Approximating TSP Solution by MST based Graph Pyramid*

Y. Haxhimusa[1,2], W. G. Kropatsch[1], Z. Pizlo[2], A. Ion[1], and A. Lehrbaum[1]

[1] Vienna University of Technology, Faculty of Informatics,
Pattern Recognition and Image Processing Group, Austria
{yll,krw,ion,lehrbaua}@prip.tuwien.ac.at
[2] University of Purdue, Department of Psychology, USA
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 a Borůvka's miniman spanning tree (MST), 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 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 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

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

low-order polynomial function of the number of cities [2–4]. The last few percent to reach optimality are computationally the most expensive to achieve.

Interestingly, humans are known to produce close-to-optimal solutions to E-TSP problems in time that is (on average) proportional to the number of cities [5–7]. 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 Fig. 1a a E-TSP example of 10 cities is shown and in c the solution given by the human. The tours produced by the subjects are, on average, only a few percent longer than the shortest tours (in Fig. 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 main 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 aspect of model in [5] is its i) (multiresolution) pyramid architecture, and ii) 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 such as early vision or E-TSP, is that they allow to solve (approximately) global optimization tasks without performing global search. A similar pyramid algorithm for producing approximate E-TSP solutions with emphasis on trade-off between computational complexity (speed) and error of the solution (accuracy) and not on modeling human performance in formulated [4, Chap.5].

In this paper we present a human computational model for solving E-TSP approximately based on the multiresolution graph pyramid. The accent is on emulating human performance, and not in finding an algorithm for solving E-TSP as optimal as possible. Our goal is to show that the results of the introduced model are well fitted to the results of the humans, and the quality and speed are comparable to human subjects. The next section presents a short overview of the pyramid representations (Sec. 2). In Sec. 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 Sec. 3.1, and in Sec. 3.2 the top-down approximative solution is described. Psychophysical experiments on E-TSP are presented in Sec. 4.

2 Irregular Graph Pyramid

In our framework, the input to TSP is represented by graphs where cities are represented with vertices, and the intercity neighborhood with edges (in Fig. 1b the Delauney graph is shown). 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

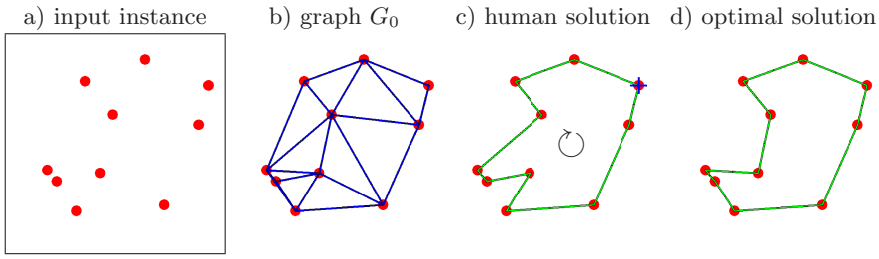


Fig. 1. E-TSP and solutions given by human and optimal solver.

consists of a graph G_k. Moreover the graph is attributed, $G = (C, N, w_v, w_e)$, where $w_v : C \rightarrow \mathbb{R}^+$ is a weighted function defined on vertices and $w_e : N \rightarrow \mathbb{R}^+$ is a weighted function defined on edges. For the E-TSP the weights w_e are Euclidean distances. The weight of a vertex can be thought of as the position of the city in the Cartesian coordinate system. Finally, the sequence G_k, $0 \leq k \leq h$ is called irregular graph pyramid.

In a regular image pyramid, the number of pixels 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 new levels on top of an image I amounts to $\log_\lambda(|I|)$. This implies that an image pyramid is build in $\mathcal{O}[\log(diameter(I))]$ parallel steps [8]. However, 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.

The model in [5] is based on regular pyramid representation, whereas the model in [7] on quad trees, thus by shifting the input different solutions are produced. Recently, [14], introduced an adaptive model trying to overcome this drawback, by adaptive partitioning the plane along the axis of Cartesian system and using the quad trees to represent the hierarchy. Our model uses graphs as representation, therefore it is invariant to shifting and rotation of the input city constellation. Moreover, using graph contraction [16] we create a pyramid that adapts its structure to the input data.

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 TSP is to find the nonempty 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 of the edges and vertices are distinct, except the start and the end vertex v_0, called the tour τ_{opt} such that the sum of edge weights is minimal, i.e. $\sum_{e \in \tau} w_e \rightarrow \min$. 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

Algorithm 1 – Approximating E-TSP Solution by a MST Graph Pyramid

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

1: partition the input space by preserving approximate location:
create graph G_0
2: reduce number of cities bottom-up until the graph contains p vertices:
build graph pyramid $G_k, \forall k = 0, ..h,$ *where* $p = |G_k|$
3: find the optimal tour τ_a for the graph G_h
4: 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^*.

