The Min-max Cut Problem for Graph Partitioning: Application to Subsurface Geological Model Characterization

Feyi Olalotiti-Lawal & Yusuke Fujita

December 4, 2013

Abstract

We present a comprehensive study of approaches to solving the Min-max Cut Problem, otherwise known as the graph partitioning problem. We focus primarily on its application to heterogeneous porous media zoning problems which facilitates efficient paramter estimation and better understanding of the subsurface flow mechanisms. This follows from the idea of representing a discretized subsurface model as a graph with each grid representing a vertex and the similarity of properties among neighboring grids represented in an adjacency matrix. Our exploration of approaches to solving the problem begins from constuction algorithm where apply spectral clustering techniques. We propose a local search (LS) approach similar to standard onesand we also explored two of the common metaheuristics namely the Simulated Annealing (SA) and the Tabu Search (TS) techniques. Finally, we show a comparitive analysis of the performance of all these approaches.

Introduction

The min-max cut is a variant of the max cut problem which can be more computationally involving. For max cut problems, a graph is partitioned such that the sum of the edge weights along the cut is maximized. For the min-max problem however, a graph is partitioned into a number of clusters such that intra-cluster similarities are maximized while inter-cluster similarities are minimized.

Graph partitioning has a broad range of applications including efficient image processing and image segmentation [6],[9],[15] (which has been a critical component of medical and geophysical imaging), Very Large Scale Integration (VLSI) design and (social)network management [5], [3],[1] and data transformation [12], biological networks and epidemology [11], just to name a few.

Until recently, there has been almost no application of this technique to subsurface hydrological/petroleum reservoir flow systems, despite its visible potential in heterogeneity characterization as shown by [2]. The graph clustering approach can be applied to exploit the geological features of the fluid bearing rocks in the subsurface model calibration process. A traditional way of calibrating subsurface models is to a set of zonal multiplier values that minimizes the misfit between the observed data and simulation results. As can be noticed in Figure 1, case A takes advantage of the geologic features in the zoning and thus avoids a geologically unrealistic model as obtained in case B. Although responses from both models match observed data, model from case A will provide better understanding of the subsurface ans hence, an improved credibility of the subsurface performance prediction.

Coming back to the min-max cut problem, given a graph $G = (V, E)$ with adjacency matrix W. Defining $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$, the min-max problem involves deterimining k optimal

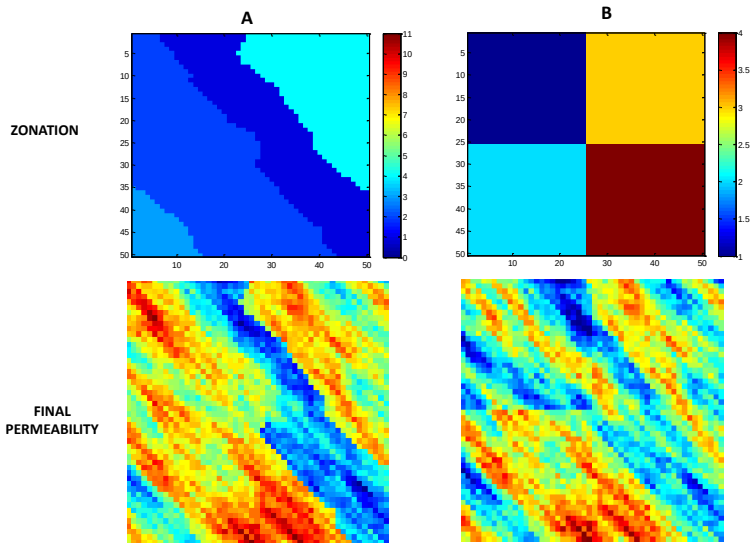


Figure 1: Benefit of graph clustering technique in subsurface model calibration: The model obtained from case A shows more geologically consistency and continuity than that from case B

partitions, $A_1, \ldots, A_k$ for $\bigcup_{i=1}^{k} A = V$ and $\bigcap_{i=1}^{k} A = \emptyset$ which minimizes the cut defined as [14]:

$$cut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i) \tag{1}$$

while at the same time maximizing the similarity of items within each partition i. [5] proposed the RatioCut to represent this similarity as the size of partition, $|A_i|$, while [6] introduced the Ncut which is measured by the weight of the edges, $vol(A_i)$ for each partition i $\forall i \in \{1, \ldots, k\}$. Here, we adopt the RatioCut approach so that the true objective function becomes:

$$Ratiocut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A}_i)}{|A_i|} \tag{2}$$

Here $\overline{A}$ is a complement of A, that is $\overline{A} = V \setminus A$. This optimization problem has been shown to be NP hard [16] and therefore cannot be solved to optimality in polynomial time but requires some heuristic and approximation approaches for reasonable optimal solutions. Majority of previous researchers have explored spectral clustering techniques which involves an eigenvalue decomposition of the Laplacian matrix and implementing Kmeans clustering algorithm to combine multiple resulting eigenvectors to obtain the clusters. This approach, compared to other approximate clustering methods is know to be computationally demanding [12]. Recently however, [7] proposed a distributed algorithm for decentralizing the computational load by which the eigenvector elements of each vertex for the first k eigenvectors can be computed by a separate processor. It was shown that each iteration requires $O(k^3)$ time and the number of iterations required to obtain a deviation of $\epsilon > 0$ from the exact eigenvectors is $O(log^2(n\epsilon^{-1})T)$ where T is the mixing time of the random walk on the graph, n is the graph size.

A number of authors have also introduced certain local search (LS) algorithms. The most popular of these is the famous systematic 2-OPT exchange porposed by [8]. The algorithm involves exchanging two vertices in different clusters seqentially and only once until all vertices have been

exchanged while constantly updating and retaining the best solution along the line. The entire process is repeated in a number of iterations until the best solution s reached. It was shown that each Lin-Kernighan LS algorithm requires $O(n^2)$ time. Later, [4] propsed a linear time algorithm for each LS iteration which was touted more efficient than the Lin-Kernighan algorithm.
Many authors have incorporated these LS techniques and their variants in several metaheuristic algorithms such as the Simulated Annealing, Genetic Algorithm, Ant Colony Optimization [2] and so on for the purpose of solving the graph clustering problem with varying method-specific complexity features. In this work, we attempt to solve the min-max cut problem in different instances represented by different synthetic and sections of benchmark geologic models of the subsurface. For each instance, we begin by implementing a construction heuristic using the spectral clustering approach. We tehn propose an hybrid LS algorithm on which the two metaheuristic algorithms we applied (SA and TS) are based. We compare the quality of the solutions as well as the computational time required for each of these approaches.

Construction Heuristic

Given any n size graph $G = (V, E)$, which in our application will be a spatial distribution of some subsurface flow property such as permeability, with some adjacency matrix A and a (diagonal)degree matrix D where $d_i = \sum_{j=1}^{n} a_{ij}$. We construct a symmetric and positive semi-definite matrix L knownm as the graph Laplacian given by $L = D - A$. It can be shown that the eigenvector of the L matrix provides the basis for the graph clustering [14].
For every eigenvector $h \in \Re^n$, by deinition we have:

$$\begin{aligned} h'Lh &= h'Dh - h'Ah = \sum_{i=1}^{n} d_i h_i^2 - \sum_{i,j=1}^{n} a_{ij} h_i h_j \\ h'Lh &= \frac{1}{2}\Big(\sum_{i=1}^{n} d_i h_i^2 - 2 \sum_{i,j=1}^{n} a_{ij} h_i h_j + \sum_{i=1}^{n} d_j h_j^2 \Big) = \frac{1}{2} \sum_{i,j=1}^{n} a_{ij} (h_i - h_j)^2 \end{aligned} \tag{3}$$

Approximating RatioCut for 2 clusters We consider the simple case realzation where $k = 2$. We seek to minimize $RatioCut(A, \overline{A})$ where $A \subset V$. We define the vector $h = (h_1, \ldots, h_n)' \in \Re^n$ with elements:

$$h_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\overline{A}|} & \text{if } v_i \in \overline{A} \end{cases}$$

