Multiple-resolution clustering for recursive divide and conquer

Steven E. Noel and Harold H. Szu
Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA 22448

ABSTRACT

In recent work, a recursive divide-and-conquer approach was developed for path-minimization problems such as the Traveling Salesman Problem (TSP). The approach is based on multiple-resolution clustering to decompose a problem into minimally-dependent parts. It is particularly effective for large-scale, fractal data sets, which exhibit clustering on all scales, and hence at all resolutions. This leads to the application of wavelets for performing the necessary multiple-resolution clustering. While the general topic of multiple-resolution clustering via wavelets is relatively immature, it has been explored for certain specific applications. However, nothing in the literature addresses the specific type of multiple-resolution clustering needed for the divide-and-conquer approach. That is the primary goal of this paper.

Keywords: multiple-resolution clustering, recursive divide and conquer, traveling salesman problem, multiresolution analysis, path minimization, combinatorial optimization, theorem of orthogonal division errors, random fractals, point processes, unsupervised clustering.

1. INTRODUCTION

Recent work has proposed a recursive approach to combinatorial optimization, based on the general principle of divide and conquer.[3] The approach is particularly applicable to path minimization problems such as the TSP. TSP-like problems arise frequently, and strike at the core of the limits of computation. Successful approaches to such problems are therefore critical in our increasingly computer-intensive society.

The recursive divide-and-conquer-approach relies on detecting, at various resolutions, the inherent clustering in data. It appears to have the greatest advantage for larger data sets which contain clusters at many resolutions. The approach was inspired by work with fractal data, which does exhibit clustering at all resolutions (scales). The data sets of interest are essentially random (multi)fractal point processes. Of course, random fractal phenomena, including point processes, are ubiquitous in nature. The multiple-resolution divide-and-conquer approach therefore has potentially broad application.

Mandelbrot's work generated great interest in fractal models.[15] Our understanding of fractals has since become fairly comprehensive.[16,19,20] Fractal models for continuous signals such as $1/f$ noise have been well studied, [17] including developments in modeling fractal signals with wavelets.[18] In fact it appears that wavelets are a natural way to represent fractal signals. While classical random point processes such as the Poisson are well understood,[24] it is only more recently that strictly fractal (non-stationary) models for point processes have been fully developed.[22] There has also been recent progress in wavelet approaches to fractal point processes.[21] It appears that a wavelet representation may be as natural for fractal point processes as for fractal signals.

Wavelets have also been applied to phenomena which are essentially fractal point processes, such as the distribution of matter in the universe.[23] The applications typically attempt to understand structure or features at various resolutions. Such features could often be considered clusters. The concept of multiple-resolution clustering also appears in the literature for other than point processes. Examples are found in perceptual grouping,[12] texture analysis,[13] and classification of magnetic resonance imagery.[14]

However, despite the progress for specific applications, there does not seem to be a well-developed, general approach to the formation of clusters at multiple-resolutions, particularly by way of wavelets. There is therefore relatively little to rely on in

Further author information -
S.E.N.: Email: snoel@nswc.navy.mil
H.H.S.: Email: hszu@nswc.navy.mil

SPIE Vol. 3078 • 0277-786X/97/$10.00

trying to do multiple-resolution clustering for the divide-and-conquer approach. The goal of this paper is to begin to make progress in that direction.

Of course, there are a large number of existing clustering algorithms in the literature, both supervised and unsupervised. This paper makes no attempt to review them, nor make comparisons to any particular ones – that is left for future work. We do note however that the recursive divide-and-conquer approach would benefit from nested, unsupervised clustering, which wavelet clustering seems to exhibit. The wavelet approach also has the advantage of being based explicitly on resolution, in particular on multiple resolutions. This is precisely the form of clustering needed by divide and conquer.

2. RECURSIVE DIVIDE AND CONQUER

The general principle of divide and conquer is a very obvious and frequently successful approach to problem solving. The idea is to divide a large problem into pieces, solve the divided sub-problems separately, then combine them into a full solution. When this approach yields optimal solutions, we say the problem is additive or linear, in the sense that solutions are simply the sums of sub-solutions. While this linearity sometimes holds for the TSP, in general it does not. This, along with the factorially increasing number of possible solutions with respect to problem-instance size, forces a compromise between algorithmic complexity and solution optimality.

On one extreme are traditional methods such as dynamic programming and branch and bound.[1] These enumerative techniques guarantee optimality, but can achieve no better than $O(n2^n)$ complexity. They are therefore infeasible for larger problem instances. On the other extreme are heuristic approaches,[33] which often explicitly employ some form of divide and conquer. These have no guarantee for optimality, but they tend to yield acceptable results within a reasonable time. Unfortunately, solution quality is typically degraded for larger problem instances.

Novel approaches have also been developed which are inspired by optimization processes that occur in nature. They include neural networks,[25,26,27] simulated annealing,[28,29] genetic algorithms,[30] and the elastic net method.[36] These algorithms employ implicit, adaptive forms of divide and conquer. At the beginning of the optimization process, the more global decisions are made. As the process proceeds, the global parts of the solution tend to become fixed, and only more local decisions are made. Effectively, the change from global to local optimization corresponds to an increase in the level of divide and conquer employed. In practice the change must be done relatively slowly to maintain solution quality.

When implemented serially, these algorithms represent a compromise, in terms of complexity and optimality, between enumerative and heuristic techniques. In fact they usually have some mechanism for controlling the compromise. However, they typically provide obvious opportunities for parallelism, though the parallel speedup is limited by communication costs among processors.[35] The possible speedup has been estimated at 3 order of magnitude beyond the serial limit.[4]

However, 6th-generation neural networks have been built which are asynchronous, lacking the clock-driven, lock-step communication of conventional parallel machines.[34] These have an additional 3 orders of magnitude speedup. Attempts at explaining this have lead to the realization that an important mathematical principle applies. The principle has been called the "Theorem of Orthogonal Division Errors".[4] It is based on an isomorphism between the lossless division of a TSP and the lossless division of a least-mean-square problem. The division of a TSP into 2 sub-problems is represented by

$$\min|\mathbf{V}|^2 = \min|\mathbf{A}|^2 + \min|\mathbf{B}|^2 + 2\,\min(\mathbf{A},\mathbf{B}) \qquad (1)$$

Here **V** is the ensemble "displacement vectors" for all possible TSP tours, and **A** and **B** are the displacement vectors for the 2 sub-problems. The theorem states that the division becomes lossless when the inner product (**A**,**B**) vanishes, so that the sum of the sub-solutions is the full solution. The inner product represents the cross-correlation between the 2 sub-problems. The theorem therefore describes an allocation of sub-problems that minimizes communication among processors. While in general it is non-trivial and often impossible to completely eliminate the cross-correlation, the goal is to at least minimize it.

An architecture for parallel divide and conquer is shown in Figure 1. A division algorithm splits the problem into sub-problems and allocate them among parallel processors. A recombination algorithm then combines the resulting sub-solutions. The parallel processors could execute optimization algorithms of any type. In fact, the type could be matched to the sub-problem at hand, based on its size and the desired solution quality.

