Approximating TSP Solution by MST Based Graph Pyramid*

Yll Haxhimusa[1,2], Walter G. Kropatsch[1], Zygmunt Pizlo[2], Adrian Ion[1], and Andreas Lehrbaum[1]

[1] Vienna University of Technology,
Faculty of Informatics, Institute of Computer Aided Automation,
Pattern Recognition and Image Processing Group, Austria
{yll,krw,ion,lehrbaua}@prip.tuwien.ac.at
[2] University of Purdue,
Department of Psychological Sciences, USA
{yll,pizlo}@psych.purdue.edu

Abstract. The traveling salesperson problem (TSP) is difficult to solve for input instances with large number of cities. Instead of finding the solution of an input with a large number of cities, the problem is approximated into a simpler form containing smaller number of cities, which is then solved optimally. Graph pyramid solution strategies, in a bottom-up manner using Borůvka's minimum spanning tree, convert a 2D Euclidean TSP problem with a large number of cities into successively smaller problems (graphs) with similar layout and solution, until the number of cities is small enough to seek the optimal solution. Expanding this tour solution in a top-down manner to the lower levels of the pyramid approximates the solution. The new model has an adaptive spatial structure and it simulates visual acuity and visual attention. The model solves the TSP problem sequentially, by moving attention from city to city with the same quality as humans. Graph pyramid data structures and processing strategies are a plausible model for finding near-optimal solutions for computationally hard pattern recognition problems.

1 Introduction

Traveling salesperson problem (TSP) is a combinatorial optimization task of finding the shortest tour of n cities given the intercity costs. When the costs between cities are Euclidean distances, the problem is called Euclidean TSP (E-TSP). TSP as well as E-TSP belongs to the class of difficult optimization problems called NP-hard and NP-complete if posed as a decision problem [1]. The straightforward approach by using brute force search would be using all possible permutations for finding the shortest tour. It is impractical for large n since the number of permutations is $\frac{(n-1)!}{2}$. Because of the computational

* Supported by the Austrian Science Fund under grants P18716-N13 and S9103-N04, and the USA Air Force Office of Scientific Research.

F. Escolano and M. Vento (Eds.): GbRPR 2007, LNCS 4538, pp. 295–306, 2007.

intractability of TSP, researchers concentrated their efforts on finding approximating algorithms. Good approximating algorithms can produce solutions that are only a few percent longer than an optimal solution and the time of solving the problem is a low-order polynomial function of the number of cities [2,3,4]. The last few percent to reach optimality are computationally the most expensive to achieve.

It is by now well established that humans produce close-to-optimal solutions to E-TSP problems in time that is (on average) proportional to the number of cities [5,6,7]. This level of performance can not be reproduced by any of the standard approximating algorithms. Some approximating algorithms produce smaller errors but the time complexity is substantially higher than linear, other algorithms are relatively fast but produce substantially higher errors. It is therefore of interest to identify the computational mechanism used by the human brain.

A simple way to present E-TSP to a subject is to show n cities as points on a computer screen and ask the subject to produce a tour by clicking on the points. In Figure 1a, an E-TSP example of 10 cities is shown and in c the solution given by a human. The tours produced by the subjects are, on average, only a few percent longer than the shortest tours (in Figure 1c and d the cross depicts the starting position and the arrow the orientation used by the subject). The solution time is a linear function of the number of cities [5,6]. Two attempts to emulate human performance by a computational model were undertaken in [5,6]. In [5], authors attempt to formulate a new approximating algorithm for E-TSP motivated by the failure to identify an existing algorithm that could provide a good fit to the subjects' data. The main aspects of the models in [5,7] are its

- (multiresolution) pyramid architecture, and
- a coarse to fine process of successive tour approximations.

