Approximative Graph Pyramid Solution of the E-TSP ⋆

Yll Haxhimusa [a,b,*], Walter G. Kropatsch [b], Zygmunt Pizlo [a], Adrian Ion [b]

[a] *Purdue University, College of Liberal Arts, Department of Psychological Sciences, USA*

[b] *Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing, Austria*

Abstract

The traveling salesman problem (TSP) is difficult to solve for input instances with large number of cities. Instead of finding the solution for an input with a large number of cities, the problem is transformed into a simpler form containing smaller number of cities, which is then solved optimally. Graph pyramid solution strategies, using Borůvka's minimum spanning tree step, convert, in a bottom-up processing, 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, leads to an approximate 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, and the quality of the solutions is similar to the solutions produced by humans. The graph pyramid data structures and processing strategies provide good methods for finding near-optimal solutions for computationally hard problems. Isolating processing used by humans to solve computationally hard problems is of general importance to psychology community and might lead to advances in pattern recognition.

Key words: Hierarchical Representation, Graph Pyramids, Human Problem Solving, Traveling Salesman Problem, Borůvka's Minimum Spanning Tree.
PACS:

1 Introduction

Problem solving is one of the most fundamental human cognitive abilities, which is at least as important as perception, memory, or learning. A problem has usually three components:

- known facts provided when the problem is presented,
- goal(s), and
- operations to be performed to achieve the goal.

There are several general strategies to solve problems [1]: (1) combining algorithms; (2) hill-climbing (local computations that allow finding extrema of a given cost function - *bottom-up* computations); (3) means-ends analysis (global computations that break the problem into smaller sub-problems and then solve these sub-problems - *top-down* computations), (4) working backwards, from the goal to the start; and (5) using analogies. There has been growing interest in studying how humans solve combinatorial optimization problems, with a special emphasis on the Traveling Salesman Problem (TSP) [2–4]. Traveling salesman problem (TSP) is a combinatorial optimization task of finding the shortest tour of n cities given the intercity costs. In a more formal way the goal is to find the least weight Hamiltonian cycle in a complete graph K_n. When the costs (weights) between cities are Euclidean distances, the problem is called (symmetric) 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 [5]. A straightforward approach by using brute force search would be trying all possible permutations in order to find the shortest tour. This approach 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 approximation algorithms or finding a special class of TSP problems that can be solved by polynomial time algorithms. Good approximation 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 [6–8]. Local heuristic approaches are one of the main tools to produce near optimal tours for large instances of TSP in a short time [8, Chap.5]. Both discrete

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

* Corresponding Author. Address: Purdue University, 703 Third Street, West Lafayette, IN 47907-2081, USA

Email addresses: yll@psych.purdue.edu, yll@prip.tuwien.ac.at (Yll Haxhimusa), krw@prip.tuwien.ac.at (Walter G. Kropatsch), pizlo@psych.purdue.edu (Zygmunt Pizlo), ion@prip.tuwien.ac.at (Adrian Ion).

URL: http://www.psych.purudue.edu/ yll (Yll Haxhimusa).

optimization [7,8] as well as continuous optimization [9–12] methods are used to solve the Euclidean as well as non Euclidean TSP.

Human subjects do not search the whole problem space when solving combinatorial optimization problems. It is by now well established that humans produce close-to-optimal solutions to TSP problems in time that is (on average) proportional to the number of cities [3,13,14]. This level of performance cannot be reproduced by any of the standard approximating algorithms in computer science or operations research. Some approximating algorithms produce tours whose lengths are close to that of the shortest tour, but the time complexity is substantially higher than linear. Other algorithms are relatively fast but produce tours that are substantially longer than those produced by human subjects. It has been recently shown that human subjects can also produce very good solutions to other combinatorial optimization tasks like the minimum spanning tree (MST) [15], shortest path [16], optimal stopping [17] or the 15-puzzle problem [18]. Interestingly, optimization tasks are important not only in the context of human problem solving. Other cognitive functions can also be modeled as optimization tasks: decision making, motor control and perception. It follows that using optimization tasks may actually be the best way to study human cognition in general, and problem solving, in particular.

In this work we adopt an information-processing methodology to study human problem solving. In particular, we are interested in *problems to find* (e.g. solving the TSP), and *not problems to prove* (e.g. theorem proving) [19]. In a problem to find, there is the *input data*, the *conditions*, and the task is to solve for the *unknown*. TSP can be visually presented to the subjects, as shown in Figure 1a,d, it is easy to define, and it is considered to be natural and easy to solve by subjects. The cities in this problem can be considered as being abstract representation of some interests (feature) points that can be used for example in robot navigation, or object recognition. We describe in this paper a computational model for solving E-TSP approximately. The model is based on a multiresolution graph pyramid. Previous multiresolution models [3,20] are not rotational invariant. Our new model described in this paper is invariant to both translation and rotation of the input city constellation. *The emphasis of our work is on emulating human performance (time and accuracy), and not in finding an algorithm for solving E-TSP as optimally as possible*. By accuracy we mean the quality of solution, i.e. the tour length produced by the approximation algorithm compared to the tour length solution by the optimal algorithm. We also expect the time complexity of the approximation algorithms to be comparable to the time complexity of humans. We have chosen the length of the tour as well as the time complexity as it is the goal of the TSP. In fact we use the ratio of these two tours as the error measure. Other criteria, like the shape could be used to measure the quality of contours representing TSP solutions [21,22].

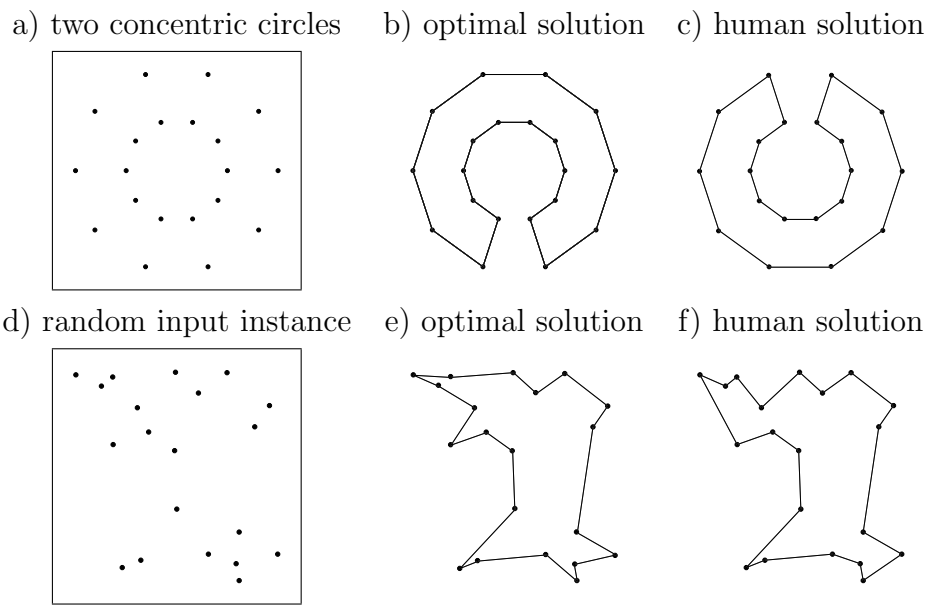


Fig. 1. 2D E-TSP and solutions given by the optimal solver and human subject.