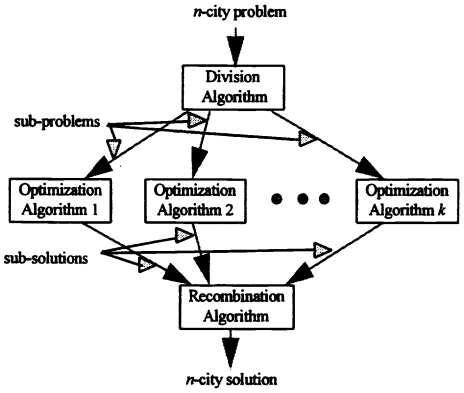


Figure 1: Architecture for parallel divide and conquer

However, divide and conquer has a potential for reducing problem complexity that could yield a far greater speedup beyond parallelism. This makes the approach attractive even when implemented serially. Consider the simple example of an n-city TSP divided into k equal parts. The number of possible tours in the full problem is $n!/n = (n-1)!$, disregarding symmetry in the inter-city distances. As we will later see, the divided sub-problems are optimizations of Hamiltonian paths versus Hamiltonian loops. Each sub-problem therefore has complexity $(n/k-2)!$. Since we are solving k sub-problems, complexity with divide and conquer is $k[(n/k-2)!]$. Considering actual numerical values gives a more concrete appreciation of the speedup. For example, a 100-city problem divided 10 times has complexity $10(8!) \approx 4\times10^5$, which is a tremendous improvement over the original $99! \approx 9\times10^{155}$. This reduction in problem complexity improves for larger n and k. Of course, additional computation is needed for the division and recombination operations, but in general this should be relatively insignificant.

Having established the desirability of a divide-and-conquer approach, we move on to multiple-resolution divide and conquer. The Theorem of Orthogonal Division Errors gives no particular prescription for divide and conquer, only the mathematical necessity of minimizing cross-correlation among sub-problems. Given this, we are led to the idea of clustering for performing the division, since intuitively, well-separated clusters have minimal cross-correlation. In fact, this appears to be what humans do when they try to solve TSPs by hand for clustered data. Since we are interested in fractal data, which has clusters at all scales, we expect to be able to perform the clustering recursively. Such a recursive or nested clustering is shown in Figure 2.

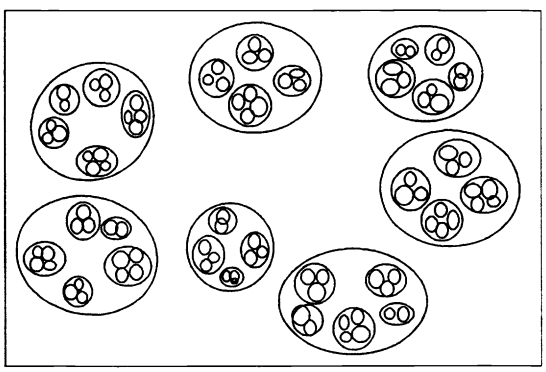


Figure 2: Recursive (nested) clustering of cities

The nested clustering process divides the TSP into sub-problems. We then need to find optimal solutions for the sub-problems, and find a way to combine them. As a human might do when solving a TSP, we find the optimal solution for the highest-level (largest) clusters first. We use some appropriate inter-cluster distance measure for this highest-level optimization,

such as distances between cluster centers. This is shown in Figure 3(a). We now know the way in which the highest-level clusters are linked for a near-optimal tour. We can then identify boundary clusters at the next lowest level to actually link the highest-level clusters. These could be, for example, the closest pairs of sub-clusters between 2 linked clusters. Figure 3(b) shows the identification of the boundary sub-clusters, which are labeled B. Now within each high-level cluster, we can find the optimal path through the lower-level clusters which begins and ends at the boundary sub-clusters. This is shown in Figure 3(c). We then form the union of the inter-cluster and intra-cluster links to get the near-optimal tour at the second level of clustering, as shown in Figure 3(d). We can continue this process recursively until we reach lowest-level clusters of manageable size. In this sense the higher-level optimal paths constrain the lower-level ones, through the boundary sub-clusters. This is desirable since the higher-level paths contribute more to overall solution optimality. This top-down recursion could also be considered a form of data compression, in the sense that if given a medium of limited bandwidth, the most important parts of the solution could be communicated first. Intuitively, the optimality of recursive divide and conquer would depend on the degree of inherent clustering in the data at the various levels.

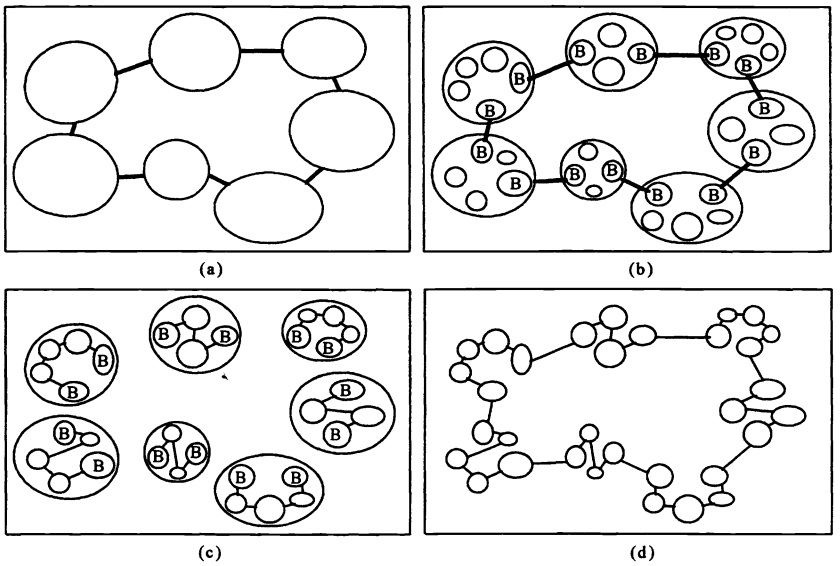


Figure 3: Optimization at highest-level of clusters (a), Identification of boundary clusters (b), Optimal intra-cluster paths for highest-level clusters (c), Optimization at second level of clusters (d)

Finally, we look at a more illustrative example of this recursive divide and conquer. It plainly shows how multiple levels of clustering imply multiple resolutions. In the example, a manager for the fictitious Burger World restaurant chain is given the task of visiting all the company's locations in the continental US. Attempts at constructing a minimum-distance itinerary cause him to lose sleep, until he finally realizes that the locations are fairly well clustered, so that he can treat groups of them separately. He need only determine starting and stopping locations for each group. He further deduces that he can do this starting with longer groups and working to smaller and smaller ones. In doing this he needs to work with various maps of increasing resolution. An example sequence of such maps is shown in Figure 4.

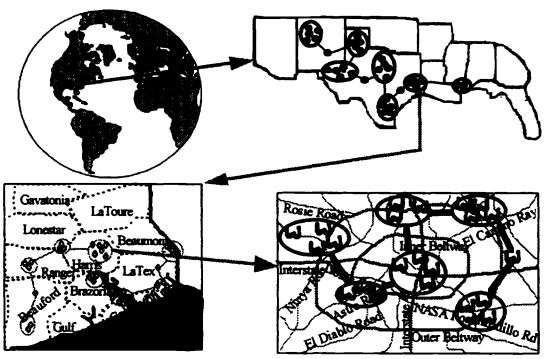


Figure 4: Illustrative example of divide and conquer

3. MULTIPLE-RESOLUTION CLUSTERING WITH WAVELETS

Recursive divide and conquer requires some form of nested clustering. As we have seen, this implies clustering at multiple resolutions. We therefore consider wavelets as the clustering tool, since they are explicitly based on multiple resolutions. We expect the lower-resolution representations of our data to yield higher-level (larger) clusters, and higher-resolution ones to yield lower-level (smaller) clusters. We will begin with one-dimensional examples and then move to examples on the plane. Of course, a one-dimensional TSP is trivial, but starting with a treatment in a single dimension allows us to more easily grasp the ideas.

