Cooperative Spatial Regionalization in a Distributed Memory Environment

Jason Laura and Sergio J. Rey

ABSTRACT: Regionalization, a fundamental GIScience research area, is an NP-Hard, complex com- binatorial problem that seeks to aggregate n polygon spatial units into p regions or zones (p ≤ n). We describe a distributed memory implementation of the Max-*p*-Regions spatial regionalization problem using a cooperative bulk synchronization approach. We show that the distribution of solutions is pushed towards optimality (or the current best known) using this approach.

KEYWORDS: Spatial Regionalization, CyberGIS, Spatial Analysis, Max-*p*-Regions, Heuristic

Introduction

Duque, Anselin, and Rey (2012), citing Fischer (1980), defines a region as a 'set of spatially contiguous areas which show a high degree of similarity regarding a set of attributes'. Likewise, Shirabe (2009) suggests that districting is the process of 'aggregating predefined discrete geographic units into larger clusters'. W. Li, Church, and Goodchild (2014b), in defining the p-Compact regions problem, set the attribute to be regional compactness, i.e., the likeness of a region to a circle. Within this work, regionalization is defined as the process by which atomic areal units are combined, under some set of constraints (common constraints include contiguity, compactness, or the application of some objective function), such that larger regions are generated (Duque, Ramos, and Surinach 2007). Broadly, spatial regionalization algorithms, which enforce a contiguity constraint, can be divided into two classes: (1) flow based aggregation, and (2) attribute based aggregation. The former, p-Functional regions problems seek to define regions by interdependence using flows as a key means of aggregation (Duque, Anselin, and Rey 2012; Kim, Chun, and Kim 2015). The latter, p-Regions problems, seek to define regions based on the degree of similarly (or dissimilarity), but not inter-dependence, of atomic units (Duque, Anselin, and Rey 2012). This work focuses on the p-Regions class of problem which seeks to aggregate n atomic units into p regions with n ≤ p (Duque, Church, and Middleton 2011). For the remainder of this work, references to the p-Regions problem focus on attribute (in the case of the Max-P regions problem) or compactness (in the case of the p-Compact regions problem) based aggregation.

The process of spatial regionalization finds wide application in a number of domains. Wise, Haining, and Ma (1997) utilize spatial regionalization within a healthcare context in order to ensure confidentiality and Pathman, Ricketts III, and Konrad (2006) aggregates survey data from the zip code level to generate Primary Care Service Areas for analysis. Wise, Haining, and Ma (2001) suggests that regionalization is an essential exploratory spatial data analysis tool to allow users to experiment with the Modifiable Areal Unit Problem (MAUP) (Gehlke and Biehl 1934; O'Sullivan and Unwin 2010), and identify outliers. Shirabe (2005) illustrate the selection of contiguous parcels in a land use planning case and Shirabe (2009) provides a generalized districting model designed for political, social, or economic regionalization. Openshaw (1977; Morril 1976; Pang et al.

2010) explicitly apply regionalization to the political redistricting problem. Miller and Shaw (2001) aggregate areal units to identify Transportation Analysis Zones (TAZ) and both Gonzalez-Ramirez et al. (2011), and W. Li, Church, and Goodchild (2014a) utilize a compactness constrained regionalization approach to identify optimal delivery zones and TAZs, respectively. With vast increases in total spatial data sizes (Yang et al. 2010) we anticipate additional application of regionalization as a means to reduce total problem size through aggregation, assuming that robust, computationally viable regionalization methods can be implemented.

Described from a computational standpoint, p-Regions algorithms seek to solve complex combinatorial problems and are known to be NP-Hard (Duque, Church, and Middleton 2011). Mixed integer programming (MIP) formulation, used to compute exact solutions, is difficult due to the spatial contiguity constraint. Duque, Church, and Middleton (2011) describe three MIP techniques, using global and connected component sub-graph representations of a study area. Preventing cycling within a sub-graph while ensuring contiguity is maintained adds significant cost to the regionalization process. Duque, Church, and Middleton (2011) describes MIP experiments with runtimes which were capped at three hours, total problem sizes constrained to 49 atomic units ($n = 49$) and optimality achieved only with the smallest number of regions ($p \leq 6$ being the upper limit). Likewise, Kim, Chun, and Kim (2015) utilizes MIP to solve p-Functional regions problems with $n = 25$ and reports run times exceeding 17 hours for the computation of an exact solution, in some instances. Clearly, the addition of the contiguity constraint and the need to process larger, non-trivial data sets in a reasonable amount of time precludes the use of exact, MIP solution methods. Therefore, heuristic solution methods are employed for medium to large data sets where computational complexity is traded for, potentially, solution quality (Duque, Anselin, and Rey 2012; W. Li, Church, and Goodchild 2014a). In instances where the problem set is sufficiently large, exact solutions may not even be feasible.