They showed that performance of this model (proportion of optimal solutions and average solution error) is statistically equivalent to human performance. Pyramid algorithms have been used extensively in both computer and human vision literature (e.g. [8]), but not in problem solving. The work of [5,9] was the first attempt to use pyramid algorithms to solve the E-TSP. One of the most attractive aspects of pyramid algorithms, which make them suitable for problems

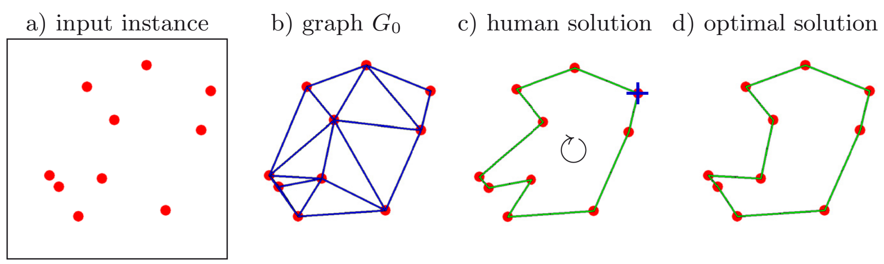


Fig. 1. E-TSP and solutions given by human and optimal solver

such as early vision or E-TSP, is that they allow to solve (approximately) global optimization tasks without performing a global search. A similar pyramid algorithm for producing approximate E-TSP solutions with emphasis on trade-off between computational complexity (speed) and error in the solution (accuracy) and not on modeling human performance is formulated in [4, Chap.5], and [10].

In this paper we present a computational model for solving E-TSP approximately based on the multiresolution graph pyramid. *The emphasis is on emulating human performance (time and accuracy), and not in finding an algorithm for solving E-TSP as optimally as possible.* The interested reader can consult a large body of the literature in Operations Research for algorithms for E-TSP [4,3] that can produce near to optimal tours. Again, these algorithms have computational complexity that is substantially higher than linear.

Our goal is to show that the results of our model are well fitted to the results of the humans, and the quality and speed are comparable to that of human subjects. The next section presents a short overview of the pyramid representations (Section 2). In Section 3 the solution of the E-TSP using a minimum spanning tree (MST) based graph pyramid is introduced. The bottom-up simplification of the input data is shown in Section 3.1, and in Section 3.2 the top-down approximative solution is described. Psychophysical experiments on E-TSP are presented in Section 4.

2 Irregular Graph Pyramid

In our framework, the TSP input is represented by graphs where cities are represented by vertices, and the intercity neighborhoods by edges (see Figure 1b). Each vertex of the constructed input graph must have at least two edges for the TSP tour to exist. A level (k) of the graph pyramid consists of a graph G_k. Moreover the graph is attributed, $G = (C, N, w_v, w_e)$, where $w_e : N \to \mathbb{R}^+$ is a weighted function defined on edges N. The weights w_e are Euclidean distances in the E-TSP and $w_v : C \to \mathbb{R}^+$ is a weighted function defined on cities C. I.e. each vertex (city) has as a weight its position in the Cartesian coordinate system

Finally, the sequence G_k, $0 \leq k \leq h$ is called irregular graph pyramid.

In a regular pyramid, the number of vertices at any level k is λ times higher than the number of pixels at the next (reduced) level $k+1$. The so called reduction factor λ is greater than one and it is the same for all levels k. The number of levels on top of G amounts to $\log_\lambda(|G|)$. This implies that a pyramid is build in $\mathcal{O}[\log(diameter(G))]$ parallel steps [8]. Regular image pyramids are confined to globally defined sampling grids and lack shift invariance [11]. In [12,13] it is shown how these drawbacks can be avoided by adaptive irregular pyramids.

In Graham's model [5], clusters are not explicitly represented. Instead, the centers of the clusters were used in the E-TSP solution process. The centers were modes (peaks) of the intensity distribution produced by blurring the image. To make clusters explicit, Pizlo et. al [14] used an adaptive model in which adaptive top-down partitioning of the plane along the axis of Cartesian system was used. The hierarchy was represented by a binary tree. This top-down clustering had the advantage that the entire E-TSP did not have to be represented at once in the memory. The disadvantage was that although this algorithm was invariant to translation, it was not invariant to rotation. Our new model uses graphs as representation, which are invariant to both translation and rotation of the input city constellation. However, the clustering is performed in bottom-up fashion.