The multiresolution analysis (MRA) of Mallat[7] and Meyer[8] provides representations of data at various resolutions. A one-dimensional MRA gives a Jth-level wavelet approximation of a continuous signal $f(t)$ as

$$f(t) \approx S_J(t) + D_J(t) + D_{J-1}(t) + \ldots + D_1(t) \tag{2}$$

The orthogonal signal components $S_J(t)$, $D_J(t)$, $D_{J-1}(t)$, ..., $D_1(t)$ are

$$\begin{aligned} S_J(t) &= \sum_k s_{J,k}\phi_{J,k}(t) \\ D_j(t) &= \sum_k d_{j,k}\psi_{j,k}(t) \end{aligned} \tag{3}$$

where k indexes the wavelet translations. Here the $s_{J,k}$, $d_{J,k}$, ..., $d_{1,k}$ are the wavelet transform coefficients, and the $\phi_{J,k}(t)$ and $\psi_{j,k}(t)$ are the approximating wavelet functions. The wavelet functions are generated from the father wavelet ϕ and mother wavelet ψ by

$$\begin{aligned} \phi_{j,k}(t) &= 2^{-j/2}\phi\left(\frac{t-2^j k}{2^j}\right) \\ \psi_{j,k}(t) &= 2^{-j/2}\psi\left(\frac{t-2^j k}{2^j}\right) \end{aligned} \tag{4}$$

The wavelet coefficients $s_{J,k}$ and $d_{J,k}$ are

$$\begin{aligned} s_{J,k} &\approx \int \phi_{J,k}(t) f(t) dt \\ d_{j,k} &\approx \int \psi_{j,k}(t) f(t) dt, \qquad j = 1,2,\ldots,J \end{aligned} \tag{5}$$

It is well known that wavelet transforms can be interpreted as banks of matched filters.[11] The self-similarity (scale invariance) of the filters is consistent with the idea of finding similar structure (clusters) in the data at all scales. These matched filter banks can be implemented in an analog fashion, for example electronically or optically. This has the advantage of insensitivity to shifts in the data that plagues digital implementations. Shift variance makes it more difficult to develop pattern-recognition algorithms in the wavelet transform domain. Unfortunately, the problem of computing the inverse wavelet transform with analog hardware is largely unsolved.

For digital implementation, the wavelet coefficients are computed with Mallat's fast pyramid algorithm.[7] This uses a low-pass analysis filter L, a high-pass analysis filter H, and a decimation-by-2 operation, as shown in Figure 5. The input to the pyramid algorithm is the discrete sampled signal

$$s_{0,i} = f_i, \; i = 1,2,\ldots n \tag{6}$$

We assume that the points to cluster are embedded in a real Euclidean space. The points thus need to be discretized as some form of digital vector (or matrix for higher dimensions). Our idea is to map the points to binary vectors or matrices, where a one indicates the presence of a point and a zero indicates the absense. We also need to insure that the vectors or matrices have enough elements to fully resolve the points.

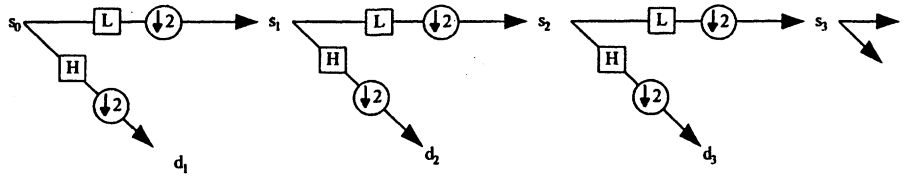


Figure 5: Pyramid algorithm for wavelet decomposition

In practice, we prefer to match the wavelet functions ϕ and ψ to the characteristics of the data. However, this is generally difficult to accomplish adaptively with a digital implementation. We will therefore experiment with different wavelet functions. One possibility is the classical Haar wavelet family,[9] shown in Figure 6(a). Another is the famous "D4" wavelet of Daubechies,[6] the first type of continuous orthogonal wavelet with compact support. The D4 wavelet family is shown in Figure 6(b). We may wish to have our multiple-resolution clusters "smoother" in some sense, in which cause we may turn, for example, to the Daubechies "D8" family, which is wider and smoother than the D4.[6] The D8 family is shown in Figure 6(c). We may also desire symmetry from our wavelet basis. We could then use, for example, the Daubechies "S8" family, shown in Figure 6(d), which was constructed to be as symmetric as possible.[6]

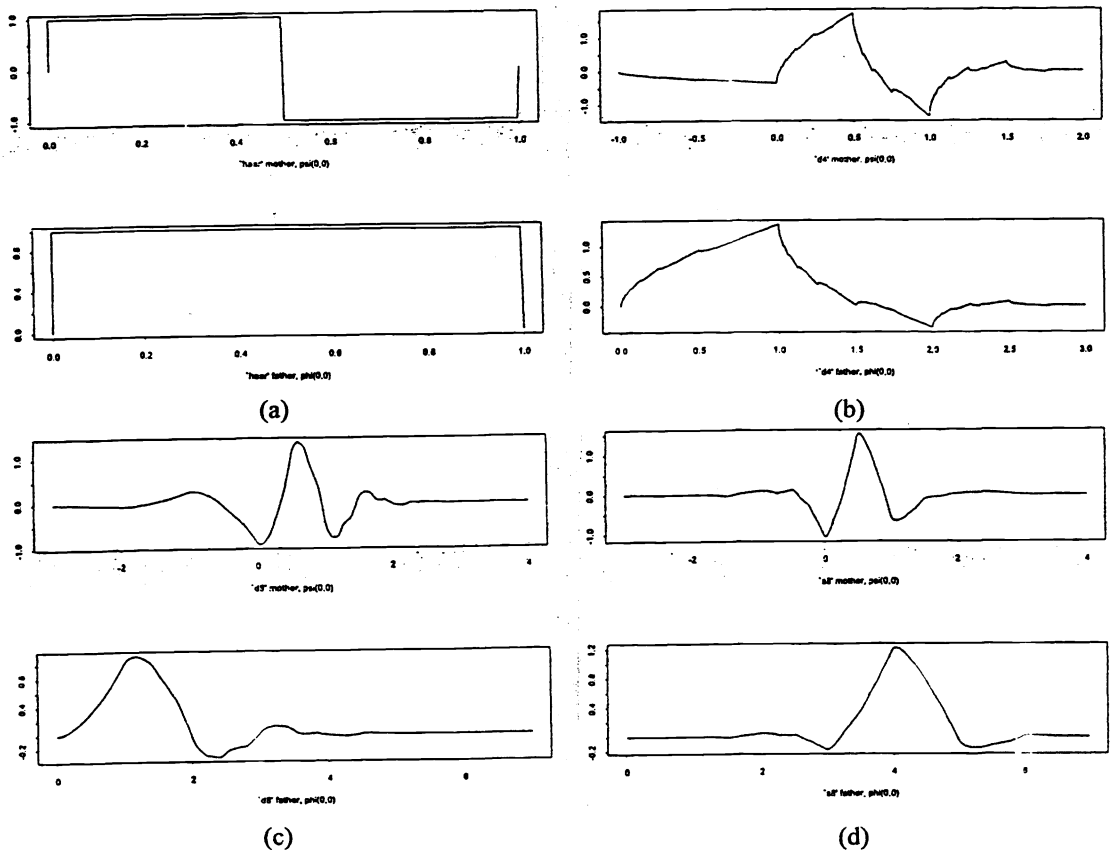


Figure 6: Haar (a), Daubechies D4 (b) , Daubechies D8 (c) and Daubechies S8 (d) wavelet families