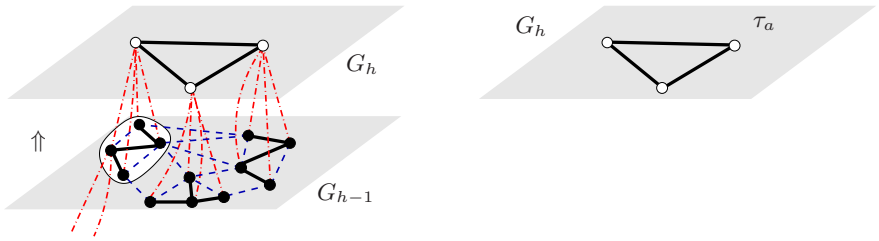
– bottom-up processes to reduce input size, and
– top-down refinement to find an (approximate) solution.

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 the 3 city instance there is only one solution, not needing any search, and this is the optimal one. For the 4 city input (not all colinear) 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 Alg. 1. Partitioning of the input space is treated in Sec. 2. In the Subsections 3.1 and 3.2 steps 2 and 4 of Alg. 1 are discussed in more details.

3.1 Bottom-up Simplification using an MST Pyramid

The main idea is that cities being close neighbors are put into a cluster, and thus reducing the resolution of the input. We are driven to use this closeness concept since it seems that humans use this cue heavily when they solve the E-TSP. Since the whole input instance is not in the visual field of view in the same resolution (in general) and the time needed for the problem to be solved by humans is (near) linear, seems that there might be some organization of the input data during the E-TSP solving process.

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 easily 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



a) bottom-up reduction of resolution b) initial tour τ_a

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

at least 1 1/2 times of the optimal solution of TSP i.e. Christofides heuristics solution of TSP $\leq 3/2$ of 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 Alg. 2. The idea of Borůvka is to do greedy steps like in Prim's algorithm [15], 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 [20], 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.

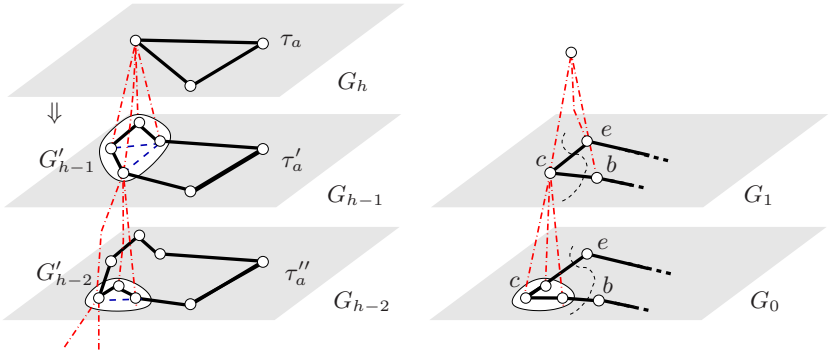
The parameter $s \in \mathbb{N}^+$, the number of vertices in the top level of the pyramid, is chosen such that an optimal tour can be found easily (usually $s = 3$, or $s = 4$) Note that larger s means shallow pyramid and larger graph at the top, which also means higher time complexity to find the optimal tour at the top level. Thus r and s can be used to control the trade off between speed and quality of solution. An example of how the Alg. 2 builds the graph pyramid (only the last two levels) is shown in Fig. 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 (e.g. the

Algorithm 2 – Reduction of the TSP Input by an MST Graph Pyramid

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

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

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



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

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

tree enclosed in the figure) of no more than r $(= 4)$ cities. These trees are then contracted to the parent vertices (enclosed white vertices in G_{h-1}). 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 children 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 quickly.

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. In this paper 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 one (see Fig. 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 necessary 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 children at the next lower level G'_{h-2}, and the tour τ''_a is computed. The process of tour refinement proceeds similarly until there are no more parent-children relationships (graph G_0, Fig. 3b) vertices of

the receptive field of c, $RF(c)$), i.e. the vertices at the base of the pyramid at the same branch are reached. E.g. in Fig. 3b, the tour is refined as the shortest path between the begin 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 (see Fig. 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 fovea in the human visual system. The tour is refined to fine 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 Alg. 3, and Prc. 1, and 2.

Other refinement approaches can be chosen as well, just by changing Prc. 1 and 2. One can use different approaches of refinement for e.g. one can think of using many vertices and expand them in parallel (multi-path refinement), or use the one-path refinement until a particular level of the pyramid and continuous with the multi-path refinement afterward. Note that there is a randomness in choosing which of the vertices to refine, which is well associated to the 'random' decision of humans in choosing from which vertex to start the tour.