Putting this in Equation 3, we show that the eigenvalue problem happens to be a reduction from the optimal RatioCut problem:

$$\begin{aligned} h'Lh &= \frac{1}{2} \sum_{i,j=1}^{n} a_{ij} (h_i - h_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \overline{A}} a_{ij} \Big(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}} \Big)^2 + \frac{1}{2} \sum_{i \in \overline{A}, j \in A} a_{ij} \Big(- \sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}} \Big)^2 \\ &= cut(A, \overline{A}) \Big(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2 \Big) \\ &= cut(A, \overline{A}) \Big(\frac{|A| + |\overline{A}|}{|A|} + \frac{|A| + |\overline{A}|}{|\overline{A}|} \Big) \\ &= |V|.RatioCut(A, \overline{A}) \end{aligned}$$

Also we can show that the vector h is orthogonal to the constant one vector $\mathbf{1}$:

$$h \cdot \mathbf{1} = \sum_{i=1}^{n} h_i = \sum_{i \in A} \sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i \in \overline{A}} \sqrt{\frac{|A|}{|\overline{A}|}} = |A|\sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}|\sqrt{\frac{|A|}{|\overline{A}|}} = 0 \tag{4}$$

Therefore the min-max problem becomes:

$$\min_{A \subset V} \quad h'Lh \quad \text{subject to} \quad h \cdot \mathbf{1} = 0 \tag{5}$$

This is a NP-hard, discrete optimization problem since the elements of h are only allowed to take two particular values. A relaxation in this setting is to ignore the discreteness and allow $h \in \Re^n$. By the Rayleigh-Ritz theorem [10], it can be seen that the solution of this problem is given by the vector h which is the eigenvector corresponding to the second smallest eigenvalue of L. The smallest eigenvalue of h is 0 with a corresponding eigenvector of $\mathbf{1}$. An approximate solution to the RatioCut is therefore obtained by suitable thresholding of the second eigenvetor of L. That is $v_i \in A$ if $h_i \geq 0$ and $v_i \in \overline{A}$ otherwise.

Approximating RatioCut for arbitrary k clusters This is similar to the case of 2 clusters described above. Here however, we partition V into k sets $A_1, \ldots, A_k$. We define k indicator vectors $h_j = (h_{1,j}, \ldots, h_{n,j})' \ \forall i \in 1, \ldots, n \ ; j \in 1, \ldots, k$ as:

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_j|} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

Similar to the 2 cluster case described above, it is easy to show that the RatioCut for a cluster can be expressed as:

$$h_i'Lh_i = \frac{cut(A_i, \overline{A}_i)}{|A_i|} = (H'LH)_{ii} \tag{6}$$

Where we define matrix $H \in \Re^{n \times k}$ with k columns orthonormal to each other, that is $H'H = I$. Plugging these together, we obtain the general expression for the RatioCut:

$$RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} h_i'Lh_i = \sum_{i=1}^{k} (H'LH)_{ii} = Tr(H'LH) \tag{7}$$

Therefore the general realxed min-max cut problem for graph partitioning becomes:

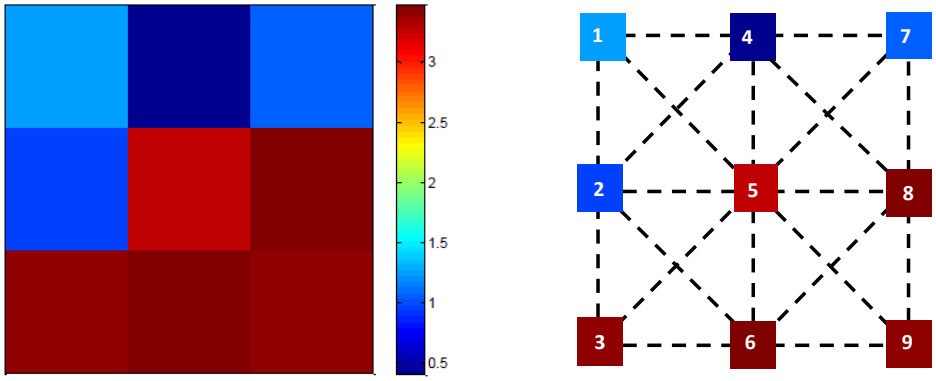
$$\min_{H \in \Re^{n \times k}} \quad Tr(H'LH) \quad \text{subject to} \quad H'H = I \tag{8}$$

This is a standard from of a trace minimization problem and also by the Rayleigh-Ritz theorm [10], we see that the solution is given by choosing H as a matrix which contains the first k eigenvectors of L as columns. Here for our application, as shown in the algorithm implemented in Algorithm 1, we consider the coordinates of $H(i, j)$ for $j = 1, \ldots, k$ as points in $\Re^k$ and cluster into clusters $A_1, \ldots, A_k$ using the kmeans algorithm [13].

Algorithm 1 A construction heuristic for min-max problem cut using spectral clustering

Input: Graph $G = (V, E)$, Adjacency matrix A, Number of clusters k
Output: Optimal k clusters

1: Compute Laplacian: $L = D - A$
2: Compute first k eigenvectors: $[h, s, h'] = eig(L)$
3: Compute first k eigenvectors: $h_i \quad \text{for } i = 1, \ldots, k$
4: From a matrix of eigenvector: $H \in \Re^{n \times k}$
5: Consider each row in H as a coordinate in $\Re^k$, form k clusters using the kmeans algorithm
6: Return optimal clusters



(a) (b)

Figure 2: (a)Log. of permeability distribution, (b) graph representation of the model

We applied this approach with a simple 3×3 grid reservoir model (9 vertices) with each cell assigned with a value of permeability. In estimating the adjacency matrix A we adopt similar approach for image segmentation and image processing [6] by which affinities are weighted based of their mutual similarities and the distance between them. Here, the pixel becomes the grid cells to which are assigned permeability values. Each element of the adjacency matrix is calculated as follows

$$a_{i,j} = \begin{cases} e^{\left(-\frac{\|p_i - p_j\|_2^2}{\sigma_P}\right)} e^{\left(-\frac{\|x_i - x_j\|_2^2}{\sigma_x}\right)} & \text{if } \|x_i - x_j\|_2^2 < r \\ 0 & \text{otherwise} \end{cases}$$

Where p_i and x_i are respectively the property and spatial location for some cell i, while r is a user defined correlation radius of the grid cell properties for the geologic model under consideration. The factors σ_P and σ_x are weights which can be used to magnify or attenuate the effect of the similarities in properties or closeness in the computed adjacency matrix.
We illustrate the principle with a 3×3 reservoir shown in Figure 2. The first figure showns the spatial distribution of the loagrithm of permeability while the second shows the graph representation of the system.

For this model, each grid cell has the dimensions $\Delta x = \Delta y = 50ft$. The Adjacency matrix A was constructed using the equation mentioned above. The corresponding Laplacian matrix L was then contructed using a r value of 100. As a rule of thumb [6] σ_P and σ_x should take values between 5-20 percent of the range of the 2-norm differences in the grid properties and distances respectively. For this case $\sigma_P = 1$ and $\sigma_x = 34ft$. The resulting symmetric L obtained is given below:

$$L = \begin{bmatrix} 0.1133 & -0.0748 & 0 & -0.0384 & -0.0001 & 0 & 0 & 0 & 0 \\ -0.0748 & 0.0800 & -0.0001 & -0.0049 & -0.0003 & 0 & 0 & 0 & 0 \\ 0 & -0.0001 & 0.0886 & 0 & -0.0066 & -0.0819 & 0 & 0 & 0 \\ -0.0384 & -0.0049 & 0 & 0.0976 & 0 & 0 & -0.054 & 30 & 0 \\ -0.0001 & -0.0003 & -0.0066 & 0 & 0.1725 & -0.0789 & 0 & -0.0800 & -0.0067 \\ 0 & 0 & -0.0819 & 0 & -0.0789 & 0.2488 & 0 & -0.0067 & -0.0813 \\ 0 & 0 & 0 & -0.0543 & 0 & 0 & 0.0546 & -0.0002 & 0 \\ 0 & 0 & 0 & 0 & -0.0800 & -0.0067 & -0.0002 & 0.1687 & -0.0818 \\ 0 & 0 & 0 & 0 & -0.0067 & -0.0813 & 0 & -0.0818 & 0.1698 \end{bmatrix} \quad (9)$$