Figure 7 shows the 2nd level of MRA using the Haar (b), D4 (c), D8 (d), and S8 (e) wavelet families. The input vector (a) is reminiscent of a Cantor set, and has 3 levels of clustering. The Haar basis may be more appropriate, since it has abrupt discontinuities like the data itself. It may also make the design of clustering algorithms easier, since its signals at various resolutions are less complex, though it may be better to work directly with wavelet coefficients versus signals, regardless of the wavelet basis. The D4 representation is visibly less smooth, and its pulses are less symmetric than for the D8 and S8. All bases other than the Haar seem to exhibit an overshoot or ringing effect similar to Gibbs' phenomenon.

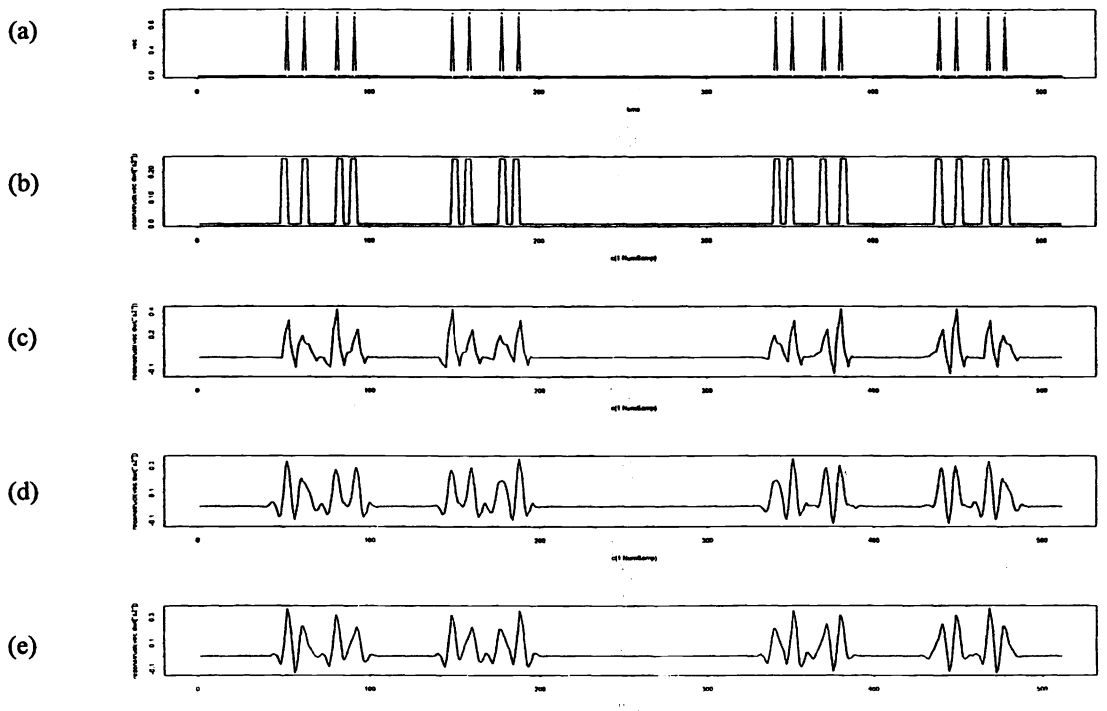


Figure 7: Input vector (a), and MRA for $J = 2$ for Haar (b), D4 (c), D8 (d), and S8 (e) bases

For the recursive divide-and-conquer approach, we wish to form nested clusters of various sizes, in the sense that larger clusters "contain" smaller ones. Wavelets seem to provide this, as we see in Figure 8. It is the first 4 levels of MRA for the Cantor-set example, using the S8 basis. The smoothing effect of the low pass filters generates pulses of increasing width, which eventually merge to become fewer in number. These merged pulses could be interpreted as clusters of the original points. The MRA provides clusters of various sizes, at the appropriate locations, as the recursive divide and conquer requires. The spatial coincidence of the pulses at various resolutions also provides information on how smaller clusters are contained in larger ones. Figure 9 shows this even more clearly. It is the 3rd, 5th, and 7th levels of the same MRA, with vertical rescaling at each level. Figure 10 shows MRA for a random data set which has more than one level of clustering. The multiple-resolution or nested clustering effect is also apparent for this data.

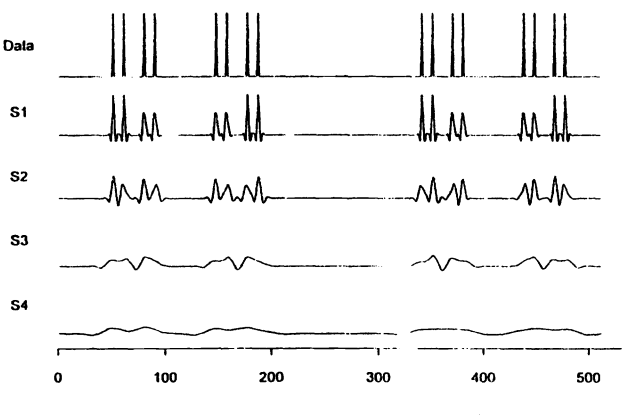


Figure 8: MRA for $J = 1,2,3,4$ for Cantor-set example

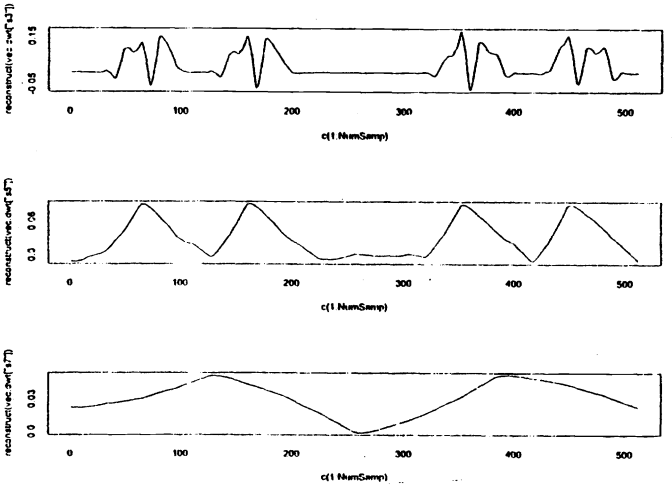


Figure 9: MRA for $J = 3,5,7$ with rescaling for Cantor-set example

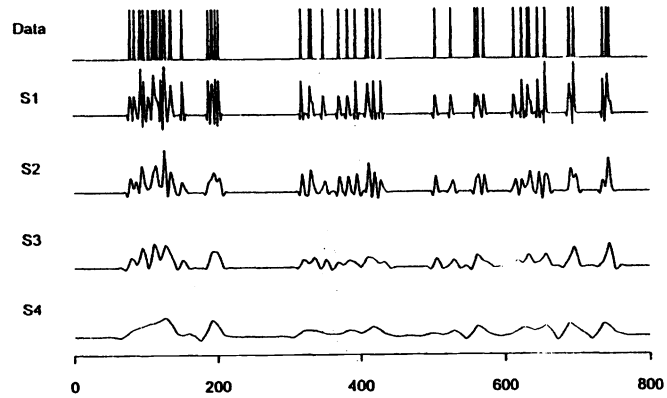


Figure 10: MRA for $J = 1,2,3,4$ for random clustered data

We now move to a 2-dimensional treatment. The most popular 2-dimensional extension of wavelet decomposition employs father and mother wavelets as

$$\begin{aligned} \phi(x,y) &= \phi(x)\phi(y) \\ \psi^H(x,y) &= \phi(x)\psi(y) \\ \psi^V(x,y) &= \psi(x)\phi(y) \\ \psi^D(x,y) &= \psi(x)\psi(y) \end{aligned} \tag{7}$$

