성북지역 녹색경제 이행과 메콩유역 농업부문 융합혁신 전략 연구

오근택, 강상인

[ABSTRACT]

본 연구는 2014 년부터 진행된 KEI 녹색경제 협동연구의 일부로 녹색경제 확산을 위한 국제협력 방안을 모색하기 위해 진행된 것이다. 2016 년 연구에서는 아태지역 내 주요 개발도상국들의 녹색경제 전환을 위한 정책동향 분석을 첫 번째 목표로 한다. 연구의 두 번째 목표는 메콩유역 국가인 베트남, 캄보디아, 라오스의 농산업부문의 융합혁신을 통한 녹색경제 성장전략 연구이다. 이 같은 두 개 연구목표의 설정은 본 연구의 기획 과정에서 녹색경제 확산이라는 포괄적 협동연구 주제와 메콩지역에 대한 심층연구를 함께 다루도록 결정된 것에 기인한다. 첫 번째 연구목적에 할당된 본 연구의 제 2 장에서는 먼저 2015 년 채택된 UN 지속가능발전목표(SDGs) 가운데 물, 에너지, 농산업부문 목표들의 상호 연관성에 대해 분석하였다. 함께 분석된 아태지역 내 주요 개발도상국의 사회경제개발계획은 해당 국가의 녹색경제 이행 동향을 보여주는 것으로 향후 역내 국가들을 대상으로 한 녹색경제 확산 관련 정책연구 기반을 제공하는 것이다. 분석 과정에서 역내 국가들은 공통적으로 사회경제발전계획의 이행에 필요한 개발재원 부족에 직면해 있음이 확인되었으며, 본 장에서는 이를 해결하기 위한 개발금융 및 기후금융 활용방안을 함께 살펴보았다. 제 2 장 1 절에서 살펴본 물과 에너지 및 농업 관련 유엔 SDGs 는 상호 밀접한 연관관계를 형성하고 있었으며, 특히 기아해소 및 식량안보와 관련된 농업부문의 목표는 SDGs 의 궁극적인 목표라 할 수 있는 빈곤해소의 바탕이 되는 것으로 나타났다. 이러한 분석은 본 과제의 지역연구 대상인 메콩유역의 3 국이 농업부문에서 수자원의 효율적 관리와 지속가능한 에너지시스템 적용을 통한 스마트농업을 활성화함으로써 국가경제 전체의 녹색경제 전환을 선도해야 한다는 전략적 정책방향의 타당성을 뒷받침할 수 있는 이론적 근거를 제시하는 것이다.