The paper is structured as follows. The next section presents a survey of the work related to human problem solving. Section 3 gives a short overview of the pyramid representations. In Section 4 a pyramid model for solving the E-TSP is described. The model uses a minimum spanning tree (MST) based graph pyramid. The bottom-up simplification of the input data is presented in Section 4.1, and in Section 4.2 the top-down sequence of approximations of the TSP solution tour is described. Psychophysical experiments on E-TSP are presented in Section 5.

2 Status of Research

Human problem solving has been studied as an information-processing activity using different approaches. In this section, several major approaches are briefly described, with a more detailed overview of the algorithmic methods. Production systems is a commonly used formalism to represent and solve problems in cognitive psychology and artificial intelligence (see [23] for an overview). These systems require rational ('good') representations of the problem in the database, the production rules and control mechanisms. When production systems are used, one has to analyze the problem first, and then build a complete representation of the problem. Production systems have been successfully applied in mathematics (SHUNYATA [24]), physics (FERMI [25]), chemistry (MECHEM [26]), and medicine (MYCIN [27]). Most of the existing studies and methods that use productions systems focus on a particular context of application and their results cannot be easily extended to other applications. Moreover, it is not clear how the particular cognitive mechanisms are involved

and how 'good' the created representations actually are and how 'well' are they organized in the memory. Problems that are well-defined can be solved by formulating and executing an algorithm, which performs a finite number of steps. Unfortunately, for some problems this is not possible. In such cases one can formulate 'rules of thumb' and build a so-called heuristic algorithm, which may or may not lead to the solution [28]. A nice overview of an algorithmic approach to problem solving in artificial intelligence was presented by [29] and in pattern recognition by [30,31]. Despite the attractiveness of the approach based on heuristic algorithms, this approach has its difficulties: problems are usually not solved optimally (if at all) and it is not clear how to define 'the rules of thumb' for each particular problem, as well as transporting these algorithmic solutions to other problems.

Consider human and animal visual perception. Humans and animals are able to delineate, detect and recognize objects in complex scenes 'at a blink of an eye'. Authors in [32,33] performed complexity analysis and showed that hierarchical representation of visual information and hierarchical processing of this information is one of the best, if not, the best way to solve visual problems. They presented a list of critical properties of visual processing in biological systems and most of them are related to the use of a hierarchical (pyramid) architecture: (i) parallel processing, (ii) *hierarchical organization* through abstraction of prototypical visual knowledge (this leads to short search times that are proportional to the logarithm of the image size, rather than to the image size, itself) (iii) *spatially local receptive fields* (the physical world is spatio-temporally localized and events, objects, and their physical characteristics are not arbitrarily spread over time and space), (iv) *hierarchical abstraction* of semantic context from the input data, (v) direct access to higher level maps [1] based on some abstract properties of the input, (vi) prediction about which parts of the hierarchical representation should be used to analyze a given set of visual data, (vii) a *bottom-up* 'pre-attentive' processing of input information, and (viii) a *top-down* control mechanism, that allow restricting search space.

It is now commonly accepted that the human visual system has a hierarchical architecture and that the visual mechanisms can be adequately modeled by hierarchical (pyramid) algorithms. Specifically, neurophysiological and neuroanatomical data indicate that the visual systems of cats, monkeys and humans are hierarchical, with neurons on lower layers having smaller receptive fields and neurons on higher layers having larger receptive fields [34]. Pyramid algorithms are adequate models of the Gestalt rules of perceptual organization such as proximity and good continuation [35,36]. They also provide an adequate model of Weber's law, speed-accuracy trade-off in size perception, as well as mental size transformation [37]. In these models, visual processing involves both bottom-up (fine to coarse) and top-down (coarse to fine) analysis.

[1] Called layers in the image pyramid framework.

The top-down processing seems also critical in solving the image segmentation problem, which is a difficult inverse problem [38]. This problem has received a lot of attention in the psychological literature, and is known as the figure-ground organization phenomenon [39]. The human visual system is able to distinguish between highly relevant and less relevant regions in the field of view. Authors in [40,41] and [42], describe two processes: *bottom-up* processes retrieve features directly from the input image, whereas *top-down* processes are driven by available knowledge about the world.

Pyramid algorithms have been used extensively in both computer and human vision literature (e.g. [43]), but not in problem solving. The work of [3,44] is the first attempt to use pyramid algorithms to solve the visual version of the E-TSP. One of the most attractive aspects of pyramid algorithms, that makes them suitable for problems such as early vision or E-TSP, is that they allow solving (approximately) global optimization tasks without performing a global search.

The TSP tours produced by the subjects are, on average, only a few percent longer than the shortest tours (in Figure 1c,f). The solution time is a linear function of the number of cities [3,13]. In [3], authors formulated a pyramid algorithm for E-TSP motivated by the failure to identify an existing algorithm that could provide a good fit to the subjects' data. More recently, hierarchical (pyramid) algorithms have been used to model mental mechanisms involved in other types of visual problems [14,45] (e.g. 15 puzzle problem, MST etc.). The main aspects of the models in [3,14] are

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

A foveating algorithm for TSP [20] emulates the visual attentional mechanisms of humans. This algorithm was motivated by the properties of the human visual system. Specifically, the human retina provides high resolution representation only in the center of the visual field. The computational complexity of these models was very low and similar to that characterizing mental processes (i.e. linear). At the same time, the solutions produced by these models were characterized by similar errors as those produced by the subjects.

Hierarchical partitioning is one of the best known methods of clustering in a top-down direction [30,46], while agglomerative clustering works in the bottom-up direction. The minimum spanning tree (MST) can be used to perform the latter [30]. In [47] it was shown that the MST approaches the performance of the Bayes classifier as the number of data points goes to infinity.

Greedy algorithms and the nearest neighbor algorithm, based on computational experiments, are good choices for tour construction heuristics, and produce *acceptable results* for the Euclidean TSP, but are very poor for the general

symmetric and asymmetric TSP [48]. In spite of this limitation, and since we are studying E-TSP we have used a greedy method (MST). On the other side, Pizlo et. al. [44] showed that the nearest neighboring algorithm is not best suited to model human subject.

