (e.g., minimized complexity induction criteria can eliminate some cross-overs).

5. CONCLUSIONS

A specialized subset theory of complexity induction was developed. It was hypothesized that if complexity induction is correctly controlled such that participation thereof in the TSP solution process for a 2D data set using a nested triangles derived convex hulls solver adhering to both shortest delta distance and inner-to-outer layer consumption criteria then improved valid paths might result relative to the same solution approach sans complexity induction. The algorithm focused on the incentivization, de-incentivization of formation of "switchbacks".

For the dataset studied the incentivization of minor, non-severe "switchbacks" was shown to be beneficial. Consequently, said dataset, the hypothesis was proven true. This subset complexity induction theory generates lots of data for subsequent experimental use in proper feedback loops

The AUC concept has been put forth as a useful observation and monitoring tool. For the convex hull solution approach the axes must be relative to sequential and increasing merge decision identification number (always an integer). For other solution methods (not convex hull based) the axes of the subject plot might be directly relatable to identification of point or city visited, or path build from any starting point. The AUC (area under the curve) was introduced as a path characteristic. Its inclusion in an automated parameters feedback loop could be beneficial.

Application of complexity induction theory possibly allows for broadened and heightened machine learning opportunities for the 2D data set convex hulls solution method.

6. FUTURE WORK

The script was constructed such that the system parameters, designed to control complexity, are applicable to and unchanging for the entire run. A more complex algorithm might vary the parameters for individual convex hull layers as well as, perhaps, partially across convex hull layers.

"Visit Densification" Zones

This study was specific to 2D convex hulls solution approach. However, findings were encouraging and provide enough underpinning support for an early mention that complexity induction tracking ought be applicable to any 2D dataset TSP solution approach. It goes without saying that simple tracking thereof changes nothing other than additional computational cost. Bookkeeping for complexity induction use in other TSP solution approaches might include subtraction of complexity induction as well as addition.

A learned algorithm might allow for earlier sacrifices for the sake of later gain. It is proposed that beginnings of proof of this caveat is suggested in this study.

Other 2D dataset complexity introducers besides "switchbacks" or "folds" might be identified.

It is provable that, for any set of n points in 3D space, nested tetrahedrons will require the

greatest number of merge decisions in order to find a TSP solution using traditional convex hull methodology. It is now cautiously postulated in this paper that the simplest state of all of the infinite states that might describe any 3D data set is the very rare occurrence in which the data set can be fully defensibly described as nested tetrahedrons only (with three, two, one or zero leftover interior-most points). Occam's razor and abductive reasoning aside, a certain irony is claimed to be herein present: for equivalent n, the simplest state requires the most number of calculations to arrive at a (nested convex hulls derived) TSP solution.

All of that having been said, as nested triangles are to planar data sets so nested tetrahedrons are to the next higher order data sets. Each allows for simplest definition thereof. Or at least, if any dataset can be properly so characterized then any definition that omits this rarifying fact would indeed seem to be remiss.

The unexpected mileage obtained from a nested triangles derived 2D solver was documented herein. Perhaps a 3D solver, with core being a nested tetrahedrons script, might also, like the nested triangles solver, go some distance.

Again, Chazelle stated in 1985, as final sentence of the CONCLUSIONS of the Ref 8 paper:

"We mention the extension of our techniques to **higher dimensions** as well as the dynamization of the two-dimensional case as two interesting open problems in the general area of convex hull maintenance."

Also, Eddy stated in 1977, as first sentence of final paragraph of the Ref 7 paper:

*"There is no feature inherent in the basic idea of this algorithm which prevents implementation for **dimensions higher than 2**."*

Obviously earlier Researchers (including ones I inadvertently omitted from my Literature Search) were of similar mind as the above two.

Finally, any blending of deterministic 2D solution approaches (convex hull inclusive or otherwise) or other such "dynamization" might benefit from contemplation of inclusion of a working complexity induction system.

8. REFERENCE

Reference 1:
Love, R.F., Dowling, P.D., Brimberg, J. The Use of Nested Hulls In The Symmetric Traveler Problem 1998, Engineering Optimization, 31:2, 261-271, DOI: 10.1080/03052159808941373 To link to this article: https://doi.org/10.1080/03052159808941373

Reference 2:
Matai, R. Singh, S., Mittal, M. Traveling Salesman Problem: An Overview of Applications, Formulations and Solution Approaches, https://www.intechopen.com/chapters/12736 published 12/30/2010, DOI: 10.5772/12909, retrieved 04/21/2023

Reference 3:
Anderson, R., Ashlagi, I., Gamarnik, D., Roth, A., Finding Long Chains in Kidney Exchange Using the Traveling Salesman Problem, Research Article, Applied Mathematics, January 5, 2015, retrieved from https://doi.org/10.1073/pnas.1421853112 April 22, 2023