3 Solving E-TSP by a Graph Pyramid

Let $G_0 = (C, N, w_v, w_e)$ be the input graph, with weights on edges given as distances in L_2 space. The goal of the TSP is to find an nonempty ordered sequence of vertices and edges $(v_0, e_1, v_1, ..., v_{k-1}, e_k, v_k, ..., v_0)$ over all vertices of G_0 such that all the edges and vertices are distinct, except the start and the end vertex v_0. This tour is called the optimal tour τ_{opt} and the sum of edge weights in this tour is minimal, i.e.

$$\tau_{opt} = \sum_{e \in \tau} w_e \rightarrow \min,$$

where w_e is the weight of edge e.

We use local to global and global to local processes in the graph pyramid to find a good solution τ^*, approximating the E-TSP. The main idea is to use:

- bottom-up processes to reduce the size of the input, and
- top-down refinement to find an (approximate) solution.

The size of the input (number of vertices in the graph) is reduced such that an optimal (trivial) solution can be found by the combinatorial search, e.g. for a 3 city instance (not all cities are co-linear) there is only one solution, not needing any search, and this is the optimal one. For a 4 city input (not all co-linear) there are three solutions from which two are non-optimal since they cross edges. A pyramid is used to reduce the size of the input in the bottom-up process. The (trivial) solution is then found at the top of the pyramid and refined in a process emulating fovea by humans using lower levels of this pyramid, i.e. the vertical neighborhoods (parent-children relations) are used in this process to refine the tour. The final, in general non-optimal, solution is found when all the cities at the base level of the pyramid are in the tour. The steps needed to find the E-TSP solution are shown in Algorithm 1. Partitioning of the input space is treated in Section 2. Sections 3.1 and 3.2 discuss steps 2 and 4 of Algorithm 1 in more detail.

3.1 Bottom-Up Simplification Using an MST Pyramid

The main idea is that cities being close neighbors are put into a cluster and considered as a single city at reduced resolution. By doing this recursively one produces a pyramid representation of the problem. It is well known that the human visual system represent images on multiple level of scales and resolution [15,16].

Algorithm 1. Approximating E-TSP Solution by an MST Graph Pyramid

Input: Attributed graph $G_0 = (C, N, w_v, w_e)$, and parameters r and s

partition the input space by preserving approximate location:
create graph G_0
reduce number of cities bottom-up until the graph contains s vertices:
build graph pyramid $G_k, \forall k = 0, ..., h,$ *where* $s = |G_h|$
find the optimal tour τ_a for the graph G_h
refine solution top-down until all vertices at the base level are processed:
refine τ_a *until level* 0 *is reached*

Output: Approximate TSP solution τ^*.

There are many different algorithms to make hierarchical clustering of cities [17]. We choose for this purpose the MST principle, especially Borůvka's algorithm [18] since it hierarchically clusters neighboring vertices. The time complexity of Borůvka's algorithm is $\mathcal{O}(|E| \log |V|)$. It can be shown that MST can be used as the natural lower bound and for the case of the TSP with the triangle inequality, which is the case for the E-TSP, it can be used to prove the upper bound as well [19]. The first step in Christofides' heuristics [2] is finding an MST as an approximation of TSP. Christofides shows that it is possible to achieve at least $\frac{3}{2}$ times of the optimal solution of TSP i.e. Christofides heuristics solution of TSP is at most 50% longer than the optimal solution.

For a given graph $G_0 = (C, N, w_v, w_e)$ the vertices are hierarchically grouped into trees (clustered) as given in Algorithm 2. The idea of Borůvka is to do greedy steps like in Prim's algorithm [20], in parallel over the graph at the same time. The size of trees (clusters) are not allowed to contain more than $r \in \mathbb{N}^+$ cities. These trees must contain at least 2 cities, due to the fact that the pyramid must have a logarithmic height [21], since the reduction factor λ is $2 \leq \lambda \leq r$. This parameter can be related also to the number of 'concepts' that humans can have in their 'memory buffer', and is usually not larger than 10.