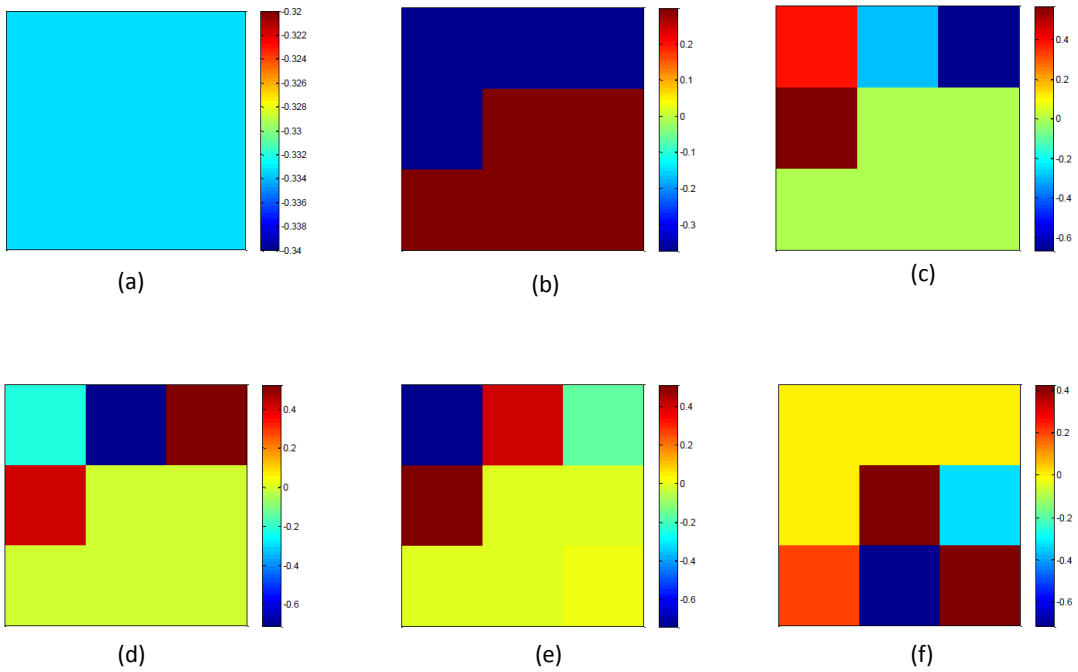


Figure 3: Eigenvectors of the Laplacian obtained from the 3 × 3 model showing (a) 9th eigenvector (b) 8th eigenvector (c) 7th eigenvector (d) 5th eigenvector (e) 3rd eigenvector (f) 1st eigenvector

It should be noted that the matrix become more populated as the property correlation radius r increases. Performing a singular value decomposition of the L matrix results in the set of eigenvectors shown in Figure 3 with the corresponding eigenvalues shown in Figure 4.
Consistent with our previous discussion on the spectral clustering algorithm, it can be noticed that the smallest (9th) eigenvalue vanishes while the corresponding (9th) eigenvector takes a constant value. For normalized formulations, the eigenvector corresponding to the smallest eigenvalue takes a constant value of unity. For a case where 2 clusters are required, as discussed above, the second smallest (8th) eigenvector presents a solution to the minimal RatioCut for the graph. The cluster

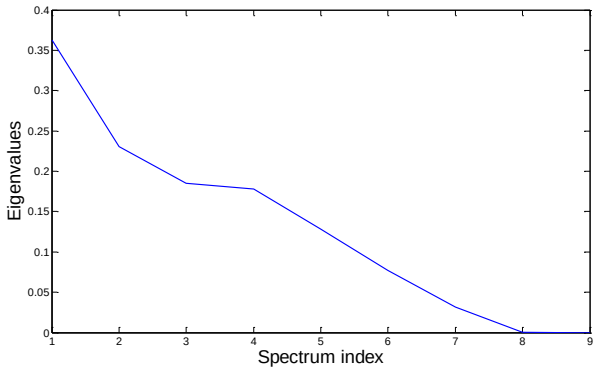


Figure 4: Eigenvalues of the Laplacian obtained from the 3 × 3 model

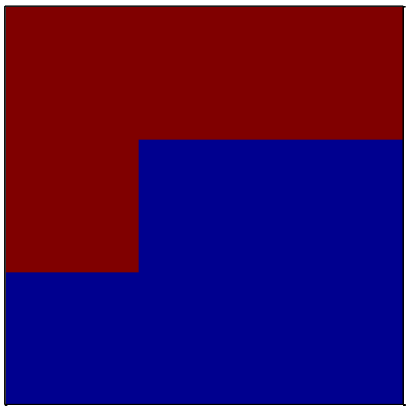


Figure 5: Two clusters obtained for the 3×3 model using spectral clustering

obtained is shown in Figure 5 and this is similar to that obtained using the kmeans algorithm. By inspection, it is easy to see that similar grids have been grouped into clusters in the optimal result. Also calculating the objective function using Equation 5, we obtain an optimal RatioCut value of 3.15×10^{-4}. This value will be compared with results from other heuristics in subsequent sections.

This same approach was implemented from 5 other instances of different permeability distribution and grid (graph) sizes as shown in Figure 6. The case A involving a 3×3 gridcells (9 vertices) has been discussed in detail above. Case B model, shown in Figure 6(a), is a 10×10 model (forming a 100 vertex graph). Cases C and D, shown respectively in Figures 6(b) and (c), have 20×20 gridcells (400 vertices) and 50×50 gridcells (2500 vertices) respectively. Cases E and F shown in Figures 6(d) and (e) respectively are sections cut out from the 40th and 50th layers of the benchmark SPE10 model. Cases E and F are both channelized models (existing in fluvial depositional environment) and they respectively have 50×50 gridcells (2500 vertices) and 60×100 gridcells (6000 vertices). These instances have been selected randomly such that each model has different features including size and property trends in the geology. This is to test the robustness of the algorithms considered in our work.

To implement the construction algorithm for the min-max cut problem on the cases A to F, we begin by construction the adjacency matrix, A for each model. Table 1 shows the parameters used in constructing the A matrix. As discussed earlier, the Laplacian matrix L, which serves as an input for eigen-decomposition, is constructed from the A matrix.

The construction algorithm was implemented for cases B to F (cases A already done as an earlier illustration). For each of these cases, 3 and 5 clusters were created as shown in Figures 7 and 8 respectively. Minimal RatioCuts obtained and the corresponding computational time requirements

Table 1: Parameters for Adjacency Matrix Construction

CASE	Model Size	r	σ_P	σ_x
A	3×3	50	0.0400	250
B	10×10	300	0.2172	9000
C	20×20	300	0.1940	9000
D	50×50	300	0.0264	9000
E	50×50	300	1.0147	9000
F	60×100	300	1.1796	9000