The mother wavelets analyze structure preferentiality in the direction of their orientation. The wavelet $\Psi^H(x,y)$ is oriented horizontally, the wavelet $\Psi^V(x,y)$ is oriented vertically, and the wavelet $\Psi^D(x,y)$ is oriented diagonally.

The digital wavelet decomposition is implemented with a 2-dimensional pyramid algorithm, as shown in Figure 11. The input image is decomposed into a smoothed sub-image, and detailed sub-images with horizontal, vertical, and diagonal orientations. The smoothed sub-image is then further decomposed into smooth and detail sub-images at the next level. The process is then repeated recursively for J levels. The resulting sub-image is then a MRA approximation of the original signal at resolution J.

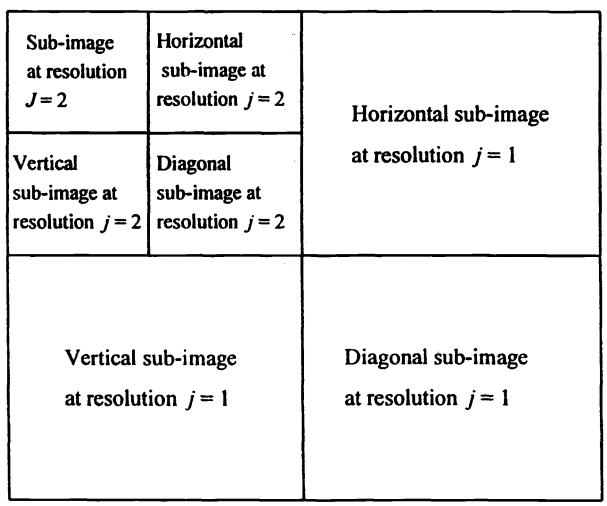


Figure 11: First 2 levels of 2-dimensional wavelet decomposition

Figure 12 shows the 2-dimensional MRA for $J = 2, 3, 4$ for a random, clustered data set. It clearly shows the same unsupervised clustering effect as for one dimension. Note that the larger, tighter clusters in the original image are represented as larger-amplitude pulses in the lower-resolution sub-images. This is reasonable, since these clusters contain more of the energy of the original image. These larger-amplitude pulses also exhibit more ringing, where their amplitude becomes negative. Note also an edge or boundary effect, in which there are low-resolution pulses in regions which are void of points, for example in the top left corner.

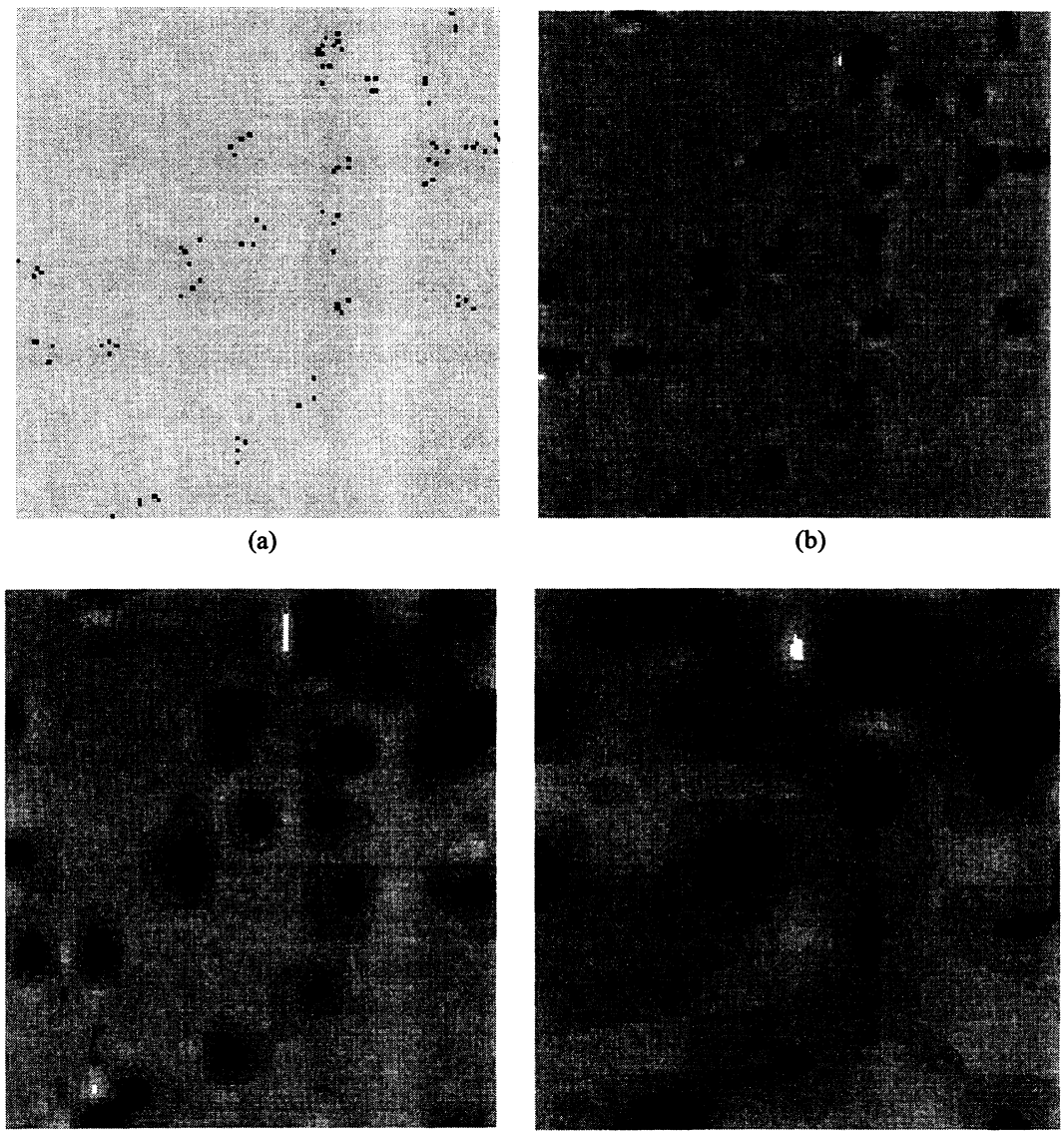
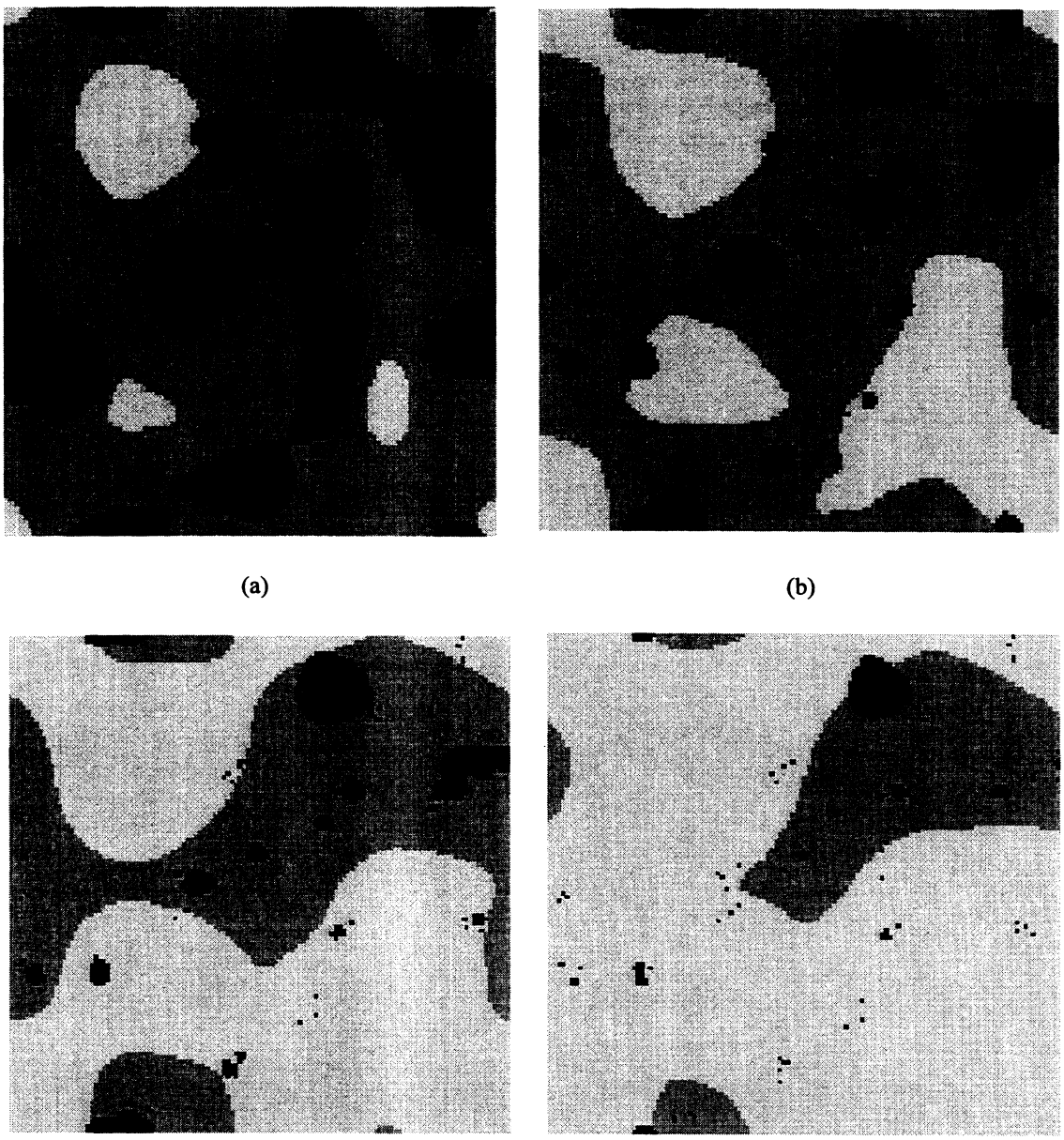