The number $s \in \mathbb{N}^+$ of vertices in the top level of the pyramid is chosen such that an optimal tour can be found easily (usually $s = 3$, or $s = 4$). Note that

Algorithm 2. Reduction of the E-TSP Input by an MST Graph Pyramid

Input: Attributed graph $G_0 = (C, N, w_v, w_e)$, and parameters r and s

$k \leftarrow 0$
repeat
  $\forall v_k \in G_k$ find the edge $e' \in G_k$ with minimum w_e incident into this vertex
  using e' create trees T with no more than r vertices
  contract trees T into parent vertices v_{k+1}
  create graph G_{k+1} with vertices v_{k+1} and edges $e_k \in G_k \setminus T$
  attribute vertices in G_{k+1}
  $k \leftarrow k + 1$
until there are s vertices in the graph G_{k+1}.

Output: Graph pyramid – G_k, $0 \leq k \leq h$.

larger s means a shallow pyramid and larger graph at the top, which also means higher time complexity to find the optimal tour at the top level. Thus r and s are used to control the trade off between speed and quality of solution.

An example of how Algorithm 2 builds the graph pyramid (only the last two levels) is shown in Figure 2. Each vertex (black in G_{h-1}) finds the edge with the minimal weight (solid lines in G_{h-1}). These edges create trees of no more than r $(= 4)$ cities. These trees are then contracted to the parent vertices (enclosed black vertices in G_{h-1} are contracted into white vertices in G_h). The parent vertices together with edges not touched by the contraction are used to create the graph of the next level (parallel edges and self loops can be removed, since they are not needed for the clustering of vertices). The dotted lines between vertices in different levels represent the parent-child relations. The new parent vertex attribute can be the gravitational center of its child vertices, or by using the position of the vertex near this gravitational center. The algorithm iterates until there are s vertices at the top of the pyramid, and since s is small a full search can be employed to find the optimal tour τ_a at the top quickly.

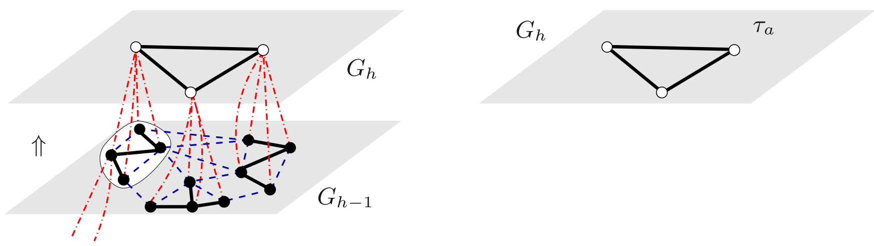
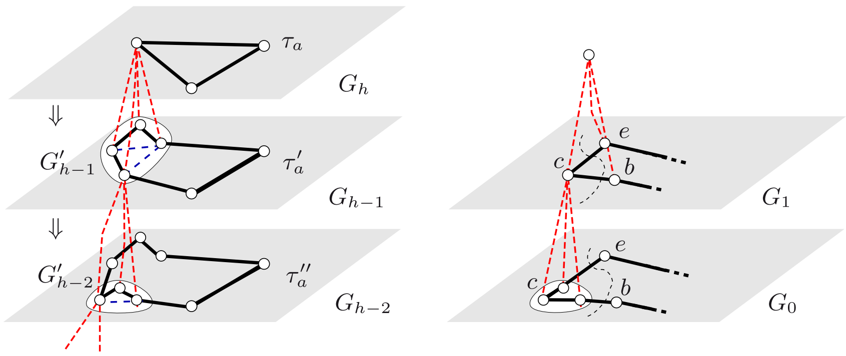


Fig. 2. Building the graph pyramid and finding the first TSP tour approximation

In our current software implementation we use the fully connected graph to represent the input instance, as expected the bottom-up simplification algorithm has at least $\mathcal{O}(|E|^2)$ time complexity [22]. This time complexity can be reduced easily to $\mathcal{O}(|E| \log |V|)$ if instead of the fully connected graph one uses a planar graph e.g. Delaunay triangulation.