p-Regions problems are ideally suited for distributed, parallel, implementation. Interest in parallel spatial analysis has recently increased as significant increases in data size, research in highly distributed parallel computing models, and the necessary hardware has become more widely available. It is within the context of the emergent Geospatial Cyberinfrastructure (Yang et al. 2010; Wang 2010; Wang 2013; Wright and Wang 2011) that we cite this work. In contrast to previous works, which have largely focused on decompose-conquer-merge strategies (Wang and Armstrong 2005; Yang et al. 2008; Padmanabhan, Wang, and Navarro 2011; Tang, Bennett, and Wang 2011; Rey et al. 2013) for parallel spatial implementations, spatial regionalization is amenable to cooperative exploration of a solution space. Works focusing specifically on p-Regions problems in parallel domains are limited with Laura et al. (2015) providing a low communication implementation of the p-Compact-Regions problem. Widener, Crago, and Aldstadt (2012) provides an implementation that focuses on the identification of spatial clusters, a loose analog to the p-Regions problem. Outside of the spatial analysis domain, parallel implementations have been designed to solve the quadratic assignment problem (QAP) (Gabrielsson 2007; James, Rego, and Glover 2009a; James, Rego, and Glover 2009b), the generalized assignment problem (GAP) (Liu and Wang 2015) using genetic algorithms, and the multi-commodity capacitated network design problem

(Crainic 2002) using a parallel Tabu Search. Likewise, Ram, Sreenivas, and Subramaniam (1996) and Onbasoglu and Ozdamar (2001) provide parallel heuristic implementations to solve traveling salesman and generalized mathematical optimization problems, respectively.

This work focuses on the implementation of the Max-p regions (MPR) algorithm in a multi-node, distributed memory HPC environment using a cooperative heuristic implementation.

Max-p Regions Problem

In addition to the contiguity and minimization of inter-regional heterogeneity, constraints described above, MPR seeks to endogenously identify the maximum possible number of regions a given a study area can contain, i.e., p is not know a priori. This requires an additional minimum region size constraint, the floor constraint, to ensure that n ≠ p for all realizations.

The objective formulation, as presented by Duque, Anselin, and Rey (2012) of the Max-P Regions problem statement is:

$$\textbf{Minimize:} \tag{1}$$

$$Z = (-\sum_{k=1}^{n}\sum_{i=1}^{n} i^{k0}) * 10^h + \sum_{i}\sum_{j|j>i} d_{ij}t_{ij} \tag{2}$$

$$\textbf{Subject to:} \tag{3}$$

$$\sum_{i=1}^{n} x_i^{k0} \leq 1 \forall k = 1, \ldots, n \tag{4}$$

$$\sum_{k=1}^{n}\sum_{c=0}^{q} x_i^{kc} = 1 \forall i = 1, \ldots, n \tag{5}$$

$$x_i^{kc} \leq \sum_{j \in N_i} x_j^{k(c-1)} \forall i = 1, \ldots, n; \forall k1 =, \ldots, n; \forall c = 1, \ldots, q \tag{6}$$

$$\sum_{i=1}^{n}\sum_{c=0}^{q} x_i^{kc} l_i \geq \text{threshold} * \sum_{i=1}^{n} x_i^{k0} \forall 1, \ldots, n \tag{7}$$

$$t_{ij} \geq \sum_{c=0}^{q} x_i^{kc} + \sum_{c=0}^{q} x_j^{kc} - 1 \forall i, j = 1, \ldots, n | i < j; \forall k = 1, \ldots, n \tag{8}$$

$$x_i^{kc} \in 0, 1 \forall i = 1, \ldots, n; \forall k = 1, \ldots, n; \forall c = 0, \ldots, q; \tag{9}$$

$$t_{ij} \in 0, 1 \forall i, j = 1, \ldots, n | i < j \tag{10}$$

where, k is a vector of potential regions, i is an area within the region, h is a scaling factor,d_{ij} is an element of a dissimilarity matrix, t_{ij} is a binary membership matrix, c is an index into a contiguity order used for maintaining the contiguity constraint, q is the maximum index in a given contiguity order, $\mathrm{w_{ij}}$ is a binary adjacency element, and l is a vector of attribute values for each i.

The first half of the objective function formulation $(-\sum_{\mathrm{k=1}}^{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{i}^{\mathrm{k0}}) * 10^{\mathrm{h}}$ seeks to assign all atomic units to a region, maximize the total number of regions, and ensure that solutions with the maximum number of regions are guaranteed to provide a more optimal solution using a scaling factor (h), while the second half $\sum_i \sum_{j|j>i} \mathrm{d_{ij}}\, \mathrm{t_{ij}}$ sums global variance. This implies that solutions with more regions supersede solutions with fewer regions and that solutions with the same number of regions are rank-able by inter-regional heterogeneity, e.g. similarity. The objective function is subject to a number of constraints designed to enforce a lower threshold on region size and a contiguity constraint.

Citing Duque, Anselin, and Rey (2012) constraint 4 ensures that each region contains one, and only one seed, i.e. $\mathrm{c} = 0$ in the contiguity order. Constraint 5 ensures that each i is assigned to a single region, and a single contiguity order. In order for an atomic unit, i, to be assigned to a region, it must be spatially contiguous to some other atomic unit already in the region. Constraint 6 enforces this requirement. The minimum size threshold constraint is enforced by constraint 7. Constraint 8 ensures that pairwise dissimilarity is utilized as the metric for computing global heterogeneity and constraints 9 and 10 enforce a MIP problem.