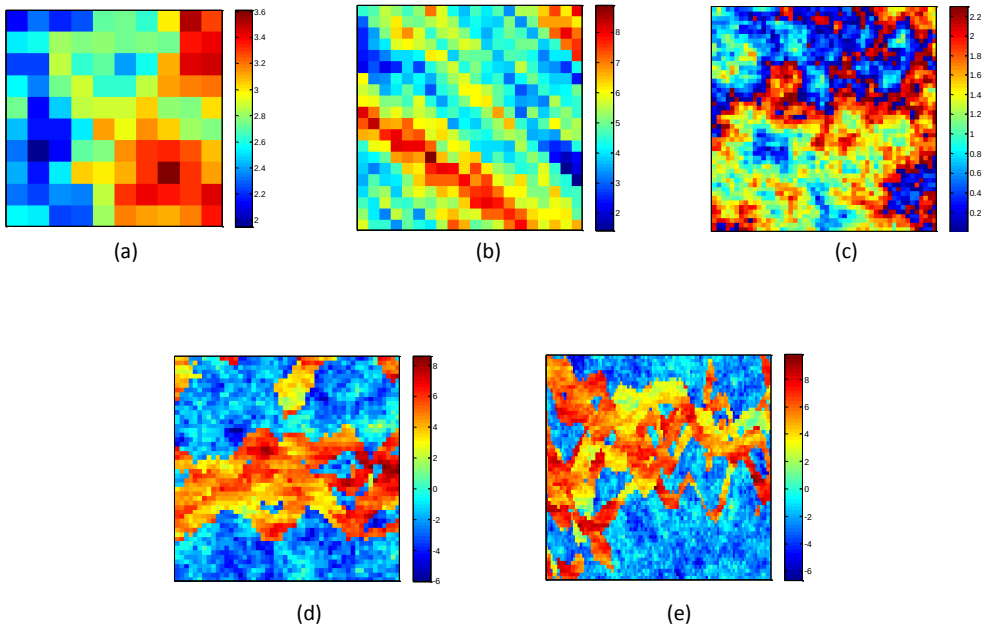


Figure 6: Permeability distribution of 5 other instances (a) Case B: 10 × 10 model, (b) Case C: 20 × 20 model (c) Case D: 50 × 50 model (d) Case E: Channelized 50 × 50 model, (e) Case F: Channelized 60 × 100 model

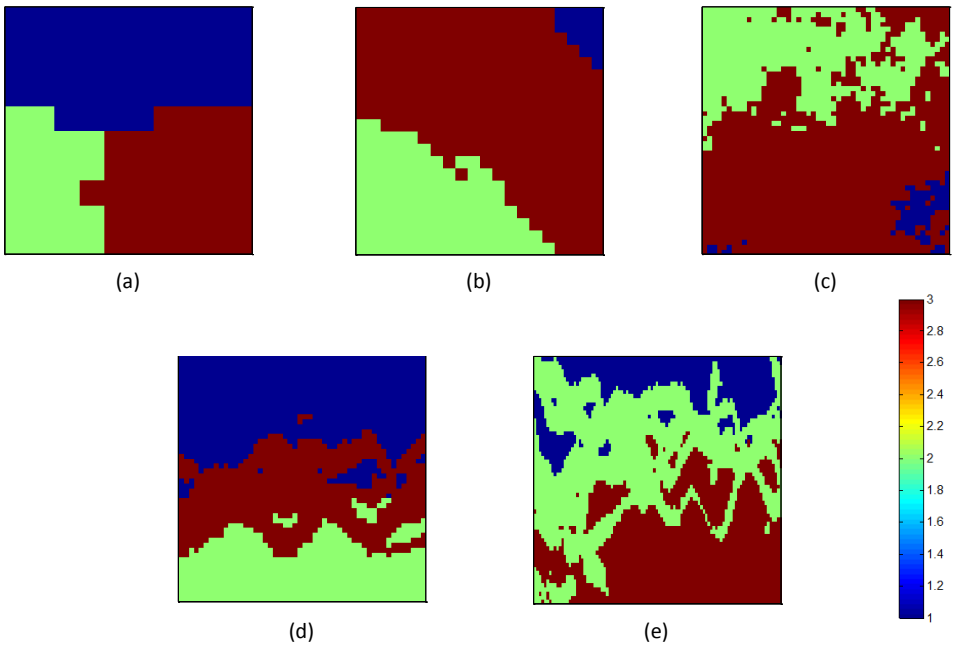


Figure 7: Geologic model zonation using the construction heuristics for 3 Clusters (a) Case B (b) Case C (c) Case D (d) Case E (e) Case F

were recorded for all cases. These will be compared with the results obtained from the local search and metaheuristics algorithms in subsequent section.

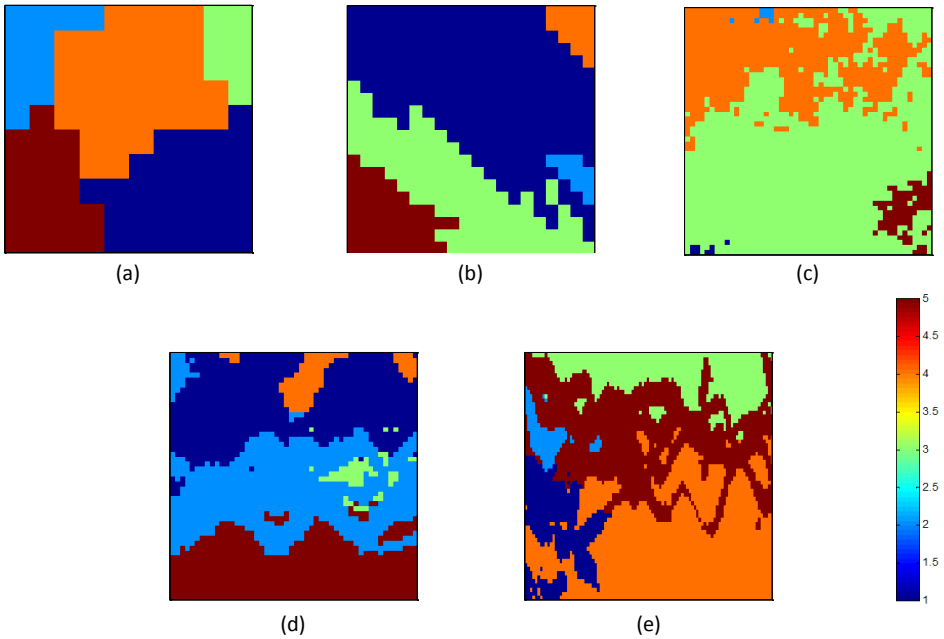


Figure 8: Geologic model zonation using the construction heuristics for 5 Clusters (a) Case B (b) Case C (c) Case D (d) Case E (e) Case F

Local Search (LS) Algorithm

As discussed earlier, the Lin-Kernighan (L-K) LS algorithm [8] applies reasonably well to common benchmark combinatorial optimazation probelms such as the TSP, vehicle routing problem and so on. A linear time LS algorithm was also proposed by [4] to improve on the L-K algorithm which runs in $O(n^2)$ time. These techniques are based on 2-OPT exchange ideas and thus will be quite suitable for instances of the minimum bisection problem, MAXCUT problem and even a 2 cluster min-max cut problem. For higher number of clusters in a min-max problem however, randomness is introduced into the search method. This produces an effect that significantly delay arrival at an optimal solution or, in worst cases, optimal solutions are never reached. Another issue with 2OPT exchange LS methods in dealing with min-max cut problem is that, unlike in other 2 cluster problem like the minimum bisection, some clusters have very small sizes compared with the sizes of others. As such, in limiting cases, there are very small search neighborhoods making the LS difficult.

To circumvent this problem, we define a new neighborhood structure for the graph clustering problem. In this structure, for k number of clusters, a sloution $[A_1, \ldots, A_k]$ is considered to belong to the neighborhood of the current solution $[A_{1_c}, \ldots, A_{k_c}]$ if for some $i \neq j$ and $i, j \in 1, \ldots, k$, $|A_i| = |A_{i_c}| - 1$ and $|A_j| = |A_{j_c}| + 1$. In other words, we remove a vertex at random from some cluster A_{i_c} and place in a different cluster A_{j_c}. We implemented the new algorithm as described in Algorithm 2.

Figures 9 and 10 graphically show the graph clustering results obtained for all cases A to F by implementing the LS algorithm for 3 and 5 clusters respectively. Case A was exempted from the 5 cluster problems due to its small size. Also case F was exempted due to the enormous computational resource reuirement to execute the LS algorithm. Exactly similar adjacency matrix used in the construction algorithm for each model were used here. Visaully, these results are appealing considering the fact that the initial property distributions are complete random for all cases but do not provide optimal cuts as smooth as those obtained from the construction heuristic. Solution qualities and computational requirements for each case were recorded for quantiative comparison and analyses in subsequent sections.