[KEYWORDS] *Agriculture*

1. INTRODUCTION

This study was conducted as part of the KEI Green Economy Cooperative Research from 2014 to seek international cooperation methods for the proliferation of green economy. The first research objective of the 2016 study is to analyze policy trends for the transition of green economy to major developing countries in the Asia-Pacific region. The second objective of the study is a research on green economic growth strategy through convergence innovation in Vietnam, Cambodia and Laos agriculture sector in Mekong basin countries. The setting of these two research objectives is based on the decision to deal with the comprehensive cooperative research subject of green economic proliferation and the in-depth study on the Mekong region in the planning process of this study. In the second chapter of this study, which was assigned to the first research objective, we first analyzed the interrelationships among goals of water, energy and agriculture sector of the United Nations Sustainable Development Goals (SDGs) adopted in 2015. The socioeconomic development plans of the major developing countries in the Asia-Pacific region, which are analyzed together, show the trends of green economic transition in the respective countries. In the process of analysis, it has been confirmed that the countries in the region are faced with a shortage of development resources necessary for the implementation of the socioeconomic development plan. This chapter also explored the ways of utilizing development finance and climate finance to solve these problems. The water, energy and agricultural SDGs of the UN have been closely linked to each other, especially in the context of food security and solutions against starvation. The goals of the agricultural sector in relation to food security are poverty reduction, which is one of the ultimate goals of the SDGs. This analysis is based on the strategic direction of the three countries of the Mekong Basin, which are the subject of this project, to lead the transition of the national economy to the green economy through the efficient management of water resources and the activation of smart agriculture through the application of sustainable energy systems. This analysis provides a theoretical basis to support validity. Although the socioeconomic status of developing countries in Asia-Pacific, analyzed in Section 2, has the constraints of stable macroeconomic operations due to the high dependence on the world economy, especially on the Chinese economy, they established and start implementing mid/long-term plans for sustainable development through the green economy transition as the top national development goal. Section 3 addresses the shortage of development resources needed to implement the mid- and long-term socioeconomic development plans that most developing countries in the Asia-Pacific region face in common. The lack of development resources has become a major factor impeding the continued growth of developing countries in the region, with the deterioration of the national debt caused by the financial crisis and the weakening of the global economic growth due to the slowdown of China`s economic growth. In this regard, this section discusses the development of the financial resources provided by major donors of the international community, including the Green Climate Bank, which will strengthen its financial base with the conclusion and entry into force of the Paris Agreement, as a useful financial means to mitigate the constraints on the development financing of developing countries, and it is necessary to utilize them in promoting green economic cooperation projects of Korea in developing countries in the future. The selection and writing of the detailed research topics covered in Chapters 3 and 4 of this study, assigned to the second research objective, which was conducted with Economic Institutes under the Academy of Social Science in Vietnam, Laos, and Cambodia, Rural Economic Research Institute and World Economic and Social Research Institute. In Chapter 3 of this study, firstly, the direction of national development in accordance with the conditions of each country is suggested based on the analysis of macroeconomic situation of the Mekong basin. Then, we analyzed the conditions of the green economy transition in the agricultural sector, focusing on the mid- and long-term development plan of the agricultural sector and the concept of the agricultural value chain, which are major growth sectors in the socio-economic development of these countries, and also, the demand for smart agricultural convergence innovation policy. The three Mekong countries have experienced an average annual growth rate of 6~8% over the last 10 years. As shown in the agriculture development strategy 2025 and vision 2030 (Laos), the agricultural sector

strategy development plan 2014-2018 (Cambodia), the agricultural production development plan 2020 and the vision 2030 (Vietnam) in the agriculture sector, were adopting a strategy to develop agriculture sector related to green economy transition which is unique to each country. However, from the viewpoint of the agricultural value chain, it was analyzed that it has various challenge factors. such as, deactivate the special input market such as fertilizer, agricultural machinery, and seeds, government support to a limited region, lack of postharvest management technology of small farmers, agricultural market structure led by agricultural products collecting merchandiser, lack of distribution infrastructure such as roads and railroads, lack of processing primary agricultural products, and lack of export competitiveness. The agricultural industry value chain provides an analytical framework for an integrated view of the front-to-back linkage sector of the agricultural sector. It also provides an important concept frame for the development of smart agricultural activation policies by introducing more efficient production and distribution. The main sources of funding and the existing partnership case related to smart agricultural proliferation by donor countries will provide an effective policy-making basis to avoid overlap with existing projects in the process of Korea`s aggressive agricultural sector development cooperation projects. In addition to summarizing the main research results in the previous chapters, Chapter 4, which is the conclusion of this study, presents the direction of the derived convergence innovation strategy in agriculture sector and the topic of follow-up study as an integrated approach to the water management and food security issues at Mekong Basin as a joint research subject of the next year agreed upon by the participating agencies. This study also suggests that the policy research institutes of the three Mekong countries participating in this study should share their experiences through the annual policy research consultation process from the viewpoint of practical policy formulation for the implementation of convergence innovation in agriculture sector. And various local case studies were included as appendices to provide reference material on useful practices for incorporating this into actual policies. Water management efficiency, alternative wetting system using elite rice seeds, hydroponic fertilizer system that combines organic shrimp farming, forest and food crops, customized agricultural information support system based on mobile, integrated agricultural system through agriculture diversification and productivity improvement project using chaff are the examples of proliferation of the convergence industrialization policy and global agricultural industry convergence innovation of international organizations and major countries. The results of the case studies included in the appendices show that the introduction of smart agriculture, which can increase resource utilization efficiency in agricultural production and distribution processes and enable more environmentally friendly production and consumption methods, and climate smart agriculture, a new agricultural system that can contribute to strengthening adaptation capacity while reducing greenhouse gas emissions in agriculture sector with high vulnerability to climate change at the same time. And it confirmed that these are the key to the sustainable development and green economy transition of the three Mekong countries, which are highly dependent on agricultural sector development.