A serial implementation of MPR consists of three distinction phases. First, preprocessing must occur in order to read a data set into memory (incurring some Input/Output cost) and a data structure to store spatial contiguity, e.g., an adjacency matrix, must be created. Next, one or more initial feasible solutions (IFS) must be realized. Finally, a local search phase (possibly utilizing Tabu Search, described below) is entered which permutes the IFS, under all constraints, and seeks to improve the objective function value. Given these three distinct phases, it is possible to split the MPR objective function into two parts and assign each part to a given phase. IFS generation seeks to minimize $(-\sum_{\mathrm{k=1}}^{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{i}^{\mathrm{k0}}) * 10^{\mathrm{h}}$, which, as described above, ensure that solutions with a higher number of regions are always favored. Contiguity constraints and permutation through edge reassignment ensures that p cannot vary during the local search phase. Therefore, only the minimization of $\sum_i \sum_{j|j>i} \mathrm{d_{ij}}\, \mathrm{t_{ij}}$ must be considered with each permutation.

Tabu Search

Tabu Search (TS) is a heuristic solution method developed to solve complex combinatorial optimization problems where some optimally definable solution exists within a finite set of potential solutions (Pham and Karaboga 2000; F. Glover 1989b; F. Glover 1989a). The application of TS suggests that fully enumerated solutions are not feasible in a given amount of time. Therefore, TS utilizes a two phase approach: (1) the generation of some initial feasible solution(s) (IFS), and (2) local search. We focus on cooperative parallelization of the second phase and note only that IFS generation can have a significant impact on the final solution based on the initial selection of seed regions.

Local search leverages iterative permutation of a solution space to explore all possible neighbors to the current solution, i.e., the existing solution is slightly permuted. Within the context of MPR, this takes the form of edge reassignment, where a single atomic unit is reassigned to an adjacent region. With each iteration of the local search phase, all possible swaps are enumerated. The process of edge reassignment can become trapped in a local optima, that is a single solution within the solution space which is not the global best, but which is locally inescapable. In order to facilitate escape from local optima, TS uses an aspiration function that accepts either, the realization which most improves the global solution quality (objective function value) or accepts the solution that degenerates the global solution least (Pham and Karaboga 2000). This process can introduce cycling, and a tabu list of prohibited moves is maintained. Local search continues until the maximum number of failures is attained. As moves are accepted they are added to the tabu list, forcing previous moves to increment out of scope. In this way a previously tabu move will become available again. In terms of this usage case, local search swaps all members of a region adjacent to another regions, checks that the swap does not violate the contiguity constraint or floor constraint, and accepts the swap that improves the objective function most.

Parallel Max-p Regions

Interest in parallel implementations of TS began almost immediately after initial serial publication of the heuristic (Taillard 1991) following the initial articulation by F. Glover (1989b) and F. Glover (1989a). Of particular interest to this work, James, Rego, and Glover (2009a; James, Rego, and Glover 2009b) explore variable tabu list length, diversification, intensification, and memory management issues found in parallel TS implementations. Taillard (1991) first introduced variable tabu list length and James, Rego, and Glover (2009a) find that variable TS list length diversifies solution trajectory. This can be implemented as a function of n, the total number of atomic geometries in the input data set, where total TS list length is described by:

$$\begin{aligned} s_{min} &= n \cdot 0.9 \\ s_{max} &= n \cdot 1.1 \\ \Delta &= \{x| \in \mathbb{N}, 0 \leq x \leq (s_{max} - s_{min})\} \\ TS_{list_length} &= n \pm \Delta \end{aligned} \tag{11}$$

Diversification indicates the perturbation method employed to escape a potential local optima. This processes is initiated when a core has been unable to make an improvement to the solution in a given number of failures. Intensification is the process by which the current optimal solution is propagated through the search space and explored by multiple cores. James, Rego, and Glover (2009a) implements intensification by asynchronously testing a single core solution against a shared memory solution space. If the current solution is a new global optima, it is propagated to 50% of the search space, thereby intensifying TS search in that neighborhood. A 'kick' algorithm, which performs a small number of random assignment swaps, is also used to slightly perturb the high quality solution (James, Rego, and Glover 2009a). The addition of intensification helps reduce total parallel processing time as globally, TS does not need to run as long before the algorithm converges. Finally, memory management in SPDS and MPDS implementations is essential to avoid concurrent overwrites and provide a means of masterless communication (p-control). James, Rego, and Glover (2009a) provides strategies to facilitate complex inter-core communication using a series of locks and semaphores. These implementations provide higher quality results than more simplistic single strategy implementations.

In creating a parallel Max-p regions implementation we focus on maximizing the probability of finding a high-quality solution (minimizing the objective function) at some cost to overall performance. To our knowledge, quantification of this relationship has not been performed; a single method with exceptional speed and an assurance of (near) optimality has not been developed. Therefore, we have designed this implementation with the goal of maximizing solution quality. We hypothesize that is possible to increase inter-core communication during the local search phase and leverage intensification and diversification strategies to force the algorithm to converge more quickly. That is, rapid failure to make a successful solution permutation, local to a single core, and subsequent iteration to the next feasible solution within the global solution space reduces global processing time while still locating high-quality solutions.

As with the serial implementation, the parallel MPR consists of three discrete phases. First, file I/O occurs, a spatial contiguity object is generated, and the necessary attribute vectors are generated. This process is identical to the serial approach. Next, i initial feasible solutions (IFS) are generated using an embarrassingly parallel implementation. Finally, local search is performed using a cooperative tabu search (TS). The following describes phases two and three of the implementation.

Within the following section, we refer to the local solution space as the shared memory, local to each compute node, within which c solutions are stored, where c is the number of cores per node. The global solution space is the aggregate of all local solution spaces, e.g., the global solution space in an eight node, eight core per node environment would consist of 8 local storage spaces and $8 * j$ solutions in the global solution space, where j is the size of each local solution space.