3 Irregular Graph Pyramid

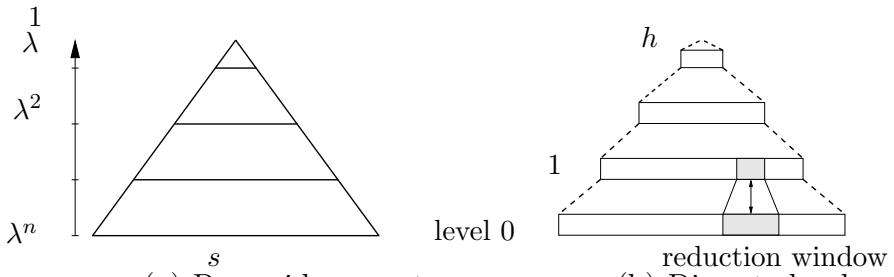
Considering the fact that the problem spaces are extremely large (the number of tours in a 17-city TSP is larger than the number of neurons in the human brain), the problem solver must be able to:

- efficiently represent the problem and organize information about it,
- avoid brute force, global search of the problem space, and
- use top-down approach in which global, abstract knowledge can guide construction of the solution.

The graph pyramid framework fulfills these conditions. In our framework, the TSP input is represented by graphs where cities are represented by vertices, and the intercity neighborhoods by edges. Note that based on the definition of the intercity neighborhoods one creates dense (e.g. complete [2]) or sparse graphs. 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, $G = (V, E, w_v, w_e)$, where $w_e : E \rightarrow \mathbb{R}^+$ is a weighted function defined on edges E. The weights w_e are Euclidean distances in the E-TSP and $w_v : V \rightarrow \mathbb{R}^+$ is a weighted function defined on cities V. 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 an 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 the top of G is equal to $\log_\lambda(|G|)$ (Figure 2). This implies that a pyramid is build in $\mathcal{O}[\log(diameter(G))]$ parallel steps [43]. Reduction windows relate one cell at the reduced level with a set of cells in the level directly below. Thus local independent (and parallel) processes propagate information up and down and laterally in the pyramid. The contents of a lower resolution cell is computed by means of a reduction function the input of which are the descriptions of the cells in the reduction window. Two successive levels of a pyramid are related by the reduction window and the reduction factor. Higher level description should be related to the original

[2] In our implementation in this paper, we use fully connected graphs.



(a) Pyramid concept (b) Discrete levels

Fig. 2. Multiresolution pyramid.

input data in the base of the pyramid. This is done by the receptive field (RF) of a given pyramidal cell c_i. The $RF(c_i)$ aggregates all cells (pixels) in the base level, of which c_i is the ancestor. Regular image pyramids are confined to globally defined sampling grids and lack shift invariance [49]. In [50,51] it is shown how these drawbacks can be avoided by adaptive irregular pyramids.

In Graham's model [3], clusters were 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 [20] used an adaptive model in which adaptive top-down partitioning of the plane along the axis of the Cartesian system was used. The hierarchy is represented by a binary tree. Although this algorithm is invariant to translations, it was not invariant to rotations. Our new model described in this paper uses graphs as representation, which are invariant to both translation and rotation of the input city constellation. Our clustering, however, is performed in a bottom-up direction.

4 Solving E-TSP by a Graph Pyramid

Let $G_0 = (V, E, w_v, w_e)$ be the input graph, with weights on edges given as distances in L_2 space. The goal of the E-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 and the sum of edge weights in this tour is minimal, i.e. the tour length is minimized. This tour is called the optimal tour τ_{opt}, and its length is

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

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 optimal tour of 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 so that an optimal (trivial) solution can be found by the combinatorial search. For example, for a 3 city instance (not all cities are collinear) there is only one tour, and this tour is obviously optimal. It follows that no search is needed. For a 4 city input (no three of which are collinear) there are three solutions, from which two non-optimal can be easily rejected since they contain self-intersection. 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 the movements of the human fovea using lower levels of this pyramid. The solution tour, in general non-optimal, 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. Creating graphs, i.e. the input space was presented in Section 3. Sections 4.1 and 4.2 discuss steps 2 and 4 of Algorithm 1 in more detail.

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

Input: Graph $G_0 = (V, E, w_v, w_e)$, 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 s vertices:
build graph pyramid $G_k, \forall k = 0, ..., h,$ *where* $s = |G_h|$
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 *of the pyramid*(G_0) *is reached*

Output: Approximate TSP solution τ^*.

4.1 Bottom-up Simplification using an MST Pyramid

In bottom-up clustering, cities that are close neighbors are put into the same cluster, using a greedy approach. These clustered cities are considered as a single city at the 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 levels of scale and resolution [52,37].

We are driven to use this closeness concept since it seems that humans use this cue 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, it seems that there might be some organization of the input data during the E-TSP solving process.

There are many different algorithms for hierarchical clustering of points, i.e.

cities [30]. Our model uses the minimum spanning tree, specifically Borůvka's algorithm [53]. The time complexity of Borůvka's algorithm is $\mathcal{O}(|E| \log |V|)$. MST can be used as the natural lower bound for TSP solutions. In the case of the TSP with the triangle inequality, which is the case for the E-TSP, MST can be used to prove the upper bound as well [54]. The first step in Christofides' heuristic [6] is finding an MST as an approximation for TSP. Then, one can produce a tour, whose length is never greater than $\frac{3}{2}$ times of the length of the shortest tour.

For a given graph $G_0 = (V, E, w_v, w_e)$ the vertices are hierarchically grouped into trees as given in Algorithm 2. The idea of a Borůvka step is to perform greedy operations like in Prim's algorithm [56], in parallel, over the entire graph. The trees (clusters) are not allowed to contain more than $r \in \mathbb{N}^+$ cities. Each tree must contain at least 2 cities, in order to ensure that the pyramid has a logarithmic height [57]. It follows that the reduction factor λ is $2 \leq \lambda \leq r$. It seems that this parameter is related to the number of 'concepts' that humans can hold in their 'short-term memory' (Millers number seven, plus or minus two [58]).