Algorithm 2 A Local Search heuristic for the min-max problem cut problem

Input: Graph $G = (V, E)$, Adjacency matrix A, Number of clusters k, $maxitr$
Output: Optimal k clusters

1: Initialize solution: Each grid randomly belongs to cluster j samples from uniform distribution $U(1, k)$
2: **for** itr=1 **to** $maxitr$ **do**
3: Move a randomly selected vertex and cluster to a randomly selected different cluster
4: Evaluate objective function (Equation 7). Accept move if objective function improves
5: If objective function does not improve over consecutive $0.01maxitr$ iterations, exit loop
6: **end for**
7: Return optimal clusters

Metaheuristics

We considered two metaheuristic technique for the graph clustering problem: Simulated Annealing (SA) and Tabu Search (TS). Both algorithms were implemented around the LS algorithm discussed above. The SA algorithm tries to avoid convergence to a local optima by probabilistically accepting moves which sometimes do not improve the objective function. The probability of accepting poor moves is reduced as the number of iteration increases by a process of 'annealing' whereby a temperature factor is varied such that acceptance probability is tightened. The SA scheme used for the min-max cut problem in this work is described in Algorithm 3. For all cases cosidered, an initial temperature value of 50 and coefficient value of 0.9 were applied.

Algorithm 3 A Simulated Annealing algorithm for the min-max problem cut problem

Input: Graph $G = (V, E)$, Adjacency matrix A, Number of clusters k, $maxitr$, T, $TFac$
Output: Optimal k clusters

1: Initialize solution: Each grid randomly belongs to cluster j samples from uniform distribution $U(1, k)$
2: Calculate initial objective function (Equation 7) f_{best}; set $T_{itr} \leftarrow T$
3: **for** itr=1 **to** $maxitr$ **do**
4: Move a randomly selected vertex and cluster to a randomly selected different cluster
5: Evaluate objective function (Equation 7): f_{itr}
6: $\alpha \leftarrow e^{-\left(\frac{f_{itr} - f_{best}}{T_{itr}}\right)}$
7: Update current solution with proposed move with probability α
8: Set $f_{best} \leftarrow f_{itr}$ with probability α
9: Set $T_{itr} \leftarrow T_{itr}TFac$ with probability α
10: If objective function does not improve over consecutive $0.01maxitr$ iterations, exit loop
11: **end for**
12: Return optimal clusters

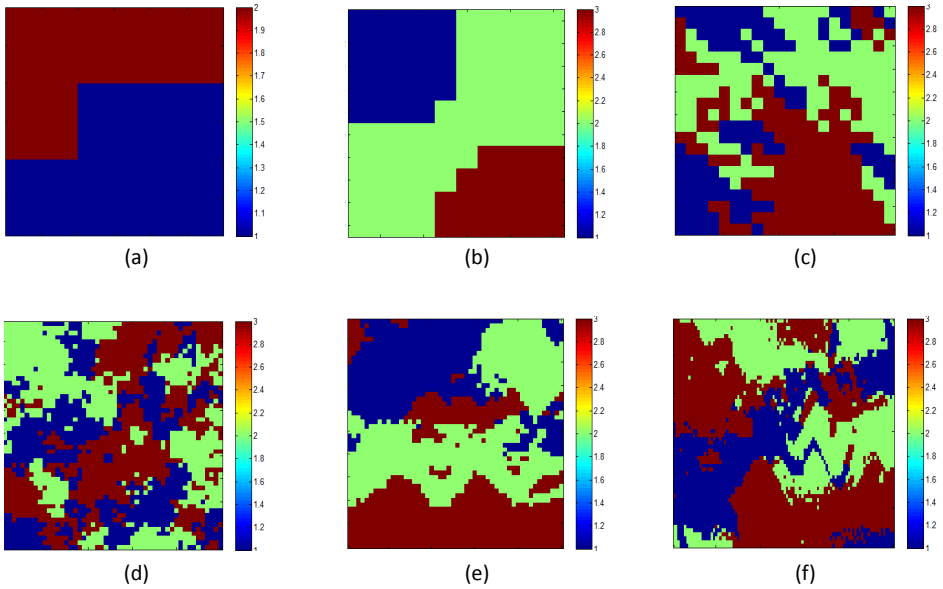


Figure 9: Geologic model zonation using the Local Search algorithm for 3 Clusters (a) Case A (b) Case B (c) Case C (d) Case D (e) Case E (f) Case F

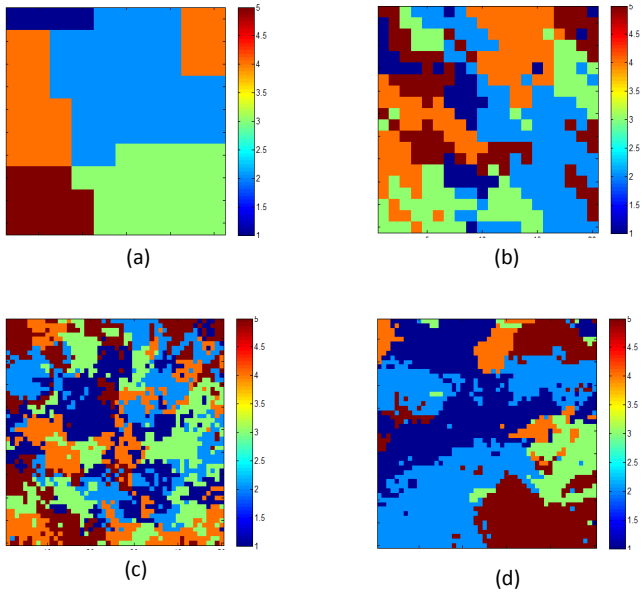


Figure 10: Geologic model zonation using the Local Search algorithm for 5 Clusters (a) Case B (b) Case C (c) Case D (d) Case E

On the other hand the Tabu Search (TS) algorithm attempts to avoid cycling between local optimal solutions by searching a reduced neighborhood $N(S,T) = N(S) \setminus T$, where S denotes the current solution and T denotes a set of solution that have been visited within the last $|T|$ iterations, otherwise known as the Tabu list. The larger the sie of T the restrictive the search is and vice versa. To improve the local search options, aspiration conditions are created. By the aspiration criterion, a move is accepted if ether it offers an objective function improvement better than the best found so far to if such move has never been proposed within the last $|A|$ iterations. In particular for the min-max cut problem with k with clusters intended, as shown in Algorithm 4, the Tabu list is represented by a $n \times k$ T_{mat} matrix where n is the graph size. Also an aspiration matrix A_{mat} of similar dimensions as T_{mat} is created. At every Tabu step, n local solutions are generated by moving (one at a time) each vertex to different cluster and calculating the resulting imporovement in objective function. Both the Tabu and Aspiration conditions are checked (in the T_{mat} and A_{mat} matrices) to decide on which local solution will be accepted. For all cases consdered, the tabu size $|T|$ was maintained at the graph size with a maximum number of iterations depends on the graph size: from 100 for case A to 70,000 for case F.
Results obtained using the metaheuristic algorithms are shwon in Figures 11 to 14. Here as well, we do not consider cases A and F for the 5 cluster problem for similar reasons mentioned earlier. Also by visual inspection, the SA algorithm gave reults slightly better results than both the TS nad LS results. This is due to the feature of SA that drives solutions away from local optima.