We implement a hybrid, distributed and shared memory implementation in order to leverage the efficiency of a low communication cost shared memory space and the scalability of message passing approaches as the total number of available nodes increases. The shared memory code utilizes Python multiprocessing and CTypes. MPI is used to manage all message passing. Throughout, bulk synchronization is used as asynchronous MPI based synchronization proved extremely difficult to manage.

Initial Feasible Solution Generation

Each compute node generates a user defined number of IFS within some number of iterations. A shared memory space and associated lock are created local to each node with a total size equal to $jxn + 1$, where j is the number of IFS to be generated and n is the total number of atomic units. In Equation 12, a sample IFS with $j = 2$ is provided. All references by position are zero offset. At positional index zero, each row holds the current number of regions, 7 and 4 in this example. The remainder of each row contains a representation of the current solution with the value representing membership within a region and the positional index representing the atomic unit ID. For example, in row one region 1 is composed of units 2 and 5. Region 2 is composed of units 1, 3, and 4.

$$\begin{matrix} 7 & 2 & 1 & 2 & 2 & 1 \\ 4 & 1 & 2 & 2 & 1 & 2 \end{matrix} \qquad (12)$$

Once the shared solution space is initialized, each available core begins the generation of an IFS. Recall that the MPR objective function biases solutions with a larger number of regions. Therefore, the parallel generation of IFS must ensure that the local solution space contains the current best known p. Secondary to that goal, diversity should be ensured to leverage the benefits of beginning local search with a diverse set of solutions.

Parallel IFS generation is a two step process which iterates until a maximum number of iterations have been performed. First, each core generates an IFS as per the serial implementation without assigning enclaves. Next, the local solution space is queried for the current maximum and those indices which contain a poorer p count. These queries can be processed in one of three ways: (1) if the current solution is poorer than current maximum, the iteration counter is deincremented, (2) if the current solution bests a currently saved solution at some positional index, the solution space is locked and the current solution added to that index, or (3) if the solution is worse than the current best and poorer than all currently stored solutions the iteration counter is zeroed. Manipulation of the iteration counter local to each core, facilitates node-local load balancing. This process allows a solution space to be iteratively populated, and if a new global best is identified, iteratively repopulated, until all rows contain a solution with an equal number of regions. Once all node-local iterations are completed, a bulk synchronization phase (Valiant 1990) is used to standardize p across the global solution space, where p is the number of regions in a solution. First, p is broadcast to all nodes. This ensures that all nodes know the p value of all other nodes. Local to each node, $\mathrm{p_{local}}$ is compared to all other p, $\mathrm{p_{global}}$. If $\mathrm{p_{local}} < \max(\mathrm{p_{global}})$, a node with the maximum $\mathrm{p_{global}}$ is randomly selected and an asynchronous, MPI managed, Get request is made to copy and replace the local solution with the selected node's solution. Once all solutions within the global solution space have standardized p, enclave assignment is completed in an embarrassingly parallel manner. Within the enclave assignment phase, index zero of each solution is updated to include the current objective function value.

Cooperative Tabu Search

The presented approach runs p concurrent tabu searches from different starting solutions, each with different, stochastically generated TS parameters, and periodically synchronizes results to propagate the 'best' solutions across all workers. Recall that IFS generation completed within each local storage space being composed of a shared $\mathrm{jxn}+1$ solution matrix with the zero index within each row containing the second component of the objective function, $\sum_i \sum_{j|j>i} \mathrm{d_{ij}}\,\mathrm{t_{ij}}$ and the global solution space has a standardized p for all solutions.

Prior to initiating local search five parameters are set. First, each core is parameterized with a maximum number of iterations that define the total number of independent local search phases a given core can make. Next, a maximum number of failures is provided. This value is permuted, local to each code using Equation 13:

$$maxfailures = maxfailures + maxfailures * U(-1.1, 1.2) \tag{13}$$

where U is a uniform distribution. Stochastically computed maximum failures are shown to improve the global solution quality (James, Rego, and Glover 2009a). Third, a constant maximum number of iterations per core is added to control the number of times that a local search will be applied to the local solution space. Fourth, an integer identifier is assigned to each core in order to facilitate traversal across the solution space, described below. Finally, each core determines tabu search list length, a differentiation of the search strategy introduced by Taillard (1991), as defined by:

$$\begin{aligned} &s_{min} = n \cdot 0.9 \\ &s_{max} = n \cdot 1.1 \\ &\Delta = \{x| \in \mathbb{N}, 0 \leq x \leq (s_{max} - s_{min})\} \\ &TS_{list_length} = n \pm \Delta \end{aligned} \tag{14}$$

James, Rego, and Glover (2009a) find that variation of this parameter helps to diversify the trajectory of solution traversal.