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

An example of how Algorithm 2 builds the graph pyramid (only the last two levels) is shown in Figure 3. 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. Since we employ Borůvka step at this stage, it might happen that the trees will contain more than allowed r cities. In these cases the algorithm will break the trees into smaller trees (Figure 4). We employ a

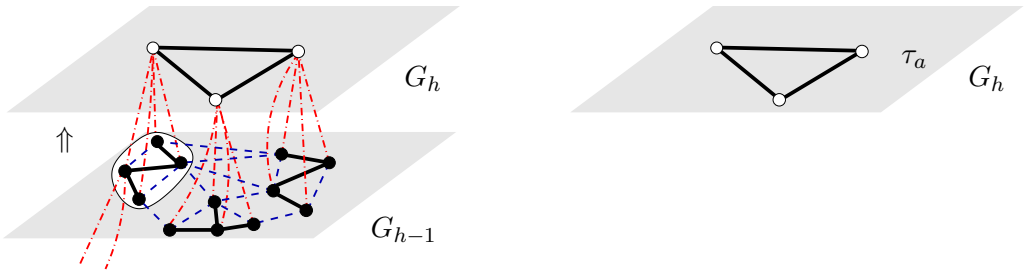
Algorithm 2 – Reduction of the E-TSP Input by an MST Graph Pyramid

Input: Graph $G_0 = (V, E, w_v, w_e)$, 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 r vertices
5: **if** the size $|T| > r$ **then**
6: break trees into sizes $\leq r$ /* for e.g. using [55] */
7: contract trees T into parent vertices v_{k+1}
8: create graph G_{k+1} with vertices v_{k+1} and edges $e_k \in G_k \setminus T$
9: attribute vertices in G_{k+1}
10: $k \leftarrow k + 1$
11: **until** there are s vertices in the graph G_{k+1}.

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

simple breaking strategy by simply removing the vertex (or vertices) from the tree that correspond to the edge(s) with the heaviest weight(s). This vertex is then merged to the nearest cluster having less than r cities. Note that, in case there is no cluster in the 'immediate' neighborhood to merge this 'broken' vertex, one allows that this vertex be merged with a cluster that is 'far', thus allowing for crossing in the resulted tour. A better strategy of breaking this trees is to break them in the 'middle' of the tree by first computing the diameter of the tree [55]. After the process of creating the trees is finished, 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 center of gravity of its child vertices, or the position of the vertex near the center of gravity. 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.



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

Fig. 3. 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. It follows that the bottom-up simplification algorithm has at least $\mathcal{O}(|V|^2)$ time complexity [59], which dominates over other terms: breaking of trees, creating the new level of the pyramid etc. This time complexity can be reduced easily to $\mathcal{O}(|E| \log |V|)$ if instead of the fully con-

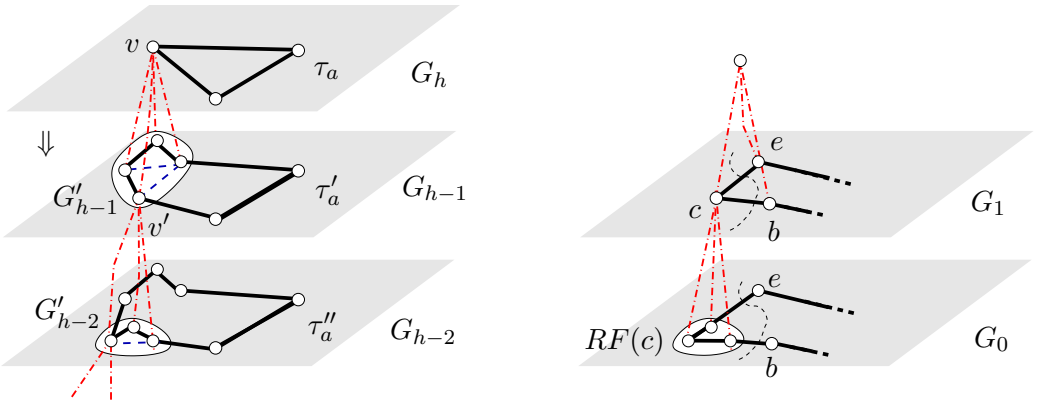
a) breaking the vertex in white b) connecting the white vertex to another tree

Fig. 4. Breaking the trees and creating trees with sizes $r = |V_t| \leq 3$.

nected graph one uses a planar graph e.g. by Delaunay triangulation. In our experimental setup we have not encountered problems with the simple tree breaking strategy, but in general it can cause that some tours will have self intersection. We position the parent vertices in the center of gravity of its children 'mass'.

4.2 Top-down Approximation of the Solution

The tour τ_a found at the 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. [20] we use the simplest 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 relation, 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 Figure 5a). Since the number of vertices (children) in G'_h cannot be larger than r, a complete search is a reasonable approach to find the path with the smallest contribution in the overall length of the tour τ'_a. Note that the edges in τ'_a are not necessarily the contracted edges during bottom-up construction.



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

Fig. 5. Refining the E-TSP tour by a graph pyramid.

The refinement process then chooses (randomly) 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 recursively until there are no more parent-children relation (graph G_0, Figure 5b vertices of the reduction window of c, $RF(c)$), i.e. vertices at the base of the pyramid are reached. E.g. in Figure 5b, the tour is refined by choosing the shortest path between the start vertex b and end vertex e and going

through 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 5 vertex b or e), and expanding it to its children. 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 representation, until the 'attention' is moved to them. The process converges when all vertices in the pyramid have been 'visited'. More formally the steps are depicted in Algorithm 3, and Function 1 and 2.

The complexity of the top-down process depends on the choice of the parameter r. The number of levels of the pyramid depends on r and is $h = \log_r(|V|)$, thus the overall number of vertices in the pyramid is $(1 + \frac{1}{r} + \frac{1}{r^2} + \cdot + \frac{1}{r^h})|V| \leq 2|V|$, since $r > 1$. In our current implementation the search time within each cluster is done in constant time. Finding the best path within r cities in an exhaustive search is $r!$. Thus the overall time complexity of the top-down process is $\mathcal{O}(c|V|)$, where c is a constant. The parameter r has to be chosen as a compromise between the quality of the tour produced and the amount of search performed. In our experiments $r \leq 10$ led to results similar to those produced by human subjects.

Other refinement approaches can be chosen as well. For example, one can use many vertices and expand them in parallel (multi-path refinement), or use the one-path refinement until a particular level of the pyramid is reached and continue with the multi-path refinement afterward. In these cases Function 1 would change. Note that any of the vertices can be chosen as the starting point for the solution process. Different starting points are likely to lead to different tours. The choice of the starting point may account, at least in part, for individual differences in human solutions. During the psychophysical experiments, subject had no clear criteria from where to start the solution. It seems to be a random choice. We have used this random criteria in our algorithm as well.

A full example showing the steps of Algorithm 1 (on a problem from Section 5) is given in Figure 6. First step is creating a fully connected graph, where each city is connected to each city (not shown in the figure for clarity). On this graph the Borůvka step (and the breaking of trees to the given size of r, if necessary) is performed (shown in figure under b). Thus after the first running of the Borůvka step one gets the trees shown in b1). Each of the trees is then merged to the parent vertices. The parent vertex is positioned on the gravity center of his children. These new vertices are then fully conected to each other, thus creating a new graph on the second level of the pyramid. The process is repeated iteratively until a level of the pyramid is reached (top of the pyramid shown under b3), where the number of vertices is small ($|V_2| = s = 4$ in this example), and the process of building the pyramid

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

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 Function 1 */
5: mark v as visited
6: $v \leftarrow$ nextVertex(G_k, v, τ^*) /* get next vertex to process. See Function 2 */
7: **until** $v = \emptyset$

Output: Approximation E-TSP tour τ^*.