4 Psychophysical Evaluation of Solutions

Four subjects (BSL, OSK, ZL, and ZP) have solved E-TSPs of different sizes: 6, 10, 20, and 50 cities [7]. Some qualitative results of the tours made by human and the model introduced are presented in Fig 4. The crosses depict the starting point chosen by the subjects and the method. 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 the model how well fits to the subject data, the algorithm is run 15 times with different parameter r. The results of the best model fitting (as well as the standard deviation) to the subject data is shown in Fig. 5.

For larger instances ($>$ 50 cities) data with human subjects are difficult to obtain, thus we tested the results of the Alg. 1 with the state-of-the-art

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

Algorithm 3 – TSP Solution by a MST Graph Pyramid

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

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

Output: Approximation 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.

1: $(c_1, \ldots, c_n) \leftarrow$ children of v /* vertices that have been contracted to v */
2: **if** $n > 0$ /* v is not a vertex from the bottom level */ **then**
3: $v_p, v_s \leftarrow$ neighbours of v in τ^* /* predecessor and successor of v */
4: $p_1, \ldots, p_n \leftarrow$ 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 */
5: 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^*

1: **repeat**
2: **if** v has unvisited children **then**
3: $v \leftarrow$ first unvisited child of v in τ^* /* given an orientation */
4: **else if** v has unvisited siblings **then**
5: $v \leftarrow$ first unvisited sibling of v in τ^* /* given an orientation */
6: **else if** v has a parent i.e. v is not a vertex of the top level **then**
7: $v \leftarrow$ parent of v
8: **else**
9: $v \leftarrow \emptyset$
10: **until** (v not visited) $\bigvee$ ($v = \emptyset$)

Output: new vertex to process v.

Concorde TSP solver[4] with respect to time and with other pyramid algorithms used in human problem solving, *adaptive pyramid* [14] and *quad pyramid* [7], 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 Fig. 6a and in b the time performance. The time plot is normalized with 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. Since in our current software implementation we use the fully connected graph to represent the input instance, as expected the algorithm has $\mathcal{O}(|E|^2)$ time complexity (as shown in Fig. 6b). This time complexity can be reduced if instead of the fully connected graph one uses a planar graph e.g. Delaunay triangulation. We show that the results of MST-based model are comparable to humans in quality and speed, and scales 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

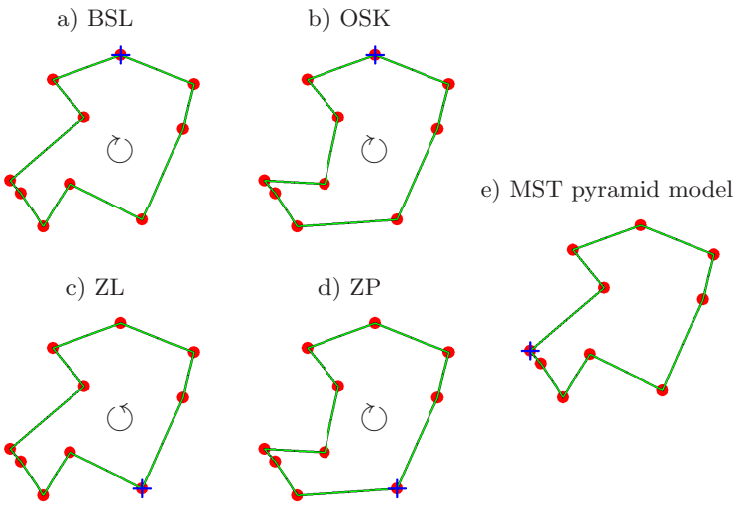


Fig. 4. E-TSP solutions by humans subjects and the MST pyramid model.

5 Conclusion

Pyramid solution strategies in a bottom-up manner convert 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.

References

1. Johnson, D.S., McGeoch, L.A.: Local Search in Combinatorial Optimization. E.H.L. Aarts and J.K. Lenstra (Eds.). In: The Traveling Salesman Problem: A Case Study in Local Optimization. John Wiley and Sons (1997) 215–310
2. Christofides, N.: Graph Theory - An Algorithmic Approach. Acad. Press, (1975)