The remainder of this section describes the process of solution traversal, intensification, and diversification from the perspective of a single core. This process occurs concurrently across all cores on all nodes. First, the core applies Tabu Search to the solution corresponding to the solution row with the same identifier, i.e., core one works on row one. Once completed, the core locks and queries the global solution space, comparing the newly computed solution to all other local solutions. If the solution is better than the solution currently held in the corresponding index, the local solution space is updated. If the solution is better than all other solutions in the solution space, it is intensified to some percentage of the local solution space. If the current solution is worse than all others, or the solution has not been improved by the Tabu Search, the solution is diversified and then written to the global solution space. Once completed the core identifier is incremented by 1 (or set to zero in the case where index + 1 is larger than j) and the next solution row processed. Intensification and diversification provide two methods for improving the solution quality. Intensification explicitly suggests that the current best solution is either the global optima or a promising path to continue to explore. For this reason, that solution is intensified to some percentage of the solution space for a variable number of cores, with divergent TS parameters, to process. Intensification is parameterized to allow the user to define the percentage of intensification. It is possible that the assumption of high solution quality is erroneous and therefore, intensification is not to 100% of the solution space. Conversely, a non-improving solution is assumed to be a dead-end. Therefore, the solution is diversified, in order to significantly alter the current realization, prior to allowing the next core to perform local search. Diversification is achieved by selecting the region composed of the maximum number of atomic units and iteratively reassigning edge units to adjacent regions until any additional atomic unit removal would violate the floor constraint.

Experiments

Two different data generation processes were employed in order to explore the impact of varying Tabu Search parameters, intensification and diversification strategies, and the impact of spatially autocorrelated data. Within this secretion we first describe the data generation process and then present test performed.

The data set shown in Figure 1 was generated in an effort to control the regionalization process and identify a known optimal solution. To that end, both random and spatially clustered point patterns were generated over a defined extent. From these point patterns, Voronio diagrams were generated and the area of each polygon computed. Using a derivation of the IFS generation code, regions were generated using random initial seed assignment and grown until a threshold total area achieved. Once this process was completed, enclaves were assigned to the smallest adjacent region in an effort to balance total area without the need to perform a complete area based regionalization. All members of a given region were attributed the same region identifier. Recall from Equation 1 that the right hand side of the objective function describes sum of the inter-regional variance. By using the region identifier as that attribute, it is known a priori that the right hand size should sum to zero. It is not possible guarantee that the known solution is in fact an optimal space partitioning at any other p value, only that it is optimal at the p defined in the data generation process. Random and clustered data sets with 100 polygons were generated.

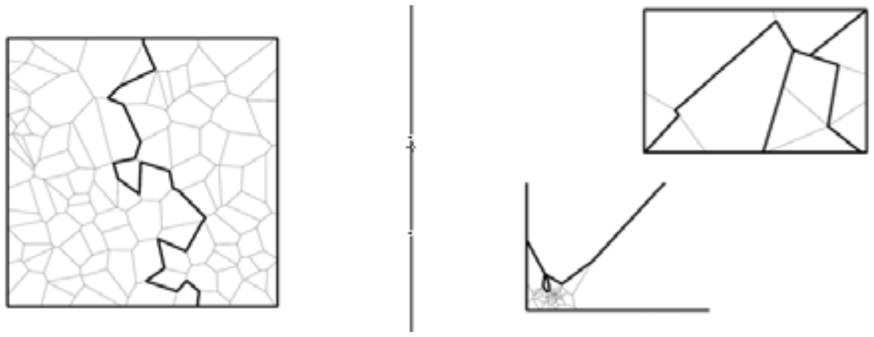
Figure 1: Synthetically Generated Data: 100 Randomly Distributed Polygons (left) and 100 Clustered Polygons (right) with Inset to Highlight Tight Clustering. All Figures Include Region Overlay of the Known Optimal Partitioning at a given p.

Five tests were performed to explore the impact of varying Tabu Search parameters, as well as the impact of increased quantity and type of inter-core communication. First, TS search parameters were allowed to vary. These parameters include the Tabu List length and total number of local search move failures before termination. Next, solution space traversal iterations from the set 1, 2, 4, 8, 16, 32, 64 were tested. That is, each processing core is allowed to iterate around the solutions space some number of times, before processing is terminated. As in the first test TS parameters are allowed to vary. Third, the maximum number of iterations was set to 16, TS parameters allowed to vary, and intensification percentages between 20% and 80% at 20% intervals tested. Next, intensification strategies were disabled and diversification enabled, meaning that a non-improving solution is permuted with the assumption that non-improvement is indicative of either poorly defined TS parameters or a local optima. Finally, TS parameters were allowed to vary, maximum iterations fixed to 16, intensification to 60%, and diversification enabled. Given the stochastic nature of the algorithm, each test was run 400 times.

All experiments were performing using four nodes from the GeoDa Center computing cluster, a homogeneous high performance computing environment. Each node is composed of two quad-core Intel Xeon X5355 processors running at 2.93GHz. All cores have access to 16 Gigabytes of shared memory RAM and are inter-connected via a high speed infiniband network. Input shapefiles, described below, are stored on a network Lustre file system that is accessed by the rank 0, managing node.

Results

In Figure 2 we illustrate that iteration succeeded in decreasing the mean objective function value for all tests. The 100 random polygon tests at iterations equal to one and two highlight the random nature of the algorithm as these two found the best, non-optimal solution. For all tests, significant improvement between the IFS objective function and the final objective function is observed. In the randomly distributed data, cases where a processing cores iterates 8 to 64 times around a solution space resulted in significantly higher convergence to local optima. The same phenomena is observable in the clustered data, though the convergence is not as distinct.