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

Input: A 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^*.

Function 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^*

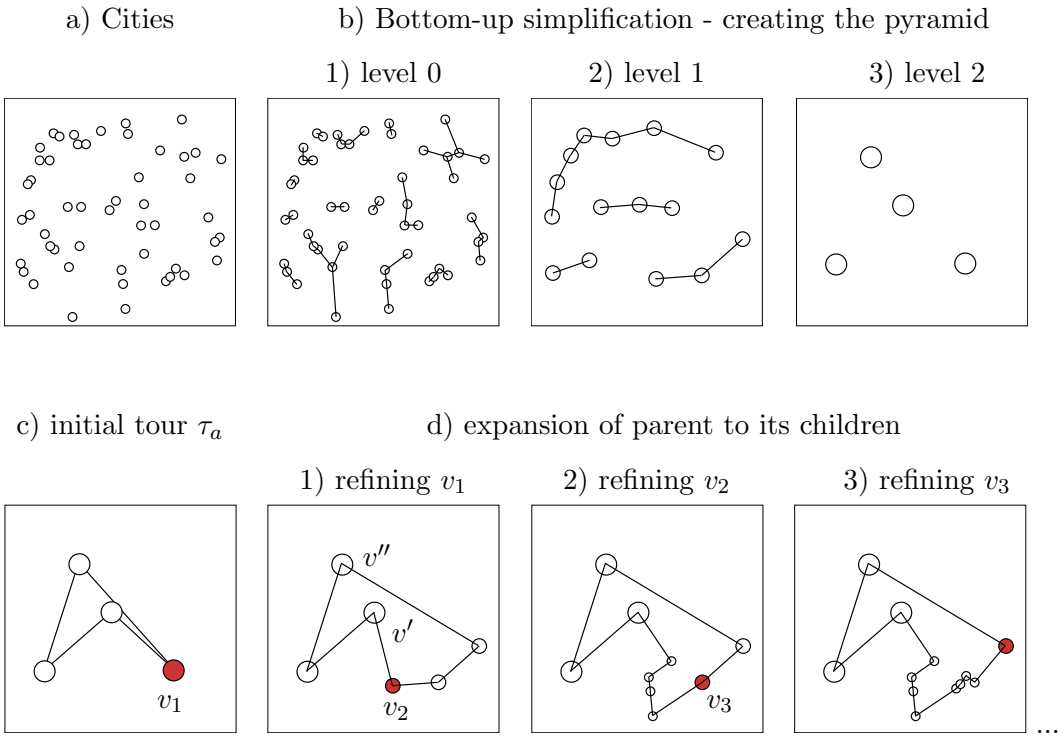
1: **repeat**
2: **if** v has unvisited children **then**
3: $v \leftarrow$ first unvisited child of v in τ^* /* given an orientation */
4: **else if** v has unvisited siblings **then**
5: $v \leftarrow$ first unvisited sibling of v in τ^* /* given an orientation */
6: **else if** v has a parent i.e. 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.

---

is stopped. Note the size of the vertices in the figure, corresponding to their bigger and bigger receptive fields going to higher levels of the pyramid. Having a small number of cities at the top of the pyramid the first approximated tour is created by a brute force search (as shown in figure under c). A vertex is chosen randomly (filled vertex shown under c, belonging to level 2) and refined by expanding it to its children (shown under d1). The shortest path by using a brute force search is found between v' and v'' and the expanded children (d1). After the approximated tour is found one of the vertices (chosen randomly) belonging to level 1 (v' and v'' are not in level 1, note the size of these vertices) is refined (shown filled in d1) to its children, and the tour is

found as previously explained. Note that we have vertices from all levels of the pyramid, mimicking human visual fovea. The process is iterated until all the input cities (vertices on level 0) have been added to the tour. The full demo can be found in http://www.prip.tuwien.ac.at/twist/results.php.

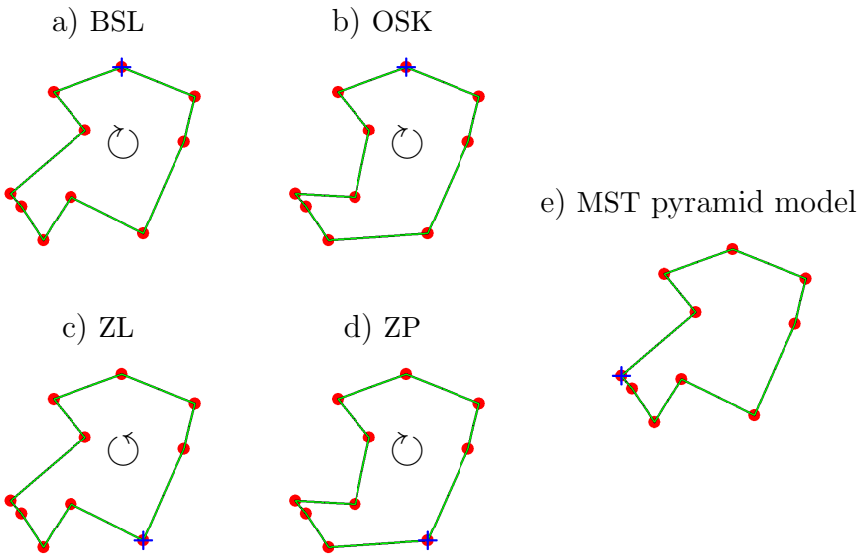


Top-down refinement of tours τ^* simulating the human fovea.

Fig. 6. Steps of the algorithm.

5 Psychophysical Evaluation of the Model

Performance of this model was compared to that of four human subjects. It is known that there is a very small individual variability in performance and that all subjects solve TSP very well [3,60,20]. Therefore, it is sufficient to test only a few subjects, each subject being tested on large number of problems. This is actually how research on human visual perception is being done (see human vision journals such as Vision Research [61], Journal of Vision [62] or Journal of Problem Solving [63]).



Random instance with 10 cities.

Fig. 7. E-TSP solutions by humans subjects and the MST pyramid model.

5.1 Subjects

Four subjects were tested (see Pizlo et al. [20], for a full description of the experiment). Subjects BSL, and OSK were naïve about the purpose of the experiment. They received only a small amount of practice before being tested. ZL and ZP, on the other hand, were quite familiar with TSP. Specifically, they solved hundreds of instances of TSP before being tested.

5.2 Stimuli

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 as short a tour as possible by clicking on the points. An example of the stimuli is shown in Figure 1a,d.