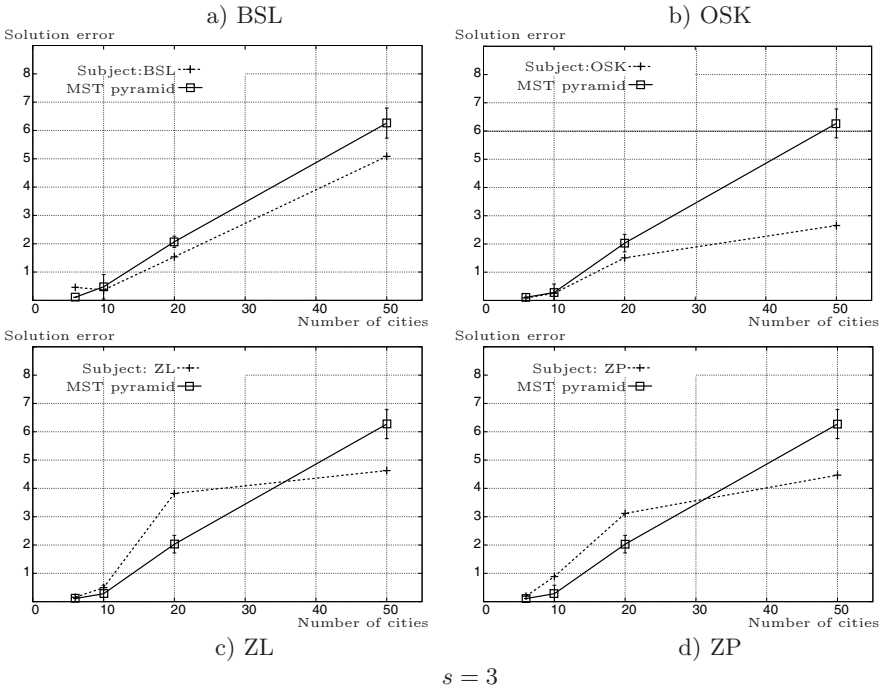
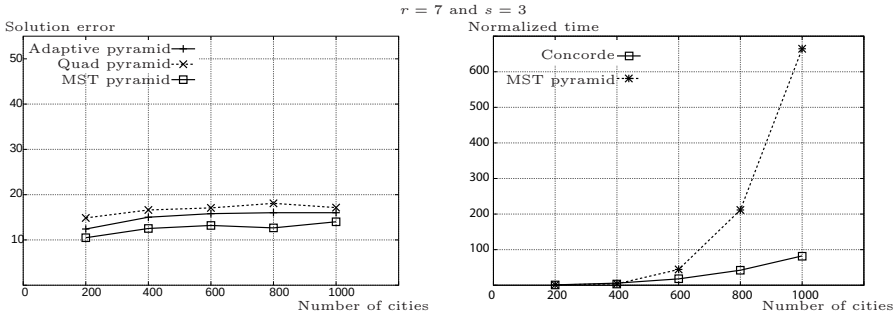


Fig. 5. Model fitting.

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 (2002)
5. Graham, S.M., Joshi, A., Pizlo, Z.: The travelling salesman problem: A hierarchical model. Memory and Cognition **28** (2000) 1191–1204
6. MacGregor, J.N., Ormerod, T.C., Chronicle, E.P.: A model of human performance on the traveling salesperson problem. Memory and Cognition **28** (2000) 1183–1190
7. Pizlo, Z., Li, Z.: Pyramid algorithms as models of human cognition. In: Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging, SPIE (2003) 1–12
8. Jolion, J.M., Rosenfeld, A.: A Pyramid Framework for Early Vision. Kluwer (1994)
9. Pizlo, Z., Joshi, A., Graham, S.M.: Problem solving in human beings and computers. Technical Report CSD TR 94-075, Purdue University (1994)
10. Arora, S.: Polynomial-time approximation schemes for euclidean tsp and other geometric problems. J. of the **ACM 45** (1998) 753–782
11. Bister, M., Cornelis, J., Rosenfeld, A.: A critical view of pyramid segmentation algorithms. Pattern Recognition Letters **11** (1990) 605–617
12. Montanvert, A., Meer, P., Rosenfeld, A.: Hierarchical image analysis using irregular tesselations. IEEE Transactions on **PAMI 13** (1991) 307–316
13. Jolion, J.M., Montanvert, A.: The adaptive pyramid, a framework for 2D image analysis. Computer Vision, Graphics, and Image Processing: Image Understanding **55** (1992) 339–348



a) Solution Error b) Time performance

Fig. 6. The solution error and the time performance.

14. Pizlo, Z., Stefanov, E., Saalweachter, J., Li, Z., Haxhimusa, Y., Kropatsch, W.G.: Traveling salesman problem: a foveating model. The J. of Problem Solving **1** (2006)
15. Prim, R. C.: Shortest connection networks and some generalizations. The Bell System Technical J. **36** (1957) 1389–1401
16. Kropatsch, W.G.: Building irregular pyramids by dual graph contraction. IEE-Proc. Vision, Image and Signal Processing **142** (1995) 366–374
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley (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** (2001) 3–36
19. Atallah, M.J., ed.: Algorithms and Theory of Computational Handbook. CRC Press (1999)
20. Kropatsch, W.G., Haxhimusa, Y., Pizlo, Z., Langs, G.: Vision pyramids that do not grow too high. Pattern Recognition Letters **26** (2005) 319–337