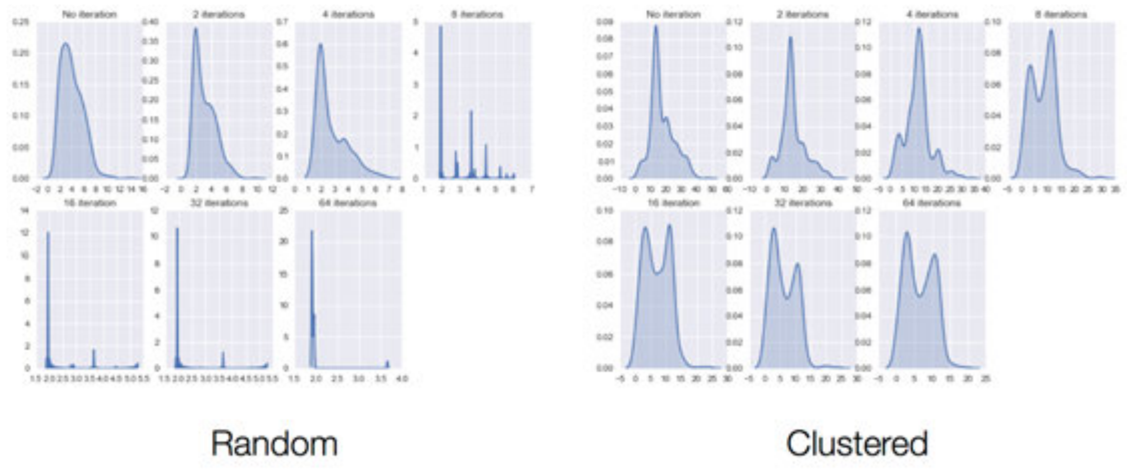

Figure 2: Distribution of Solution Values in Random and Clustered, 100 Polygon Datasets Achieved by Varying the Total Iteration Count.

Setting the maximum number of iterations to 16 and testing intensification between 20% and 80% at 20% intervals, Figure 3 illustrates significant improvement to the best known objective function and the frequency with which the known optima was reached. Unclustered data consistently reached the optima more frequently (395 / 400 tests) than clustered data (355 / 400 tests). Additionally, no correlation is observed between the quantity of intensification and the frequency with which the optimal solution was achieved.

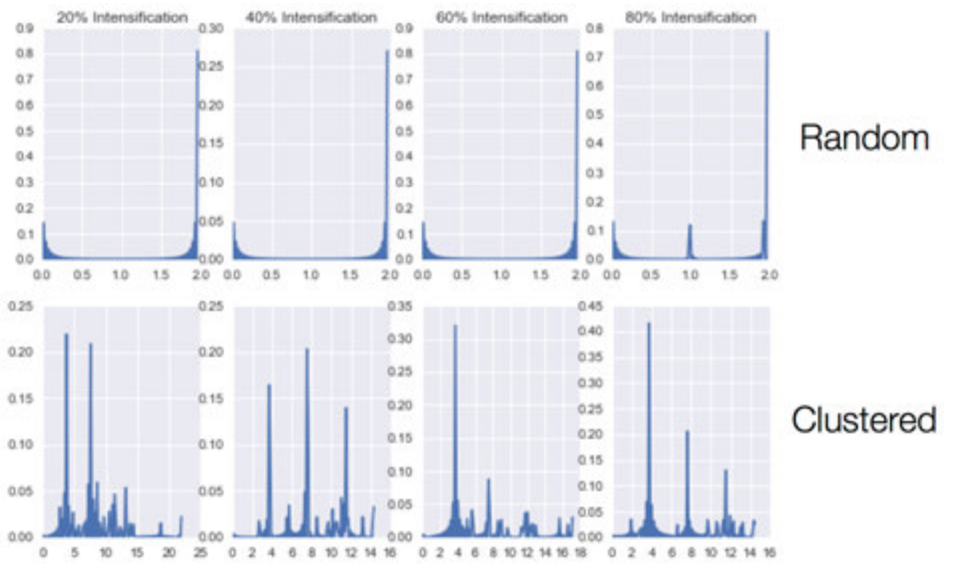

Figure 3: Solution Distributions with Intensification Between 20% and 80% for the 100 Random and Clustered Polygon Tests.

Removing all intensification and setting iterations to 16 it is clear that diversification alone is not sufficient to escape local optima. In fact, the frequency with which a high quality solution is found suggests that intensification is significantly more important than the diversification algorithm tested. Comparing these results to the intensification results, it is clear that the diversification operator is performing too well. That is, the solution is being diversified sufficiently that the algorithm is becoming trapped in a local optima, $1.9\overline{1}$ and 0.9767 for random and clustered data, respectively.

Discussion

Intuitively, spatial regionalization is a stochastically driven process. In a highly parallel environment, a frequent question is: why apply local search, the most computationally expensive process, when a higher number of IFS could be generated in the same amount of time? Using stochastically driven seed selection, region growth, and enclave assignment processes, we show significant improvement with the application of local search. To our knowledge, no study has been performed to quantify the component contribution of each aspect of IFS generation, and until such a study is completed, local search is an essential component of high-quality solution generation.

The first experiment, testing the maximum number of iterations around the solution space also shows that simply varying Tabu Search parameters is insufficient in locating an optimal solution. In instances where cooperative complexity would be too high, simple iteration across solutions with different TS search parameters does more frequently drive the algorithm to higher quality solutions.