Algorithm 4 A Tabu Search algorithm for the min-max problem cut problem

Input: Graph $G = (V, E)$, Adjacency matrix A, Number of clusters k, $maxitr$, Tabu size $|T|$, Aspiration size $|A|$

Output: Optimal k clusters

1: Initialize solution: Each grid randomly belongs to cluster j samples from uniform distribution $U(1, k)$
2: Calculate initial objective function (Equation 7) f_{best}; $\Delta f_{best} \leftarrow 0$
3: Initialize $n \times k$ matrices T_{mat} and A_{mat} to be all zeros
4: **for** itr=1 **to** $maxitr$ **do**
5: **for** i=1 **to** n **do**
6: Select vertex $v_i \in C_j$ at random. Remove v_i from cluster C_j and place in randomly selected cluster C_l such that $C_l \cap C_j = \emptyset$
7: Evaluate objective function improvement Δf_{test}
8: **end for**
9: Select best move among all n local searches such that $\big(\Delta f_{test} \leq \Delta f_{best} \& \big(T_{mat}(i,l) + |T| < itr$ or $A_{mat}(i,l) + |A| < itr\big)$
10: Update A_{mat} and T_{mat}
11: Update current solution; Δf_{best}
12: If objective function does not improve over consecutive $0.01maxitr$ iterations, exit loop
13: **end for**
14: Return optimal clusters

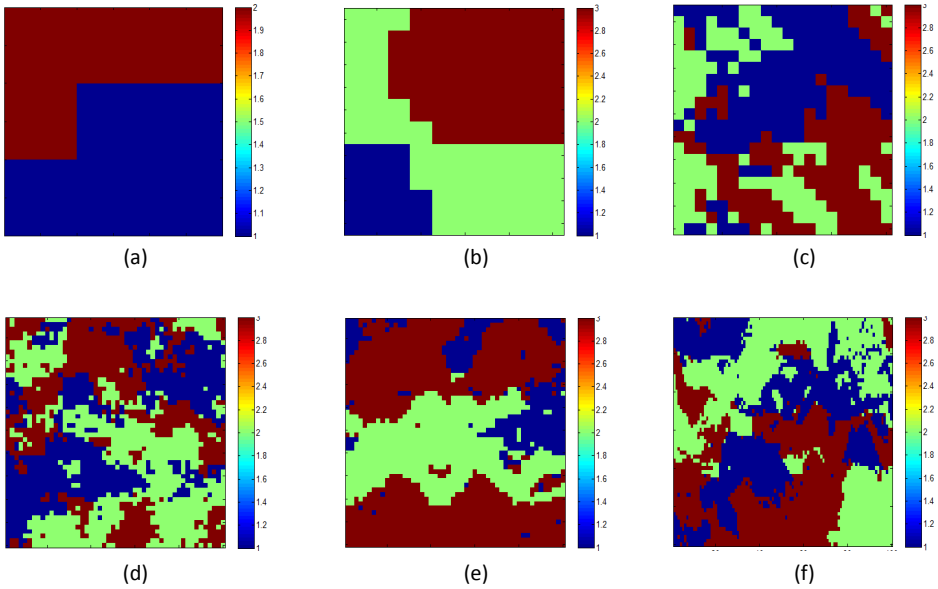


Figure 11: Geologic model zonation using the Simulated Annealing algorithm for 3 Clusters (a) Case A (b) Case B (c) Case C (d) Case D (e) Case E (f) Case F

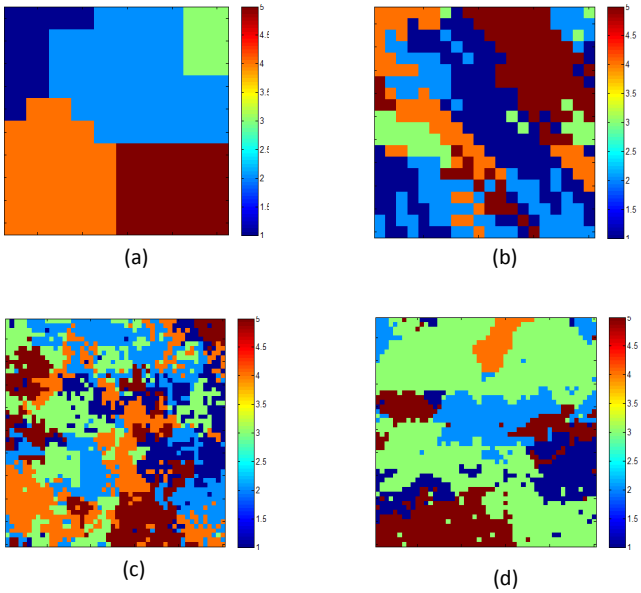


Figure 12: Geologic model zonation using the Simulated Annealing algorithm for 5 Clusters (a) Case B (b) Case C (c) Case D (d) Case E

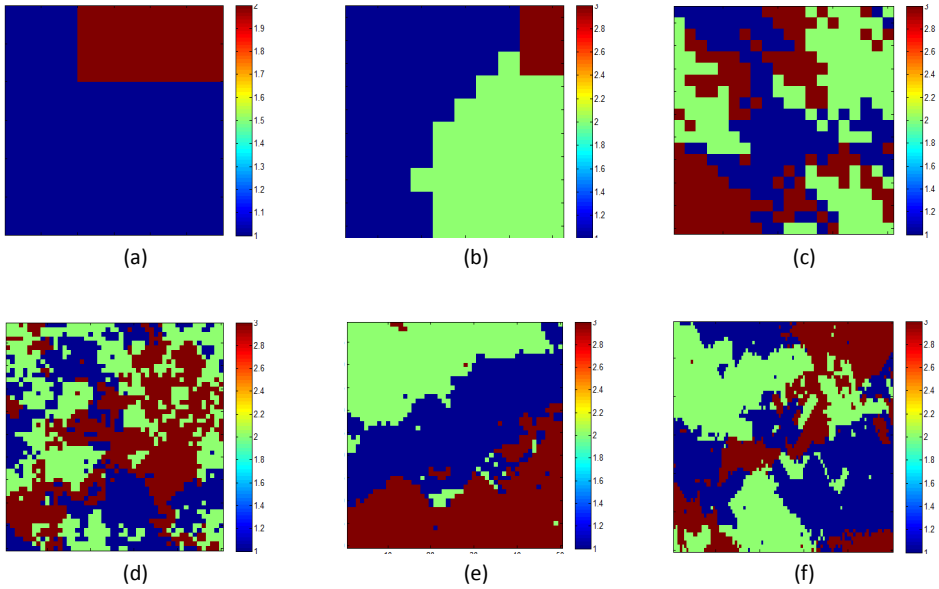


Figure 13: Geologic model zonation using the Tabu Search algorithm for 3 Clusters (a) Case A (b) Case B (c) Case C (d) Case D (e) Case E (f) Case F

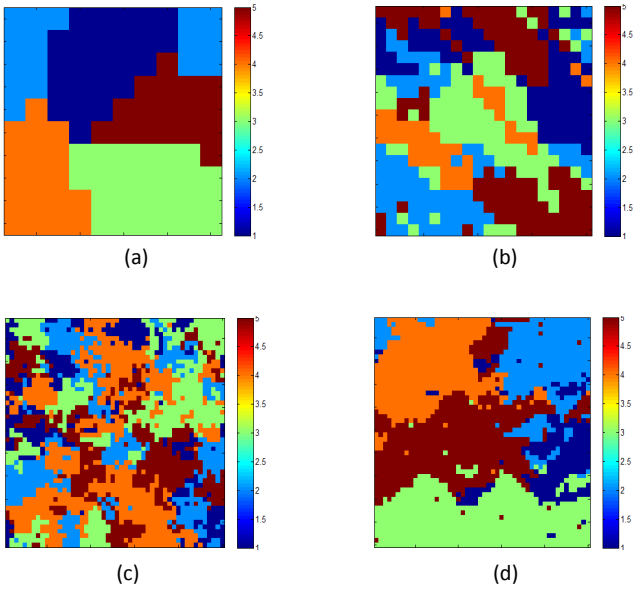


Figure 14: Geologic model zonation using the Tabu Search algorithm for 5 Clusters (a) Case B (b) Case C (c) Case D (d) Case E

Table 2: Final Objective Function Values from all 4 Algorithms Considered

CASE	No. of iter.	No. of Clusters	Construction	LS	SA	TS
A	100	2	3.15×10^{-4}	3.15×10^{-4}	3.15×10^{-4}	4.95×10^{-4}
B	20,000	3	2.5553	3.5723	3.0009	2.6256
		5	5.9673	6.8772	6.0925	6.4923
C	20,000	3	0.0700	0.8210	0.7521	0.8892
		5	0.1999	1.8283	1.1843	1.5645
D	40,000	3	0.1175	3.3214	2.9134	3.2596
		5	0.4763	7.9862	7.4634	7.9913
E	50,000	3	0.1626	7.4423	4.7892	6.7605
		5	4.0341	26.2735	25.2456	13.0032
F	60,000	3	0.5967	15.5734	18.2150	14.1943
		5	2.1383	–	–	–

Computational Results

Here we quantitatively analyse and compare all solutions obtained using construction heuristic (spectral clustering), local search and metaheuristics (SA and TS) in terms of solution quality and CPU time. Comparisons are presented for 2 (for case A), 3 (cases B to F) and 5 (cases B to E respectively) clusters.