Figure 12: Input (a) and 2-dimensional MRA for resolutions $J = 2$ (b), 3 (c), and 4 (d)

As for clustering in general, recursive divide and conquer requires the formation of discrete clusters of points. However, mere MRA representations at various resolutions do not provide this, since they are continuous. A straightforward approach is to use threshold values to define boundary contours for clusters within MRA representations. More precisely, the intersection of a plane defined by a constant threshold value and the MRA representation at a given resolution defines the boundary contour.

However, as we can see in Figure 13, this straightforward threshold approach is problematic. The figure shows thresholded MRA representations at the first 5 resolutions for a random, clustered data set. At each resolution, it shows regions for which the amplitude is above the threshold. Also, regions are "stacked" in order of low to high resolution (in other words, if regions are in conflict, the higher-resolution ones are shown). The image greyscale is darker for higher resolutions. Parts (a) – (d) each have a constant threshold for all resolution levels. However the thresholds vary from 0.3, 0.4, 0.5, and 0.6 relative to the minimum and maximum amplitude at each resolution. While in general the level of threshold is arbitrary, we see that it strongly affects the clustering at the various resolutions. At lower threshold values, some seemingly unrelated points are clustered together, and at higher thresholds some points are ungrouped.



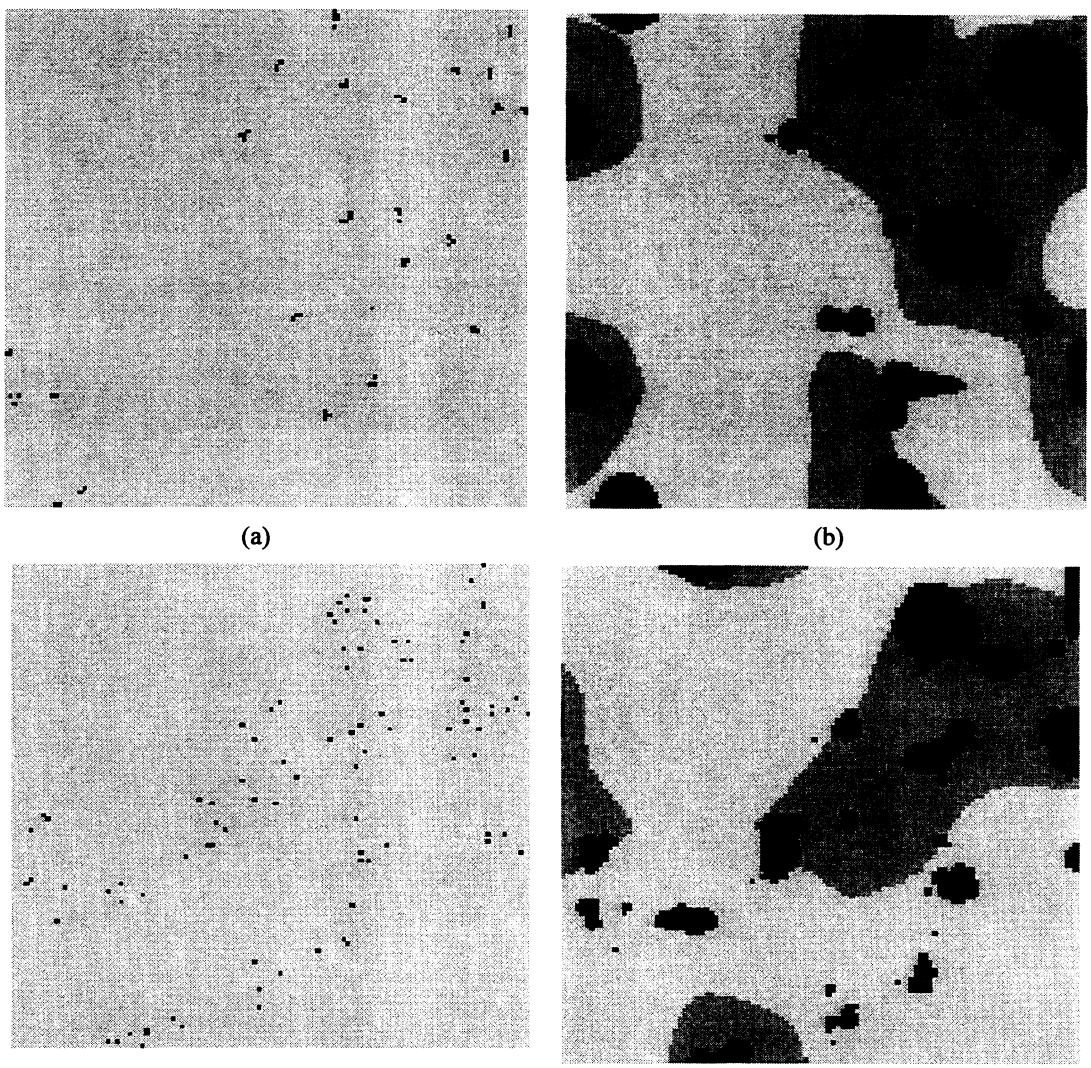
(a) (b)