Combined with iteration, intensification significantly improves the performance of a cooperative Tabu Search driven regionalization. While it clear that intensification is a key strategy, without iteration, it would not be possible to realize any significant gains. That is, the benefit to the solution is realized by intensification, but intensification is only successful if iteration across a solution space is performed. We also see that diversification must be sufficient to escape local optima, but not so destructive as to essentially constitute the creation of a new IFS. Additional work is necessary to understand criteria for assessing diversification so that a diversification operator can be derived. We caution against the wholesale avoidance of diversification as it plays an essential role in the operations research literature.

The cumulative inclusion of iteration across the solution space, intensification at 40%, and diversification illustrate the potential gains achievable using a parallel, cooperative search strategy. The methods described, at the parameters tested, do not preclude the algorithm becoming trapped in a local optima, but this method demonstrates that the likelihood of convergence to a (near) optimal solution is significantly increased. As above, we suggest that diversification, as implemented, has a negative impact on the regionalization process. This may be because of the extent of diversification, i.e. shrinking the largest region to minimal size may be too drastic of a diversification, is too large, or that the diversification is rapidly reversed returning to a local optima.

Conclusion

This work has focused on the application of a cooperative, multi-start parallel Tabu Search implementation to solve synthetically generated spatial regionalization problems. Solution quality has been the paramount goal. In the short term, future work will focus on testing inter vs. intra node intensification. That is, is it more efficient to perform bulk synchronization at some interval during local search across all nodes, or more efficient to keep intensification local to each individual node. Longer term work will focus on improvements to speed, without attention to solution quality, the quantification of the speed - quality relationship, and finally the development of a hybrid approach which allows the end user fine-grained control over where along the speed - quality boundary they wish to process. From here, it may be possible to identify convergence criteria to allow a highly communicative implementation to collectively terminate computation due to a non-improving solution. Additional work leveraging spatial regionalization will focus on an analysis of regionalization as a preprocessing step for other spatial analysis tasks, e.g. map classification. Through this work, we sought to apply the above methods and quantify potential performance gains, and the associated accuracy reductions, achievable through regionalized data size reduction.

References

Crainic, Michel, Teodor Gabriel anf Gendreau. 2002. “Cooperative Parallel Tabu Search for Capacitated Network Design.” *Journal of Heuristics* 8: 601–27.

Duque, Juan C., Luc Anselin, and Sergio J. Rey. 2012. “The Max-P-Regions Problem.” Journal of Regional Science 52 (3): 397–419. doi:10.1111/j.1467-9787.2011.00743.x.

Duque, Juan C., Richard L. Church, and Richard S. Middleton. 2011. “The P-Regions Problem.” Geographical Analysis 43 (1): 104–26.

Duque, Juan C., R. Ramos, and J. Surinach. 2007. “Supervised Regionalization Methods: A Survey.” International Regional Science Review 30: 195–220.

Fischer, M. M. 1980. “Regional Taxonomy: A Comparison of Some Hierarchic and Non-Hierarchic Strategies.” Regional Science and Urban Economics 10 (4): 503–37.

Gabrielsson, S. 2007. “A Parallel Tabu Search Algorithm for the Quadratic Assignment Problem.” Master’s thesis, University of Toronto.

Gehlke, C. E., and Katherine Biehl. 1934. “Certain Effects of Grouping Upon the Size of the Correlation Coefficient in Census Tract Material.” Journal of the American Statistical Association 29 (185A): 169–70. doi:10.1080/01621459.1934.10506247.

Glover, F. 1989a. “Tabu Search—Part I.” ORSA Journal on Computing 1 (3): 190–206. doi:10.1287/ijoc.1.3.190.

———. 1989b. “Tabu Search—Part II.” ORSA Journal on Computing 2 (1): 4–32. doi:10.1287/ijoc.2.1.4.

Gonzalez-Ramirez, R., N. R. Smith, R. G. Askin, P.A. Miranda, and J.M. Sanchez. 2011. “A Hybrid Metaheuristic Approach to Optimize the Districting Design of a Parcel Company.” Journal of Applied Research and Technology 9 (1): 19–35.

James, Tabitha, Cesar Rego, and Fred Glover. 2009a. “A cooperative parallel tabu search algorithm for the quadratic assignment problem.” European Journal of Operational Research 195 (3): 810–26.

James, Tabitha, César Rego, and Fred Glover. 2009b. “Multistart Tabu Search and Diversification Strategies for the Quadratic Assignment Problem.” IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 39 (3): 579–96.

Kim, Hyun, Yongwan Chun, and Kamyoung Kim. 2015. “Delimitation of Functional Regions Using a P-Regions Problem Approach.” International Regional Science Review 38 (2): 235–63. doi:10.1177/0160017613484929.

Laura, J., W. Li, Serge J. Rey, and L. Anselin. 2015. “Parallelization of a Regionalization Heuristic in Distributed Computing Platforms – a Case Study of Parallel-P-Compact-Regions Problem.” International Journal of Geographical Information Science 29 (4): 536–55. doi:10.1080/13658816.2014.987287.

Li, Wenwen, Richard L Church, and Michael F Goodchild. 2014a. “An Extendable Heuristic Framework to Solve the P-Compact-Regions Problem for Urban Economic Modeling.” Computers, Environment and Urban Systems 43: 1–13.

Li, Wenwen, Richard L. Church, and Michael F. Goodchild. 2014b. “The P-Compact-Regions Problem.” Geographical Analysis 46 (3): 250–73.