Final RatioCut values obtained as optimal solutions of the min-max cut problem for all cases are shown in Table 2. It is important to point out here that the number of iterations in the second column do not apply to the construction algorithm. Clearly, the construction heuristic (spectral clusterng algorithm) provides the best solutions to the optimization problem for all cases and number of clusters considered. The SA algorithm improves on the LS algorithm by avoiding convergence to local optima solutions, hence it is not suprising that the SA algorithm provides slightly better solutions to the optimization problem for all cases. The TS algorithm on the other hand consistently gave the poorest solutions. This may be eplained from the somewhat greedy nature of the algorithm whereby the best of the local non-tabu solutions or that satisfy certain aspiration conditions are retained. This create some randomness in the final cut where clusters visually appear noisy and as such, poor solutions result.

Looking at the computational time requirement for all algorithms, as shown in Table 3, the construction algorithm provides the optimal solution in the shortest period of time for all cases considered. All calculations were carried out on a 3.70GHz 64bit machine running on a 24MB memory. Since the SA algorithm builds on (randomizes) the LS algorithm, it is expected that the LS algorithm will be slightly faster than the SA algorithm. Also, due to the memory requirements and constant checking and updating of the tabu list in the TS algorithm, it is not surprising that this takes a little longer than the rest of the algorithms. One other observation here is the approximate exponential increase in computational expense as the graph (geologic model) size increases as shown in Figure 15 for all algorithms.

Summary and Conclusions

Coming back to subsurface model characterization which has been a focus for primary application of the graph clustering approaches discussed so far. What is (are) the physical significance of these results? As mentioned much ealier in this write-up, a subsurface/petroleum reservoir engineer always seeks efficient computational ways of understanding the flow performace in the subsurface by learning from available data. This is known as subsurface model calibration by which subsurface properties such as heterogeneous permeabilities are modified to match observed

Table 3: CPU Time recorded (in seconds) for all 4 Algorithms Considered

CASE	No. of iter.	No. of Clusters	Construction	LS	SA	TS
A	100	2	0.30	4.14	5.00	5.62
B	20,000	3	0.30	4.22	5.97	9.52
		5	0.30	5.50	7.35	11.09
C	20,000	3	0.36	5.46	6.01	10.10
		5	0.40	6.47	7.56	13.10
D	40,000	3	1.75	164	168.0	192
		5	1.79	244	228	243
E	50,000	3	3.34	310	329	352
		5	3.35	335	322	319
F	60,000	3	15.82	3141	2959	2970
		5	16.61	–	–	–

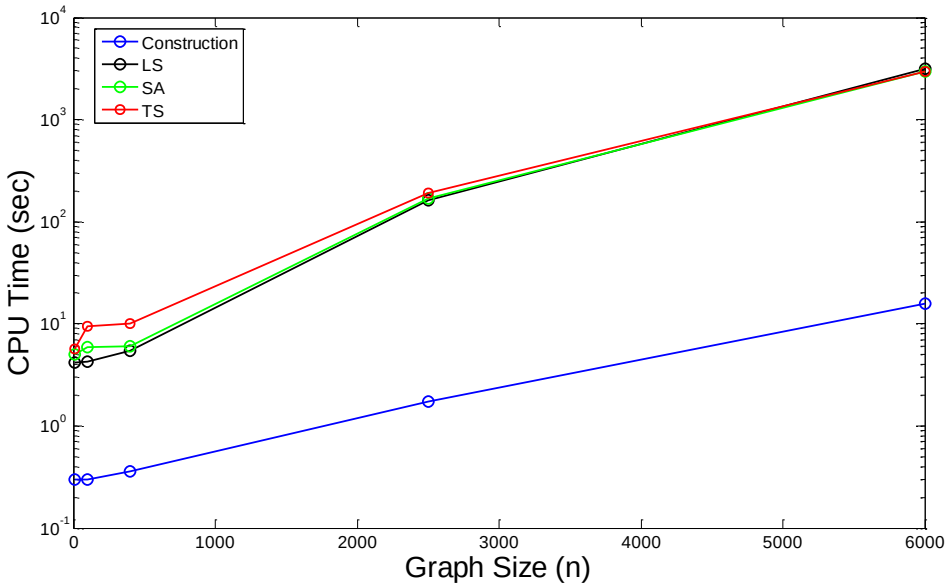


Figure 15: CPU Time (seconds) as function of graph size for all algorithms

production data while keeping the final model geologically realistic. A common efficient way is through partitioning the subsurface into zones or flow units and multiplying properties in each zones by certain multipliers. The graph clustering approach offers a great potential in this situation where a calibrated model retains important geologic features so that it does not lose its predictive capability.

Unlike in most other network problems such as the VLSI or social network management, solving the min-max cut of graph clustering problem for geologic models for the purpose of zonation can be quite computationally demanding. Most real life geologic models usually contain millions of grid cells (vertices) with very large radii of correlations, meaning that the Laplacian matrix obtained from such graphs are densely populated; increasing the diffiulty in obtaining optimal solutions within reasonable time using any of the algorithms described earlier. However, based on the results obtained, the following conclusions were reached:

- The Contruction heuristic approach applied (spectral clustering) provides the best optimal solution to the min-max cut problem in the shortest time compared to the other algorithm considered.
- LS and thus SA algorithms are better metaheuristics than the TS algorithm for possible reasons discussed earlier.
- The LS and SA still have some advantages over the construction approach in that they can be easily parallelized for better efficiency, and with better LS schemes, might result in better solutions.

References

[1] Weifu Chen and GuocanFeng. Spectral Clustering: A Semi-supervised Approach. . 2010. Neurocomputing 77(2012)229242.

[2] Akhil Datta-Gupta Eric Bhark and Behnam Jafarpour. History Matching with a Multiscale Parameterization Based on Grid Connectivity and Adaptive to Prior Information. . 201. SPE-147372 Society of Petroleum Engineering.

[3] Chris H.Q. Ding et al. A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. . 2001. Pro . IEEE Int'l Conf. Data Mining, 2001.

[4] C.M. Fiduccia and R.M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. . 1982. IEEE 19th Design Automation Conference DOI: 0420-0098/82/0000/0175 00.75.

[5] Lars Hagen and Andrew B. Kahng. New Spectral Methods for Ratio Cut Partitioning and Clustering. . 1992. IEEE Transactions On Computer-aided Design, Vol. 11, No. 9, September 1992.

[6] Thomas Leung Jitendra Malik, Serge Belongie and Jianbo Shi. Contour and Texture Analysis for Image Segmentation. . 2001. International Journal of Computer Vision 43(1), 727, 2001.

[7] David Kempe and Frank McSherry. A Decentralized Algorithm for Spectral Analysis. . 2004. STOC04 June 1315, 2004, Chicago, Illinois, USA.

[8] B.W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. . 1969. The Bell System Technical Journal, Feburary 1970.

[9] Chi Man Liu and Shuchi Chawla. Approximations Algorithms. . 2007. Lecture Notes for CS880: Approximations Algorithms.

[10] Helmut Ltkepohl and H. Lutkepohl. Handbook of Matrices. . 1997. Wiley, February 18, 1997.

[11] M.E.J. Newman. Properties of highly clustered networks. . 2003. Phys. Rev. E 68, 026121 (2003).

[12] Satu Elisa Schaeffer. Graph clustering. . 2007. Computer Science Review 1 (2007) 2764.