5.3 Procedure

Each subject solved the same 100 E-TSP problems in a different order. There were 4 different problem 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 [14]. Examples of 10 city tours produced by the subjects and by the model are presented in Figure 7. The crosses depict the starting

point chosen by the subjects and by the model. BSL, OSK, and ZP chose the clockwise direction when solving the problem, whereas ZL chose the counter-clockwise direction. The MST based pyramid model chose randomly both the starting point and the direction in individual solutions. The algorithm is applied to each problem with the values of r in the range between 2 and 9. The tour whose length was closest to the length produced by a given subject was taken for further analysis.

The tour length is measured by the solution error

$$\epsilon = (\frac{l(\tau_a)}{l(\tau_{opt})} - 1) \cdot 100\%$$

5.4 Results and Discussion

Figure 8a–d show average performance of each subject and that of the model. It can be seen that fits 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. Recall that the pyramid tries only one, randomly chosen, starting city. If more than one starting city were tried, the fit would have been better. The subjects' choice of starting city is not completely random, but at this point we do not have a model for choosing a good (or best) starting point. Figure 8e shows the average of the estimated parameter r, i.e. the number of cities searched by humans during the solutions. The individual variability in the estimated r is not large, which also confirms the result from [20]. The subject who produced better solutions (OSK) tended to have higher values of r.

Some authors suggested that subjects 'select' their tour from the set of non-self-intersecting tours [60]. The number of self-intersecting tours depends on the distribution of cities. For a random distribution of cities, the restriction that the tour is non-self-intersecting is not very tight: there are still a large number of non-self-intersecting tours [20]. In our simulation our model never produced self intersecting tour.

Next, the algorithm was run 15 times on the whole set by choosing random refinement points during the tour approximation and with different parameters r ($4 \leq r \leq 8$). In Figure 9a) the average of the subjects taken from Figure 8 and that of the model for parameter $r = 7$ is shown. The averaged results of the model for several values of r is depicted in Figure 9b).

For larger instances (> 120 cities) data with human subjects is difficult to obtain because the experiment becomes too long. Therefore we compared Al-

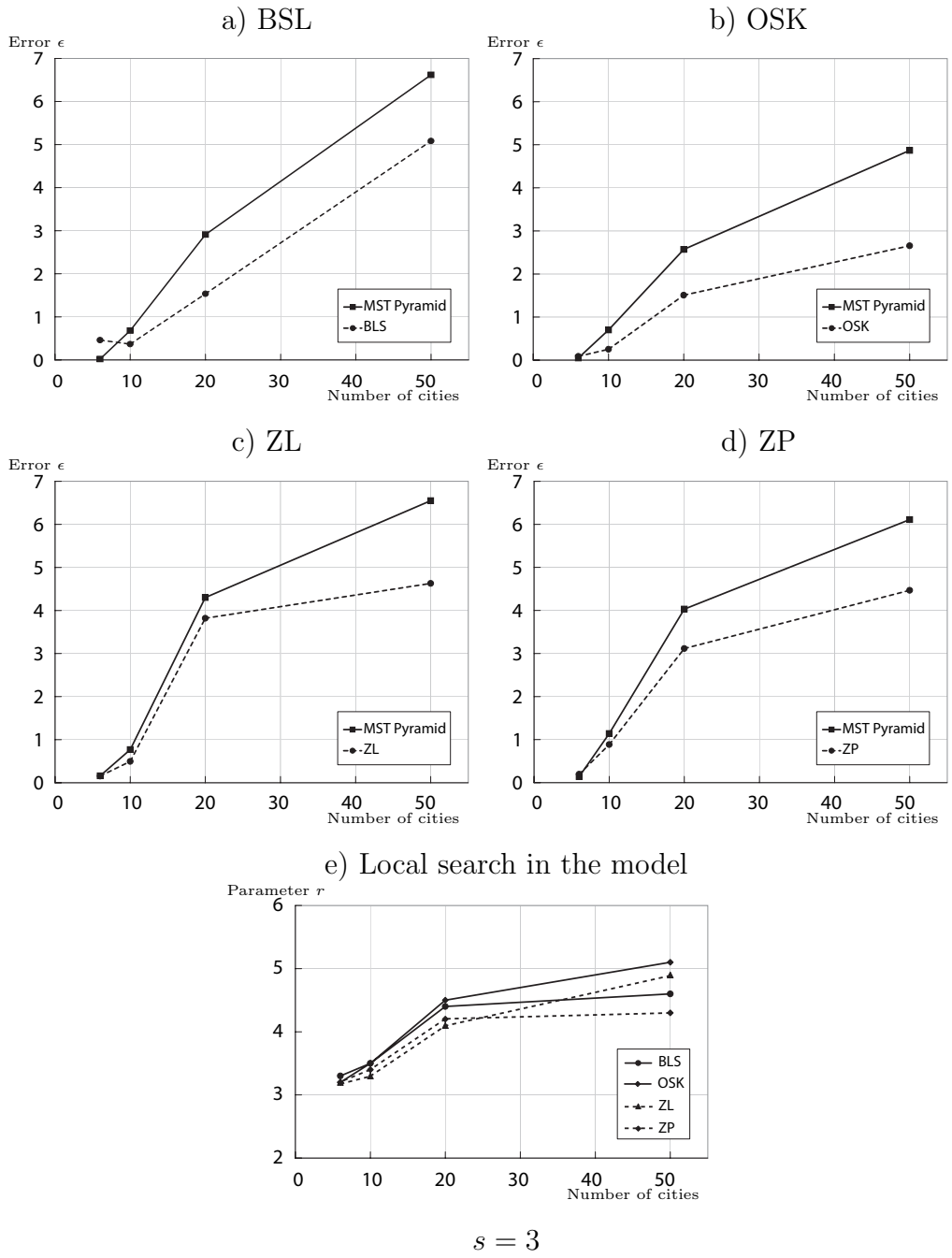
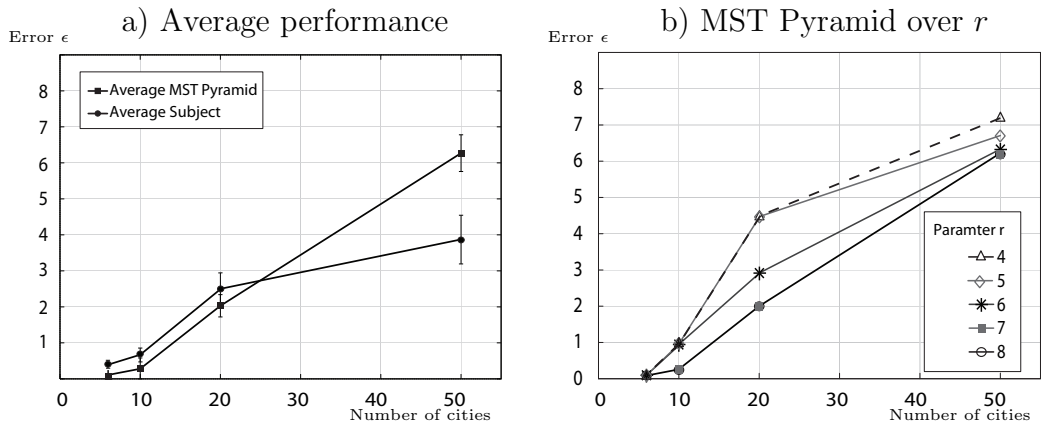