Reference 4:
Cronin, T.M., The Voronai Diagram for the Euclidean Traveling Salesman Problem is Piecemeal Quartic and Hyperbolic, CECOM Center for Signals Warfare, Warrenton, VA 22186-5100, June 1990, retrieved 11/26/2022

Reference 5:
Barachet, L.L., "Graphic Solution of the Traveling Salesman Problem", Operations Research, 1957 pp. 941-845, retrieved 05/12/2023

Reference 6:
Stewart, Bodin, Convex Hull Algorithm, page, Retrieved May 21, 2021, from https://www2.isye.gatech.edu/~mgoetsch/cali/VEHICLE/TSP/TSP017_.htm

Reference 7:
Eddy, W. F., A New Convex Hull Algorithm for Planar Sets, Carnegie-Mellon University, ACM Trans. Math. Software 3, 4 (Dec. 1977), 411-412, retrieved 01/26/2023

Reference 8:
Chazelle, Bernard (1985). On The Convex Layers of a Planar Set, IEEE Transactions on Information Theory, VOL IT-31, No. 4, July 1985, https://www.cs.princeton.edu/~chazelle/pubs/ConvexLayers.pdf

Reference 9:
Love, R.F., Dowling, P.D., Brimberg, J. The Use of Nested Hulls In The Symmetric Traveler Problem 1998, Engineering Optimization, 31:2, 261-271, DOI: 10.1080/03052159808941373 To link to this article: https://doi.org/10.1080/03052159808941373

Reference 10:
Differentiable Convex Optimization Layers, Agawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., Kolter, Z., 33rd Conf. On Neural Information Processing Systems, (NeurIPS 2019) Vancouver, CA, Retrieved May 21, 2021, from https://web.stanford.edu/~boyd/papers/pdf/diff_cvxpy.pdf

Reference 11:
Cerny, V. Thermodynamic Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, Journal of Optimization Theory and Applications, 45, pp. 41-51 (1985), retrieved Apr. 19, 2023

Reference 12:
"TSP, Data for the Traveling Salesman Problem", Florida State University, Main page. July 23, 2019, Ref. Gerhardt Reinelt, TSPLIB – A Traveling Salesman Problem Library, ORSA Journal on Computing, Volume 3, Number 4, Fall 1991, pages 376-384., Retrieved Dec. 27, 2021, from https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.htmlttps://www.math.uwaterloo.ca/tsp/world/countries.html
P01, 15 Cities (NOT from TSPLIB).
GNU LGPL license applies.

Reference 13:
"Solving TSPs, "Solving TSPs: National Traveling Salesman Problems", University of Waterloo, Mathematics Department, contact William Cook at bico@uwaterloo.ca, Main page. (n.d.). Retrieved Dec. 27, 2021, from https://www.math.uwaterloo.ca/tsp/world/countries.html
Western Sahara, 29 Cities (TSPLIB Format, wi29).

Reference 14:
"TSP, Data for the Traveling Salesman Problem", Florida State University, Main page. July 23, 2019, Retrieved Dec. 27, 2021, from https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.htmlttps://www.math.uwaterloo.ca/tsp/world/countries.html
ATT48, 48 Citiess (TSPLIB Format).
GNU LGPL license applies.

Reference 15:
"Solving TSPs, "Solving TSPs: National Traveling Salesman Problems", University of Waterloo, Mathematics Department, contact William Cook at bico@uwaterloo.ca, Main page. (n.d.). Retrieved Dec. 27, 2021, from https://www.math.uwaterloo.ca/tsp/world/countries.html
Qatar, 194 Cities (TSPLIB Format, qa1940).

Reference 16:
"Solving TSPs, "Solving TSPs: VLSI Data Sets", University of Waterloo, May 2013, provided by Andre Rohe, contact bico@uwaterloo.ca., Forschungsinstitut für Diskrete Mathematik, Universität Bonn
The Bonn Institute is a leading academic site for applied research in VLSI design.
Contact William Cook at bico@uwaterloo.ca, Main page. (May 13, n.y.). Retrieved Dec. 27, 2021, from https://www.math.uwaterloo.ca/tsp/world/countries.html
XQF, 131 Points (TSPLIB Format, xqf131).

Reference 17:
"Solving TSPs, "Solving TSPs: VLSI Data Sets", University of Waterloo, May 2013, provided by Andre Rohe, contact bico@uwaterloo.ca., Forschungsinstitut für Diskrete Mathematik, Universität Bonn
The Bonn Institute is a leading academic site for applied research in VLSI design.
Contact William Cook at bico@uwaterloo.ca, Main page. (May 13, n.y.). Retrieved Dec. 27, 2021, from https://www.math.uwaterloo.ca/tsp/world/countries.html
Canad, 4,663 Citiess (TSPLIB Format, ca4663).

Reference 18:
Number of Layers in Nested Convex Hull, retrieved April 3, 2023, https://math.stackexchange.com/questions/1042376/number-of-layers-in-nested-convex-hull