3.2 Top-Down Approximation of the Solution

The tour τ_a found at level h of the graph pyramid is used as the first approximation of the TSP tour τ^*. This tour is then refined using the pyramid structure already built. Similar to Pizlo et. al. [14] we have chosen to use the most simple refinement, the one-path refinement. The one-path refinement process starts by choosing (randomly) a vertex v in the tour τ_a. Using the parent-child relationship, this vertex is expanded into the subgraph $G'_{h-1} \subset G_{h-1}$ from which it was created i.e. its receptive field in the next lower level. In this subgraph a path between vertices (children) is found that makes the overall path τ'_a the shortest



a) top-down refinement τ_a^* b) the flow of processing

Fig. 3. Refining the E-TSP tour by a graph pyramid

one (see Figure 3a). Since the number of vertices (children) in G'_h cannot be larger than r, a complete search is a plausible approach to find the path with the smallest contribution in the overall length of the tour τ'_a. Note that edges in the τ'_a are not necessarily the contracted edges during bottom-up construction.

The refinement process then choses one of the already expanded vertices in G'_{h-1}, say v' and expands it into its child at the next lower level G'_{h-2}, and the tour τ''_a is computed. The process of tour refinement proceeds recursively until there are no more parent-children relationships (graph G_0, Figure 3b vertices of the receptive field of c, $RF(c)$), i.e. vertices at the base of the pyramid are reached. E.g. in Figure 3b, the tour is refined as the shortest path between the start vertex b and end vertex e and all the vertices (children of c) of the $RF(c)$. After arriving at the finest resolution, the process of refinement continues by taking a vertex in the next upper level in the same cluster (Figure 3 vertex b or e), and expanding it to its children and computing the tour. Note that the process of vertex expansion toward the base level emulates the movement of fovea (attention) in the process of solving the problem by a human observer. The tour is refined to the finest resolution in one part whereas other parts are left in their coarse resolution. The process converges when all vertices in the pyramid have been 'visited'[3]. More formally the steps are depicted in Algorithm 3, and Procedure 1, and 2.

Other refinement approaches can be chosen as well. One can use different approaches of refinement for e.g. one can think of using many vertices and expanding them in parallel (multi-path refinement), or use the one-path refinement until a particular level of the pyramid and continue with the multi-path refinement afterward. In these cases one needs to change Procedure 1. Note that there is a randomness in choosing which of the vertices to refine, which is may correspond to individual differences on how humans choose from which vertex to start the tour. In this case one needs to change Procedure 2.

[3] A demo is given in http://www.prip.tuwien.ac.at/Research/twist/results.php.

Algorithm 3. E-TSP Solution by a MST Graph Pyramid

Input: Graph pyramid G_k, $0 \leq k \leq h$ and the tour τ_a

$\tau^* \leftarrow \tau_a$
$v \leftarrow$ random vertex of τ^*
repeat
  refine(τ^*, v) /* refine the path using the children of v. See Prc. 1 */
  mark v as visited
  $v \leftarrow$ nextVertex(G_k, v, τ^*) /* get next vertex to process. See Prc. 2 */
until $v = \emptyset$

Output: Approximation E-TSP tour τ^*.

Procedure 1. refine(τ^*, v): refine a path τ^* using the children of v

Input: Graph pyramid G_k, $0 \leq k \leq h$, the tour τ^*, and the vertex v.