[13] Shaogang Gong Tao Xiang. Spectral clustering with eigenvector selection. . 2007. Pattern Recognition 41 (2008) 1012 1029.

[14] Ulrike von Luxburg. A Tutorial on Spectral Clustering. . 2007. Max Planck Institute for Biological Cybernetics Technical Report No. TR-149 Updated version, March 2007.

[15] Song Wang and Jeffrey Mark Siskind. Image Segmentation with Ratio Cut. . 2003. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.25, No.6, June 2003.

[16] Dorothea Wangner and Frank Wagner. Between Min Cut and Graph Bisection. .

Appendix: MATLAB Codes

```
%% CODE IMPLEMENTING GRAPH PARTITIONING ALGORITHMS INCLUDING
%% (1) Construction Heuristic: Spectral Clustering
%% (2) A Local Search Algorithm
%% (3) Metaheuristic Algorithms: Simulated Annealing & Tabu Search

%% AUTHOR: Yusuke Fujita & Feyi Olalotiti-Lawal

clear all; close all; clc; tic

load('adj20x20')


%% CONSTRUCTION HEURISTIC
tic
diagVec = sum(A,2);
D(1:length(D)+1:numel(D)) = diagVec;
L = D-A;

%% Eigen-Decomposition and Spectral Clustering
k =2;
[u s v] = eigs(sparse(L),k,'sm');
% [u s v] = svd(L);
U = fliplr(u); [ID] = kmeans(U(:,1:k),k);
figure(2);imagesc(reshape(ID,nx,ny))

toc
%% GET OBJECTIVE FUNCTION
func = @(x,L) trace(x'*L*x);
for i = 1:k;
    h = zeros(NV,1);
    h(ID==i) = 1;
    H1(:,i) = h;
    H(:,i) = h/sqrt(sum(h));
end
OF = feval(func,H,L);


%% LOCAL SEARCH
[ID] = randi(k,nx*ny,1);
figure(3);imagesc(reshape(ID,nx,ny))
H1 = zeros(NV,k); H = zeros(NV,k);
for i = 1:k;
    h = zeros(NV,1);
    h(ID==i) = 1;
    H1(:,i) = h;
    H(:,i) = h/sqrt(sum(h));
end
%
%Initial Objective function
```

```
OF = feval(func,H,L);
OF_vec = OF;

% Number of vertices
nx = 20; ny = 20; nz = 1;

% Number of clusters
k = 3;

% Graph Size
NV = nx*ny*nz;
% Set objective
func = @(x,L) trace(x'*L*x);
% Search algorithm
%    1. LS(Local Search)
%    2. SA(Simulated Annealing)
%    3. TS(Tabu Search)
Algorithm = 'TS';

% Number of iteration
nitr = 20000;


% Generate initial solution
[ID] = randi(k,nx*ny,1);
figure(3);imagesc(reshape(ID,nx,ny))
H1 = zeros(NV,k); H = zeros(NV,k);
for i = 1:k;
    h        = zeros(NV,1);
    h(ID==i) = 1;
    H1(:,i)  = h;
    H(:,i)   = h/sqrt(sum(h));
end
% Initial objective function
OF = feval(func,H,L);
OF_vec = OF;

% Local Search
if(strcmp(Algorithm,'LS')==1 || strcmp(Algorithm,'ALL')==1)
    % Begin iteration
    sampleInd = randi(k,nitr,1);
    sampleFac = rand(nitr,1);
    for i = 1:nitr
        H1test  = H1;
        k_col   = sampleInd(i);
        indvec  = find(H(:,k_col)==0);
        j       = round(sampleFac(i)*numel(indvec)); if j == 0; j = 1; end
        ind2one = indvec(j);
        H1test(ind2one,k_col) = 1;
        ind2zero = find(H1(ind2one,:)==1);
```

```
        H1test(ind2one,ind2zero) = 0;
        Htest  = H1test./repmat(sqrt(sum(H1test)),NV,1);
        OFtest = feval(func,Htest,L);
        if OFtest < OF
            H1 = H1test; H = Htest; OF = OFtest;
        end
        OF_vec = [OF_vec; OF];
    end
    ID_OP = k*ones(NV,1);
    for i = 1:k-1
        ID_OP(H1(:,i)==1) = i;
    end

%% SIMULATED ANNEALING
elseif(strcmp(Algorithm,'SA')==1 || strcmp(Algorithm,'ALL')==1)

    % Annealing parameter
    Temp  = 50;
    coeff = 0.9;
    % Begin iteration
    sampleInd = randi(k,nitr,1);
    sampleFac = rand(nitr,1);
    for i = 1:nitr
        H1test = H1;
        k_col = sampleInd(i);
        indvec = find(H(:,k_col)==0);
        j = round(sampleFac(i)*numel(indvec)); if j == 0; j = 1; end
        ind2one = indvec(j);
        H1test(ind2one,k_col) = 1;
        ind2zero = find(H1(ind2one,:)==1);
        H1test(ind2one,ind2zero) = 0;
        Htest = H1test./repmat(sqrt(sum(H1test)),NV,1);
        OFtest = feval(func,Htest,L);
        if OFtest < OF
            H1 = H1test; H = Htest; OF = OFtest;
        else
            pa   = exp((OF - OFtest)/Temp);
            prob = [pa; 1-pa];
            y    = randsample(2,1,true,prob); % 1(accept) or 2(reject)
            if(y == 1)
                H1 = H1test; H = Htest; OF = OFtest;
                                                    end
        end
        Temp   = Temp*coeff;
        OF_vec = [OF_vec; OF];
    end

    ID_OP = k*ones(NV,1);
    for i = 1:k-1
        ID_OP(H1(:,i)==1) = i;
```

```
    end

% TABU SEARCH
elseif(strcmp(Algorithm,'TS')==1 || strcmp(Algorithm,'ALL')==1)

    % Tabu list
    Max  = 100;
    Tabu = zeros(Max,2); % [i,pai(i)]

    % Begin iteration
    sampleInd = randi(k,nitr,1);
    sampleFac = rand(nitr,1);
    for i = 1:nitr
        H1test = H1; flag = 1;

        while(flag == 1)

            H1test = H1; % refresh

            k_col = sampleInd(i);
            indvec = find(H(:,k_col)==0);
            j = round(sampleFac(i)*numel(indvec)); if j == 0; j = 1; end
            ind2one = indvec(j);
            H1test(ind2one,k_col) = 1;
            ind2zero = find(H1(ind2one,:)==1);
            H1test(ind2one,ind2zero) = 0;
            Htest = H1test./repmat(sqrt(sum(H1test)),NV,1);

            % check Tabu rule
            flag = 0;
            for j = 1:k; ID(H1(:,j)==1) = j; end
            tabu  = [ind2one ID(ind2one)];
            for j = 1 : Max
                tf = isequal(tabu(:),Tabu(j,:)); if(tf == 1); flag = 1; end
            end
        end
        % Update Tabu list
        if(i > Max)
            Tabu(1:Max-1,:) = Tabu(2:Max,:);
            Tabu(Max,:)     = tabu(:);
        else
            Tabu(i,:)       = tabu(:);
        end

        OFtest = feval(func,Htest,L);
        if OFtest < OF
            H1 = H1test; H = Htest; OF = OFtest;
        end
        OF_vec = [OF_vec; OF];
    end
```

```
    ID_OP = k*ones(NV,1);
    for i = 1:k-1
        ID_OP(H1(:,i)==1) = i;
    end
end

% Plot
figure(4);  imagesc(reshape(ID_OP,nx,ny))
            xlabel('X'); ylabel('Y'); colorbar
            fname = ['RESULTS_' Algorithm];    saveas(gcf,fname,'emf')
figure(5);  plot(OF_vec);
            xlabel('ITERATION'); ylabel('OBJECTIVE')
fname = ['OBJECTIVE_' Algorithm]; saveas(gcf,fname,'emf')

% Disp
TCPU = toc;  format long;
disp('CPUtime of');    disp(Algorithm); disp(TCPU); disp('sec')
disp('Final OBJ of'); disp(Algorithm); disp(OF_vec(end));
```