Liu, Yan Y., and Shaowen Wang. 2015. “A Scalable Parallel Genetic Algorithm for the Generalized Assignment Problem.” Parallel Computing 46: 98–119. doi:http://dx.doi.org/10.1016/j.parco.2014.04.008.

Miller, H.J., and S. Shaw. 2001. Geographic Information Systems for Transportation: Principles and Applications. Oxford University Press.

Morril, R.L. 1976. “Redistricting Revisited.” Annals of the Association of American Geographers 66: 548–66.

Onbasoglu, E, and L. Ozdamar. 2001. “Parallel Simulated Annealing Algorithms in Global Optimization.” Journal of Global Optimization 19 (27-50).

Openshaw, S. 1977. “A Geographical Solution to Scale, and Aggregation Problems in Region-Building, Partitioning, and Spatial Modeling.” Transactions of the Institute of British Geographers, 169–84.

O’Sullivan, D., and D. J. Unwin. 2010. “The Pitfalls and Potential of Spatial Data.” In Geographic Information Analysis. John Wiley & Sons, Ltd.

Padmanabhan, Anand, Shaowen Wang, and John-Paul Navarro. 2011. “A CyberGIS Gateway Approach to Interoperable Access to the National Science Foundation TeraGrid and the Open Science Grid.” In

Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery, 42:1–:8. TG ’11 42. 42:1–42:8: ACM. doi:10.1145/2016741.2016786.

Pang, S., H. He, Y. Li, T. Zhou, and K. Xing. 2010. “An Approach to Redistricting Based on Simple and Compactness.” In Advances in Swarm Intelligence, 6145:415–24. Lecture Notes in Computer Science. Springer Berlin Heidelberg.

Pathman, D.E., T.C. Ricketts III, and T.R. Konrad. 2006. “How Adults’ Access to Outpatient Physician Services Relates to the Local Supply of Primary Care Physicians in the Rural Southeast.” Health Research and Educational Trust.

Pham, D.T., and D. Karaboga. 2000. Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. London, UK: Springer.

Ram, D.J., T.H. Sreenivas, and K.G. Subramaniam. 1996. “Parallel Simulated Annealing Algorithms.” Journal of Parallel and Distributed Computing 37: 207–12.

Rey, Sergio, Luc Anselin, Robert Pahle, Xing Kang, and Philip Stephens. 2013. “Parallel Optimal Choropleth Map Classification in PySAL.” International Journal of Geographical Information Science 27 (5): 1023–39.

Shirabe, Takeshi. 2005. “A Model of Contiguity for Spatial Unit Allocation.” Geographical Analysis 37 (1): 2–16.

———. 2009. “Districting Modeling with Exact Contiguity Constraints.” Environment and Planning B: Planning and Design 36 (6): 1053–66.

Taillard, E. 1991. “Robust taboo search for the quadratic assignment problem.” Parallel Computing 17 (4-5): 443–55.

Tang, W., D.A. Bennett, and S. Wang. 2011. “A Parallel Agent-Based Model of Land Use Opinions.” Journal of Land Use Science 6: 121–35.

Valiant, Leslie G. 1990. “A Bridging Model for Parallel Computation.” Communications of the ACM 33 (8): 103–11. doi:10.1145/79173.79181.

Wang, S. 2013. “CyberGIS: Blueprint for Integrated and Scalable Geospatial Software Ecosystems.” International Journal of Geographical Information Science 27 (11): 2119–21.

Wang, S., and Marc P. Armstrong. 2005. “A Theory of the Spatial Computational Domain.”

Wang, Shaowen. 2010. “A CyberGIS Framework for the Synthesis of Cyberinfrastructure, GIS, and Spatial Analysis.” Annals of the Association of American Geographers 100 (3): 535–57.

Widener, Michael J., Neal C. Crago, and Jared Aldstadt. 2012. “Developing a Parallel Computational Implementation of AMOEBA.” International Journal of Geographical Information Science 26 (9): 1707–23.

Wise, S.M., R.P. Haining, and J. Ma. 1997. “Regionalisation Tools for Exploratory Spatial Analysis of Health Data.” In Recent Developments in Spatial Analysis: Spatial Statistics, Behavioural Modelling, and Computational Intelligence, 83–100. Springer.

Wise, Stephen, Robert P. Haining, and Jingsheng Ma. 2001. “Providing Spatial Statistical Data Analysis Functionality for the GIS User: The SAGE Project.” International Journal of Geographical Information Science 15 (3): 239–54.

Wright, D. J., and S. Wang. 2011. “The Emergence of Spatial Cyberinfrastructure.” PNAS 108 (14): 5488–91.

Yang, Chaowei, Wenwen Li, Jibo Xie, and Bin Zhou. 2008. “Distributed geospatial information processing: sharing distributed geospatial resources to support Digital Earth.” International Journal of Digital Earth 1 (3): 259–78.

Yang, Chaowei, Robert Raskin, Michael Goodchild, and Mark Gahegan. 2010. “Geospatial Cyberinfrastructure: Past, Present and Future.” Computers, Environment and Urban Systems 34 (4): 264–77.

Jason Laura, Geographer, United States Geologic Survey, Astrogeology, Flagstaff, AZ 86001

Sergio J. Rey, Professor, School of Geographical Sciences and Urban Planning, Arizona State University, Tempe, AZ 85287