$(c_1, \ldots, c_n) \leftarrow$ children of v /* vertices that have been contracted to v */
if $n > 0$ /* v is not a vertex from the bottom level */ **then**
  $v_p, v_s \leftarrow$ neighbours of v in τ^* /* predecessor and successor of v */
  $p_1, \ldots, p_n \leftarrow$ argmin{length of path $\{v_p, c_{p_1}, \ldots, c_{p_n}, v_s\}$} such that $p_1, \ldots, p_n$ is a permutation of $1, \ldots, n$ /* optimal order of new vertices in the tour */
  replace path $\{v_p, v, v_s\}$ in τ^* with path $\{v_p, c_{p_1}, \ldots, c_{p_n}, v_s\}$

Output: refined TSP tour τ^*.

Procedure 2. nextVertex(G_k, v, τ^*): get next vertex to process

Input: Graph pyramid G_k, $0 \leq k \leq h$, the vertex v, and the tour τ^*

repeat
  if v has unvisited children **then**
    $v \leftarrow$ first unvisited child of v in τ^* /* given an orientation */
  else if v has unvisited siblings **then**
    $v \leftarrow$ first unvisited sibling of v in τ^* /* given an orientation */
  else if v has a parent i.e. v is not a vertex of the top level **then**
    $v \leftarrow$ parent of v
  else
    $v \leftarrow \emptyset$
until (v not visited) $\bigvee$ ($v = \emptyset$)

Output: new vertex to process v.

4 Psychophysical Evaluation of Solutions

Four subjects (including one author) were tested. Each subject solved the same 100 E-TSP problems in a different order. There were 4 different sizes 6, 10, 20, and 50 cities, with 25 instances per problem size. The cites in each problem were generated randomly on a 256×256 square grid [7]. Examples of 10 city tours produced by the subject and by the model are presented in Figure 4. The crosses depict the starting point chosen by the subjects and the model. BSL, OSK, and ZP chose the clock-wise tour, whereas ZL the counter-clock-wise tour. The MST based pyramid model choses randomly the orientation of the tour. To test how well the model fits the subject data, the algorithm is run 15 times with different parameters r ($2 \leq r \leq 7$). The results of the best model fitting (as well as the standard deviation) to the subject data are shown in Figure 5. It can be seen that fit are quite good. The worst fit is for the case of 50-city problems (especially for OSK). Specifically, the model's performance is not as good as that of the subjects. To improve the models's performance, higher values of r would have to be used. This is how the simulation were performed in [14].

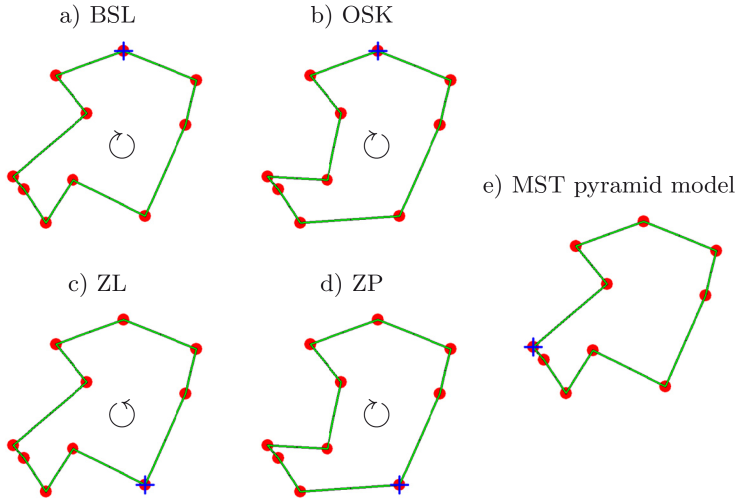


Fig. 4. E-TSP solutions by humans subjects and the MST pyramid model

For larger instances (> 100 cities) data with human subjects are difficult to obtain. Therefore we tested the results of the Algorithm 1 with the state-of-the-art Concorde TSP solver[4] with respect to time and with *adaptive pyramid* [14] with respect to the solution error. The test is done with respect to the quality of results, and the time needed to solve input examples with 200, 400, 600, 800, and 1000 cities. The error values are shown in Figure 6a and the time performance in Figure 6b. The time plot is normalized to the time needed for methods to solve the 200 city instance in one second. We have fixed the values of the parameter $r = 7$ and $s = 3$ for these experiments. Note that the Concorde algorithm solves the problem optimally, i.e. no error. We show that the results of the MST-based model are comparable to humans in quality and speed, and scale well with large input instances. This solution strategy emulates human fovea by moving attention from city to city.