(c) (d)

Figure 13:Thresholded MRA representations for relative thresholds 0.3 (a), 0.4 (b), 0.5 (c), and 0.6 (d)

After some experience in trying to set thresholds that yield meaningful cluster contours, we realize that in general a constant threshold is insufficient, even for a single resolution level. In particular, at a given level, different threshold values are needed at different locations. Thus we desire a more general threshold function. The question remains of how to create it. One possibility is to form an objective function which specifies the optimality of a given threshold function, according to some set of clustering criteria. An optimal threshold function could perhaps then be found with some global search algorithm like neural networks, simulated annealing, or genetic algorithms. One potential candidate is radial basis function neural networks.[25] Another approach may be to deal directly with the wavelet coefficients themselves, rather than the MRA representations, since the coefficients are already discrete.

Finally we look at the example in Figure 14, which shows how wavelet clustering is affected by the degree of inherent clustering in the data. The data set in part (a) has a much greater degree of clustering (has tighter clusters) than does the one in part (c). For both data sets, we use a relative threshold of 0.5 for each resolution level. The clusters for the tighter data set have more of a nesting (multiple-resolution) character. This seems consistent with the idea that the data set in part (c) is more uniform, so that clusters exist over fewer resolution ranges. We may eventually want to quantify the degree of clustering with (multi)fractal measures, to more precisely identify how it affects cluster formation.



(a) (b)

(c) (d)

Figure 14: Effect of inherent degree of data clustering on wavelet clusters

4. APPLICATION OF WAVELET CLUSTERING TO RECURSIVE DIVIDE AND CONQUER

Now that we have introduced the ideas of recursive divide and conquer and multiple-resolution clustering with wavelets, we apply them to specific examples. We first perform the multiple-resolution clustering of a set of points by the supervised setting of MRA thresholds. We then show a near-optimal TSP solution for another set of points using the divide-and-conquer approach.

Figure 15 shows a set of 100 points divided into clusters by wavelets. Note that the original 100 points were divided into 17 clusters, 10 from level $J=2$ and 7 from level $J=3$. Note also that the level-3 clusters were further divided into sub-clusters from level $J=2$. The clustering was done by the setting of appropriate cluster-boundary thresholds on the 2-dimensional MRA for the points. As already discussed, the setting of proper thresholds is problematic, because different values are needed for different regions at each resolution level. Here we set the threshold values by experimentation. In this sense the resulting multiple-resolution clustering was somewhat supervised, but at this point we are just trying to show the principle. In later work we will develop less supervised methods of forming discrete clusters.

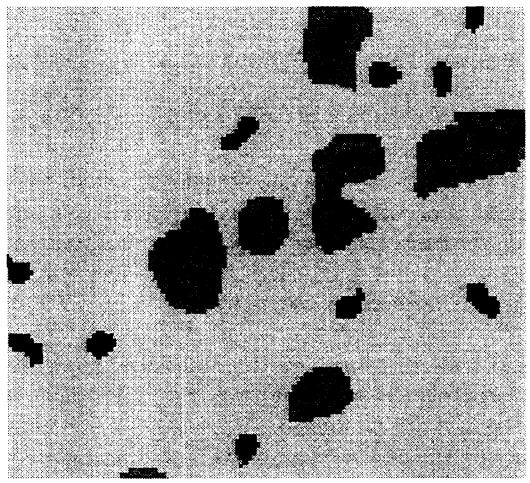


Figure 15: Multiple-resolution clustering for 100 points

We next do a computational implementation of a divide and conquer for a 100-point TSP. We first perform a single level of clustering with the k-means algorithm,[37] with the number of clusters k chosen to be 9. For the initial clustering required by the k-means algorithm we use hierarchial clusters[37] with average linkage. The data set with the resulting k-means clusters is shown in Figure 16(a). To find both the inter-cluster and intra-cluster optimal tours, we use a straightforward genetic algorithm[30] written in C language. It is the non-overlapping generational model[30] with population size 100. Parent selection is done with stochastic universal sampling,[32] a form of proportional selection. Each generation, the fitness of each member of the population is scaled linearly to lie between zero and one. The algorithm has permutation encoding,[30] with order crossover[31] and pair-swapping mutation, and operator probabilities of 0.4 and 0.002. For points within clusters, the algorithm was modified to find the optimal path (versus loop) which was constrained by the cluster boundary points. To find the boundary points, we use a simple program which compares distances between all pairs of points between each of the cluster pairs dictated by the optimal inter-cluster tour, and retains the smallest. The resulting divide-and-conquer solution is shown in Figure 16(b). Note that parts of the solution are sub-optimal (within clusters 3 and 6). This is caused entirely by the genetic algorithm, and is independent of the divide-and-conquer approach. In fact it reminds us of the desirability of the divide-and-conquer approach. If the same genetic algorithm was used for the full problem, without the divide and conquer, the convergence would be extremely slow, and the resulting solutions would be very poor.

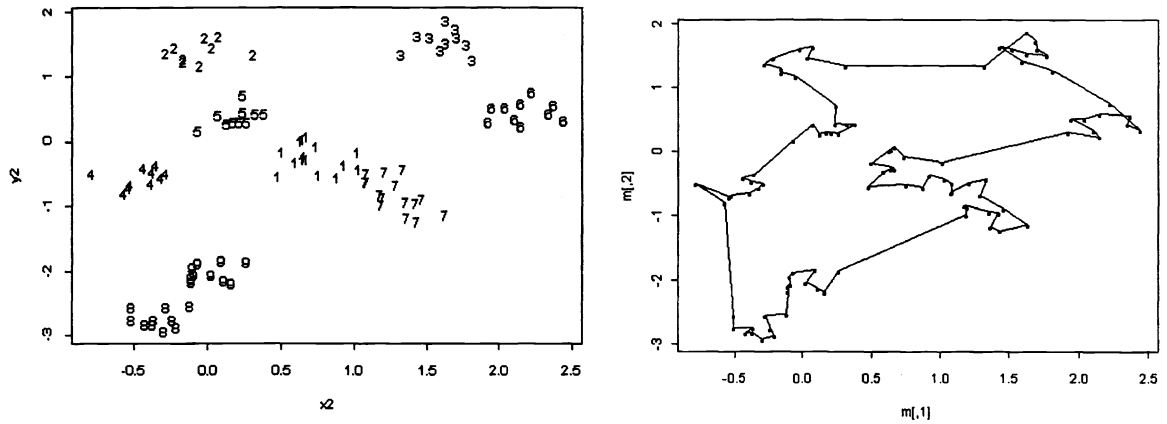


Figure 16: Problem data set with k-means clustering (a), and divide-and-conquer solution (b)

5. CONCLUSIONS AND REMARKS

We have introduced the strategy of recursive divide and conquer for combinatorial optimization, which is based on the mathematical principle of orthogonal division errors. We have seen how divide and conquer has the potential for dramatic speedup of large-scale problems, from both the reduction in problem complexity and the opportunity for parallelism. We explained that the divide-and conquer approach would benefit from the unsupervised formation of nested clusters at various resolutions. This led to the idea of forming the clusters with wavelets.