Fig. 8. a)-d)Model fitting on human data and e) estimated value of r for individual subjects.

gorithm 1 with the state-of-the-art Concorde TSP solver [3], with adaptive pyramid [20] and quad pyramid [14]. We have fixed the values of the parameter $r = 7$ and $s = 3$ for these experiments. The comparison involved average error and time needed to solve TSP problems with 200, 400, 600, 800, and 1000 cities. The errors are shown in Figure 10a and time in Figure 10b. It can be

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

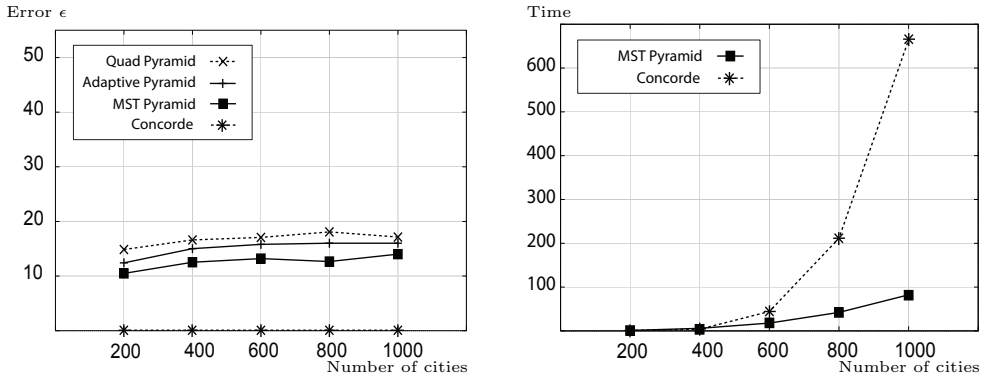


$s = 3$

Fig. 9. a) Average performance of MST Pyramid vs. average human, and b) average error of the pyramid model with different parameter r over the test set.

seen in Figure 10a that the MST based model slightly outperforms the other pyramid models. Note that the Concorde algorithm solves the problems optimally, i.e. the error is always zero. The time plot in Figure 10b was normalized to the time needed to solve the 200 city instance - this time was set to one second. It is shown that MST based model is substantionaly faster than the Concorde algorithm for larger problems.

To summarize, Figures 8 and 9 show that the results of the MST-based model are comparable to those of humans. The algorithm scales well for large input instances (Figure 10). The pyramid model provides a plausible explanation of how human subjects solve computationally hard problems by trading off time and quality of solution.



$r = 7$ and $s = 3$

a) Solution Error b) Time performance

Fig. 10. The solution error and the time performance.

6 Conclusions

Pyramid strategies 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 find the optimal solution. Expanding this solution in a top-down manner to the lower levels of the pyramid approximates the solution. The proposed 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. 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 closely fits the human data. The presented model has a low computational complexity similar to that characterizing mental processes (i.e. near-linear). The solutions produced by the model is characterized by errors similar to those produced by the subjects. In many areas of pattern recognition, like object detection, image segmentation etc. humans are used as the 'gold standard'. Isolating the process behind the solutions given by the humans is in general of importance to psychology community and might lead to advance in pattern recognition as well. Pyramid data structures and processing strategies are a reasonable approach for finding near-optimal solutions not only for NP-hard problems such as TSP, but also for pattern recognition problems, such as matching.

Acknowledgment
The authors would like to thank the anonymous reviewers for their valuable comments. We also thank Andreas Lehrbaum and Joe Catrambone for their help in performing the experimental results.

References

[1] J. E. Ormrod, Human Learning, Prentice Hall, 1999.

[2] J. MacGregor, T. Ormerod, Human performance on the traveling salesman problem., Perception & Psychophysics 58 (1996) 527–539.

[3] S. M. Graham, A. Joshi, Z. Pizlo, The travelling salesman problem: A hierarchical model., Memory and Cognition 28 (7) (2000) 1191–1204.

[4] D. Vickers, M. Butavicius, M. Lee, A. Medvedev, Human performance on visually presented traveling salesman problems., Psychological Research 65 (2001) 34–45.

[5] D. S. Johnson, L. A. McGeoch, The Traveling Salesman Problem: A Case Study in Local Optimization., John Wiley and Sons, 1997, Ch. Local Search in

Combinatorial Optimization. E.H.L. Aarts and J.K. Lenstra (Eds.), pp. 215–310.

[6] N. Christofides, Graph Theory - An Algorithmic Approach., Academic Press, New York, London, San Francisco, 1975.

[7] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, The Traveling Salesman Problem., Wiley, New York, 1985.

[8] G. Gutin, A. P. Punnen, The traveling salesman problem and its variations., Kluwer, 2002.

[9] A. L. Yuille, Generalized deformable models, statistical physics, and matching problems., in: Neural Computation, Vol. 2, 1990, pp. 1–24.

[10] W. Wolfe, M. Parry, J. MacMillan, Hopfield-style neural networks and the tsp, in: 1994 IEEE International Conference on Neural Networks, 1994. IEEE World Congress on Computational Intelligence., Vol. 7, 1994, pp. 4577–4582.

[11] J. Andrews, J. A. Sethian, Fast marching methods for the continuous traveling salesman problem., Proceedings of the National Academy of Sciences of the United States of America. 104 (4) (2007) 1118–1123.

[12] J. A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, 2006.

[13] J. N. MacGregor, T. C. Ormerod, E. P. Chronicle, A model of human performance on the traveling salesperson problem., Memory and Cognition 28 (2000) 1183–1190.

[14] Z. Pizlo, Z. Li, Pyramid algorithms as models of human cognition., in: Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging, SPIE, 2003, pp. 1–12.

[15] Z. Pizlo, E. Stefanov, J. Saalweachter, Y. Haxhimusa, K. Kropatsch, Adaptive pyramid model for the traveling salesman problem., in: 1st International Workshop on Human Problem Solving, Purdue University (http://psych.purdue.edu/tsp/workshop/program.html), 2005.

[16] J. MacGregor, Solution performance and heuristics in closed and open versions of 2d euclidean tsps., in: 1st International Workshop on Human Problem Solving, Purdue University (http://psych.purdue.edu/tsp/workshop/program.html), 2005.