[4] http://www.tsp.gatech.edu/concorde/index.html

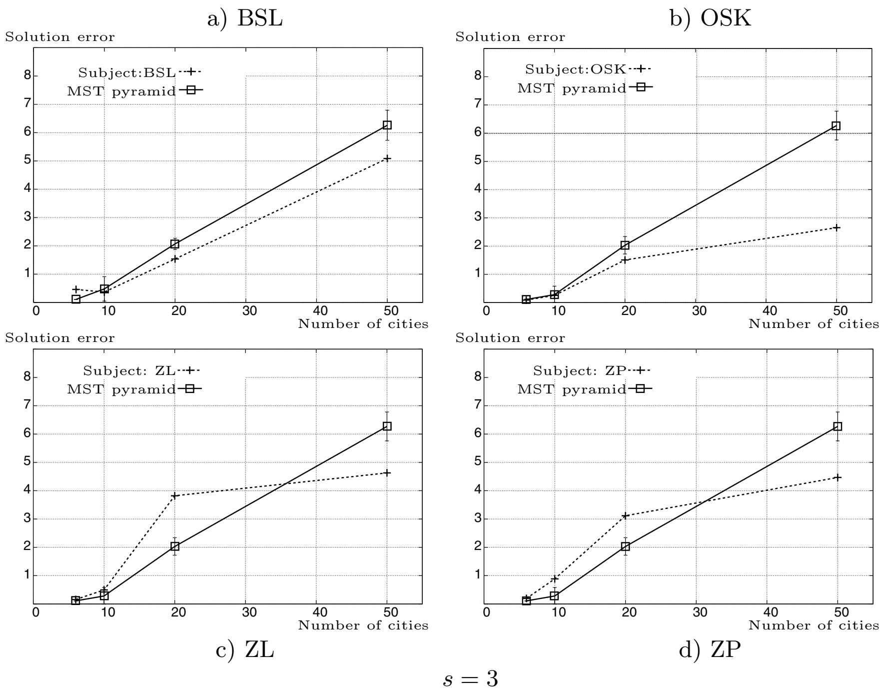


Fig. 5. Model fitting on human data

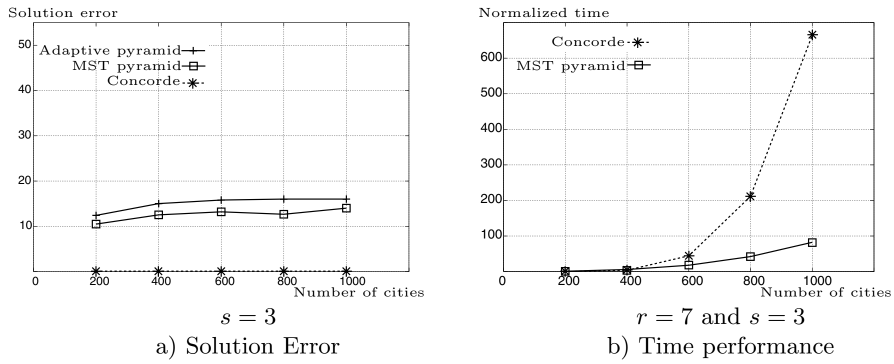


Fig. 6. The solution error and the time performance

5 Conclusion

Pyramid strategies convert in a bottom-up process a 2D Euclidean TSP problem with a large number of cities into successively smaller problems with similar layout and solution until the number of cities is small enough to seek the optimal solution. Expanding this solution in a top-down manner to the lower levels of the pyramid approximates the solution. The introduced method uses a version of Borůvka's MST construction to reduce the number of cities. A top-down process is then employed to approximate the E-TSP solution of the same quality and at the same speed as humans do. The new model has an adaptive spatial structure and it simulates visual acuity and visual attention. Specifically, the model solves the E-TSP problem sequentially, by moving attention from city to city, the same way human subjects do. We showed that the new model fits the human data. Pyramid data structures and processing strategies are a plausible model for finding near-optimal solutions for NP-hard pattern recognition problems, e.g. matching.