We then demonstrated, by way of multiresolution analysis, that wavelets can indeed provide a nested clustering effect. Lower-resolutin representations form larger clusters, higher-resolution representations form smaller ones, and spatial coincidence among the various resolutions provides the nesting information. We also showed how a simple thresholding approach could form the necessary discrete clusters from the continuous MRA representations, though this simple approach is problematic and needs improvement. Finally, we used wavelets to form nexted clusters for a specific example, and did a computational divide and conquer for another example.

In closing, we would like to stress the importance of effective methods like divide and conquer for intractable combinatorial optimization problems like the TSP. These are among the most computationally difficult known problems, and commonly arise in applications. Examples occur in such diverse areas as commercial product distribution, assignment of military targets to assets, postal scheduling, and the providing of humanitarian relief given limited resources.

ACKNOWLEDGMENTS

It is with the deepest gratitude that we acknowledge Margaret Cleven for her encouragement, support, and manuscript preparation and review. We are also grateful to Dr. Vijay Raghavan for his helpful input and discussions.

REFERENCES

1. C. Papadimitriou, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall
2. E. L. Lawler, et al (eds.), *The Traveling Salesman Problem*, John Wiley & Sons, 1985.
3. S. Noel, H. Szu, "Multiple-Resolution Divide and Conquer for Large-Scale Combinatorial Optimization," under review for *Int. Joint Conf. for Neural Networks IJCNN97*, 1997.
4. H. Szu, S. Foo, "Divide and Conquer Orthogonality Principle for Parallel Optimizations in the TSP," *Neurocomputing 7*, Elsevier, pp. 1-13, 1994.

5. G. Strang, T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge, 1996.
6. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, 1992.
7. S. Mallat, "A Theory For Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, Vol. 11, No. 7, pp. 674-693, 1989.
8. Y. Meyer, "Ondelettes, Fonctions Splines et Analyses Graduees," Lectures, Univ. of Torino, Italy, 1996.
9. A. Haar, "Zur Theorie Der Orthogonalen Funktionen - Systeme," *Math. Ann.*, Vol. 69, pp. 331-371, 1910.
10. Y. Meyer, *Wavelets: Algorithms & Applications*, SIAM, 1993.
11. H. Szu, J. Chen, Y. Sheng, "Wavelet Transforms as a Bank of Matched Filters," *Applied Optics*, July 1992.
12. T. Reed, H. Wechsler, "Segmentation of Textured Images and Gestalt Organization Using Spatial/Spatial-Frequency Representations," *IEEE Trans. On Patt. Anal. And Mach. Intel.*, Vol. 12, No. 1, pp. 1-12, 1990.
13. Z. Xie, M. Brady, "Wavelet Multiscale Representation and Morphological Filtering for Texture Segmentation," *IEE Colloq. on 'Morphological and Noninear Image Processing Techniques,'* pp. 2/1- 8, 1993
14. J. Rosiene, "Non-Separable Haar Wavelets and the Clustering of Magnetic Resonance Imagery," *Proc. 16th Ann. Int. Conf. of IEEE Eng. in Mad.and Biol. Soc.*, Vol. 2, pp.1210-1211,1994.
15. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, 1983.
16. M. Barnsley, *Fractals Everywhere*, Academic Press, 1988.
17. J. Feder, *Fractals*, Plenum, 1988.
18. G. Wornell, *Signal Processing with Fractals: A Wavelet-Based Approach*, Prentice Hall, 1996.
19. H. Peitgen, P. Richter, *The Beauty of Fractals*, Springer-Verlag, 1986.
20. R. Devaney, L. Keen (eds.), *Chaos and Fractals: The Mathematics Behind the Computer Graphics*, AMS, 1989.
21. W. Lam, G. Wornell, "Multiscale Representation and Estimation of Fractal Point Processes," IEEE Trans. On Signal Proc., Vol. 43, No. 11, pp. 2606-2617, 1995.
22. D. Johnson, A. Kumer, "Modeling and Analyzing Fractal Point Processes," *IEEE Int. Conf. On Acoust., Speech and Signal Proc.*, ICASSP-90, pp. 1353-1356, 1990.
23. B. Jones, et al., "Multi Fractal Description of the Large-Scale Structure of the Universe," *Astrophys. J.*, 332, pp. 1-5, 1988.
24. D. Snyder, *Random Point Processes*, Wiley, 1975.
25. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan, 1994.
26. J. Hopfield, D. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 55, pp. 141-152, 1985.
27. H. Szu, "Fast TSP Algorithm Based on Binary Neuron Output and Analog Neuron Input Using the Zero-Diagonal Interconnect Matrix and Necessary and Sufficient Constraints of the Permutation Matrix," *Proc. IEEE ICNN*, 1988.
28. S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimization by Simulated Annealing," *Science* 220, pp. 671-680, 1983.
29. H. Szu, "Fast Simulated Annealing," *Snowbird Utah Conf. Am. Inst. Phys.* (Denker ed.) 15, pp. 420-425, 1987.
30. D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
31. I. Oliver, D. Smith, J. Holland, "A Study of Permutation Crossover Operators on the Traveling Salesman Problem," *Proc. 2nd Int. Conf. on Genetic Algorithms*, Lawrence Erlbaum, pp. 224-230, 1987.
32. J. Baker, "Reducing Bias and Inefficiency in the Selection Algorithm," *Proc. 2nd Int. Conf. on Genetic Algorithms*, Lawrence Erlbaum, pp. 14-21, 1987.
33. S. Lin, B. Kernighan, "An Effective Heuristic Algorithm for the Travelling-Salesman Problem," *Operations Research*, Vol. 21, pp. 498-516, 1973.
34. H. Szu, "Sixth Generation Computing Architectures," *Learning and Recognition: A Modern Approach* (K. Zhao et al eds.), World Scientific, pp. 59-65, 1988.
35. H. Szu, C. Yeh, G. Rogers, M. Jenkins, A. Farsaie, "Speed Up Performances on MIMD Machines," *Int. Joint Conf. Neural Networks IJCNN-92*, Vol. III, pp. III-742 to III-747, Baltimore 1992.
36. R. Durbin, D. Willshaw, "An Analogue Approach to the TSP Using an Elastic Net Method," *Nature* 326 1987.
37. J. Hartigan, *Clustering Algorithms*, John Wiley & Sons, 1975.