[17] M. Lee, A generative model of human performance on an optimal stopping problem., in: 1st International Workshop on Human Problem Solving, Purdue University (http://psych.purdue.edu/tsp/workshop/program.html), 2005.

[18] Z. Pizlo, Z. Li, Solving combinatorial problems: The 15 puzzle problem., Memory and Cognition 33 (6) (2005) 1069–1084.

[19] G. Poyla, How to Solve it: A New Aspect of Mathematical Method, Doubleday and Co., Inc, Garden City, NY, 1957.

[20] Z. Pizlo, E. Stefanov, J. Saalweachter, Z. Li, Y. Haxhimusa, W. G. Kropatsch, Traveling salesman problem: a foveating model., Journal of Problem Solving 1 (1) (2006) 83–101.

[21] J. Giesen, Curve reconstruction, the tsp, and menger's theorem on length., in: Proceedings of the 15th Annual ACM Symposium on Computational Geometry., 1999, pp. 207–216.

[22] H. Edelsbrunner, D. G. Kirkpatrick, R. Seidel, On the shape of a set of points in the plane., IEEE Transaction on Information Theory 29 (4) (1983) 71–78.

[23] M. Wagman, Problem-Solving Processes in Humans and Computers, Praeger, 2002.

[24] K. Ammon, An automatic proof of Gödel's incompleteness theorem., Artificial Intelligence 61 (2) (1993) 291–306.

[25] J. H. Larking, F. Reif, J. Carbonell, A. Gugliotta, Fermi: A flexible expert reasoner with multi-domain inferencing., Cognitive Sciences 12 (1988) 101–138.

[26] R. E. Valdés-Perés, Some recent human-computer discoveries in science and what accounts for them., AI Magazine 16 (3) (1995) 37–44.

[27] E. H. Shortliffe, Computer-based Medical Consultations: MYCIN, American Elsevier, New York, 1976.

[28] A. Newell, H. A. Simon, Human Problem Solving., Prentice Hall, 1972.

[29] S. Russel, P. Norvig, Artificial Intelligence: A Modern Approach., Prentice Hall, 1995.

[30] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, John Wiley & Sons, 2001.

[31] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[32] J. K. Tsotsos, Analyzing vision at the complexity level., Behavioral and Brain Sciences 13 (3) (1990) 423–469.

[33] J. K. Tsotsos, On the relative complexity of passive vs active visual search., International Journal of Computer Vision 7 (2) (1992) 127–141.

[34] S. Zeki, A Vision of the Brain, Oxford: Blackwell, 1993.

[35] Z. Pizlo, M. Salach-Golyska, A. Rosenfeld, Curve detection in a noisy image., Vision Research 37 (9) (1997) 1217–1241.

[36] Z. Pizlo, Perception viewed as an inverse problem., Vision Research 41 (24) (2001) 3145–3161.

[37] Z. Pizlo, A. Rosenfeld, J. Epelboim, An exponential pyramid model of the time-course of size processing., Vision Research 35 (8) (1995) 1089–1107.

[38] C. Bouman, B. Liu, Multiple resolution segmentation of textured images., IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (2) (1991) 99–113.

[39] K. Koffka, Principles of Gestalt Psychology., Harcourt, NY, 1935.

[40] L. W. Stark, C. M. Privitera, Top-down and bottom-up image processing., in: IEEE Proceeding of International Conference on Neural Networks, Vol. 4, 1997, pp. 2294 –2299.

[41] C. M. Privitera, L. W. Stark, Algorithms for defining visual regions-of-interest: Comparison with eye fixations., IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (9) (2000) 970–982.

[42] D. Chernyak, L. Stark, Top-down guided eye movements., IEEE Transactions on Systems, Man, and Cybernetics 31 (4) (2001) 514–522.

[43] J.-M. Jolion, A. Rosenfeld, A Pyramid Framework for Early Vision., Kluwer, 1994.

[44] Z. Pizlo, A. Joshi, S. M. Graham, Problem solving in human beings and computers., Tech. Rep. CSD TR 94-075, Department of Computer Sciences, Purdue University (1994).

[45] Z. Pizlo, Z. Li, Graph pyramids as models of human problem solving., in: Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging, SPIE, 2004, pp. 205–215.

[46] J. Lance, W. Williams, A general theory of classificatory sorting strategies: I hierarchical systems, Journal on Computing 9 (1967) 373–380.

[47] N. Chowdhury, C. Murhty, Minimal spanning tree based clustering technique: Relationship whith bayes classifier., Pattern Recognition 30 (11) (1997) 1919–1929.

[48] G. Gutin, A. Yeo, A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp., Discrete Applied Mathematics 117 (1-3) (2002) 81–86.

[49] M. Bister, J. Cornelis, A. Rosenfeld, A critical view of pyramid segmentation algorithms., Pattern Recognition Letters 11 (9) (1990) 605–617.

[50] A. Montanvert, P. Meer, A. Rosenfeld, Hierarchical image analysis using irregular tesselations., IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (4) (1991) 307–316.

[51] J.-M. Jolion, A. Montanvert, The adaptive pyramid, a framework for 2D image analysis., Computer Vision, Graphics, and Image Processing: Image Understanding 55 (3) (1992) 339–348.

[52] R. J. Watt, 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 (1987) 2006–2021.

[53] J. Neštřil, E. Miklovà, H. Neštřilova, Otakar Borůvka on minimal spanning tree problem translation of both the 1926 papers, comments, history., Discrete Mathematics 233 (2001) 3–36.

[54] M. J. Atallah (Ed.), Algorithms and Theory of Computational Handbook, CRC Press, 1999.

[55] W. G. Kropatsch, Y. Haxhimusa, Z. Pizlo, Integral trees: Subtree depth and diameter., in: R. Klette, J. Žunic (Eds.), International Workshop on Combinatorial Image Analysis 2004, Vol. 3322 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2004, pp. 77–87.

[56] R. C. Prim, Shortest connection networks and some generalizations., The Bell System Technical Journal 36 (1957) 1389–1401.

[57] W. G. Kropatsch, Y. Haxhimusa, Z. Pizlo, G. Langs, Vision pyramids that do not grow too high., Pattern Recognition Letters 26 (3) (2005) 319–337.

[58] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information., Psychological Review 63 (1956) 81–97.

[59] C. H. Papadimitiou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity., Dover Publication Inc., 1998.

[60] I. van Rooij, U. Stege, A. Schactman, Convex hull and tour crossings in the euclidean traveling salesperson problem: Implications for human performance studies., Memory & Cognition 31 (2003) 215–220.

[61] J. of Vision., Website., http://www.journalofvision.org/.

[62] V. Research., Website., http://www.elsevier.com/locate/visres.

[63] J. of Problem Solving., Website., http://docs.lib.purdue.edu/jps/.