Acknowledgment. The authors would like to thank anonymous reviewers for their valuable comments.

References

1. Johnson, D.S., McGeoch, L.A.: Local Search in Combinatorial Optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) The Traveling Salesman Problem: A Case Study in Local Optimization, pp. 215–310. John Wiley and Sons, Chichester (1997)
2. Christofides, N.: Graph Theory - An Algorithmic Approach, New York, London. Academic Press, San Francisco (1975)
3. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The Traveling Salesman Problem. Wiley, New York (1985)
4. Gutin, G., Punnen, A.P.: The traveling salesman problem and its variations. Kluwer, Dordrecht (2002)
5. Graham, S.M., Joshi, A., Pizlo, Z.: The travelling salesman problem: A hierarchical model. Memory and Cognition 28, 1191–1204 (2000)
6. MacGregor, J.N., Ormerod, T.C., Chronicle, E.P.: A model of human performance on the traveling salesperson problem. Memory and Cognition 28, 1183–1190 (2000)
7. Pizlo, Z., Li, Z.: Pyramid algorithms as models of human cognition. In: Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging, SPIE, pp. 1–12 (2003)
8. Jolion, J.M., Rosenfeld, A.: A Pyramid Framework for Early Vision. Kluwer, Dordrecht (1994)
9. Pizlo, Z., Joshi, A., Graham, S.M.: Problem solving in human beings and computers. Technical Report CSD TR 94-075, Department of Computer Sciences, Purdue University (1994)
10. Arora, S.: Polynomial-time approximation schemes for euclidean tsp and other geometric problems. Journal of the Association for Computing Machinery 45, 753–782 (1998)
11. Bister, M., Cornelis, J., Rosenfeld, A.: A critical view of pyramid segmentation algorithms. Pattern Recognition Letters 11, 605–617 (1990)
12. Montanvert, A., Meer, P., Rosenfeld, A.: Hierarchical image analysis using irregular tesselations. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 307–316 (1991)
13. Jolion, J.M., Montanvert, A.: The adaptive pyramid, a framework for 2D image analysis. Computer Vision, Graphics, and Image Processing: Image Understanding 55, 339–348 (1992)

14. Pizlo, Z., Stefanov, E., Saalweachter, J., Li, Z., Haxhimusa, Y., Kropatsch, W.G.: Traveling salesman problem: a foveating model. Journal of Problem Solving 1, 83–101 (2006)
15. Watt, R.J.: Scanning from coarse to fine spatial scales in the human visual system after the onset of a stimulus. Journal of the Optical Society of America 4, 2006–2021 (1987)
16. Pizlo, Z., Rosenfeld, A., Epelboim, J.: An exponential pyramid model of the time-course of size processing. Vision Research 35, 1089–1107 (1995)
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, Chichester (2001)
18. Neštřil, J., Miklovà, E., Neštřilova, H.: Otakar Borůvka on minimal spanning tree problem translation of both the 1926 papers, comments, history. Discrete Mathematics 233, 3–36 (2001)
19. Atallah, M.J. (ed.): Algorithms and Theory of Computational Handbook. CRC Press, Boca Raton, FL (1999)
20. Prim, R.C.: Shortest connection networks and some generalizations. The. Bell. System Technical Journal 36, 1389–1401 (1957)
21. Kropatsch, W.G., Haxhimusa, Y., Pizlo, Z., Langs, G.: Vision pyramids that do not grow too high. Pattern Recognition Letters 26, 319–337 (2005)
22. Papadimitiou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publication, Mineola, NY (1998)