2. The Randomized Neighborhood Graph

In this section we discuss our algorithm for finding approximate nearest neighbors for a set of n points S $\subset$ Ed. Our approach is based on some simple techniques, which can be viewed as a generalization of a “flattened” skiplist in higher dimensions [16]. The data structure itself consists of a directed graph (with some additional structural information) whose vertex set is S and such that each vertex has degree O(log n). For each point p $\in$ S we cover Ed with a constant number of convex cones all sharing p as a common apex, and whose angular diameter δ is bounded above by a function of . The cones need not be circular. A method for constructing such a set of cones is given by Yao [23]. The number of cones centered at p is a function of d and but independent of n. For each of these cones, we add a directed edge from p to O(log n) points of S lying within the cone. We determine these neighbors of p by the following randomized process. The points of S$-${p} are permuted randomly. The rank of each point in this permutation is called its index relative to p. For each cone c centered at p, we consider the points r lying within this cone. An edge from p to r is added if r is the nearest to p among all

points of lower index in this cone. The resulting set of neighbors, denoted Nc[p], is stored in a list. It follows from standard probabilistic arguments that the expected degree of a point is O(log n). If necessary, by repeating this process a constant number of times in the expected case, we can guarantee that each vertex has degree O(log n). Observe that if a cone is nonempty then there is at least one neighbor of p in the cone (namely the point closest to p in the cone). The resulting graph is called the randomized neighborhood graph for S, and is denoted NGd(δ, S). An example of this applied to a single cone is shown in Fig. 1. One property of the randomized neighborhood graph is given in the following lemma. Lemma 2.1. Given any > 0, there is an angular diameter δ (depending on) such that given any query point q and any point p ∈ S, if p is not a (1 +)-nearest neighbor of q, then there is a neighbor of p in NGd(δ, S) that is closer to q than p is. Proof. (Sketch) Suppose that p is not a (1 +)-nearest neighbor of q. Normalize distances so that p lies on a sphere of radius (1 +) centered at q. (Throughout, unless otherwise stated, we use the term sphere to signify a (d − 1) dimensional hypersphere centered at the query point.) Then there is a point r that lies within a sphere of radius 1 centered at q. Define the angular distance between any two points that are equidistant from p to be the angle between two rays emanating from p passing through these points, and define the angular distance between these two spheres to be the infimum of this distance among all equidistant pairs, one point taken from each of the spheres. It is a straightforward geometric exercise to show that for > 0 the angular distance between these two spheres is greater than zero. Let δ be any positive value less than this angular distance. Consider a cone whose apex is at p that contains r. If p has an edge to r then we are done. If not there must be a neighbor s of p in this cone that is closer to p than r is. Again, it is a straightforward geometric argument that, given our choice of δ, s is closer to q than p is, completing the proof. ut This lemma implies that, starting at any point p ∈ S, we can walk to a (1 +)-nearest neighbor of the query point q along a path whose distances to q decreases monotonically. One might imagine any of a number of different search strategies. For example, a simple greedy search would be, from each point p, visit next the neighbor of p that is closest to the query point. In spite of its intuitive appeal we do not have bounds on the asymptotic performance of greedy search. Our search strategy is based on a modification of a simple randomized strategy. We give an intuitive explanation of the simple strategy and why it fails. Let p be the point that is currently being visited by the search, and define the set Cl(p) to be the subset of S whose distance to q is strictly less than p's distance to q. These points lie within a sphere centered at q whose radius is the distance from q to p, dist(q, p). Consider the point r ∈ Cl(p) of lowest index with respect to p. Since r could be any point of Cl(p) with equal probability, the number of points of Cl(p) that are closer to q than r is expected to be roughly |Cl(p)|/2. Thus, if r is a neighbor of p in NGd(δ, S), by moving from p to r, we eliminate half of the remaining points from consideration in the expected case. The problem with this proposed search strategy is that r need not be a neighbor of p, and so such a transition may not be possible. To understand why this is, we introduce a concept called pruning. We say that a point r lying within Cl(p) is pruned if, for all cones centered at p that contain r, there exists some point r0 lying inside the same cone and outside Cl(p) (and hence further from q than p) having lower index than r (relative to p), such that dist(p, r0) < dist(p, r). See Fig. 2. Clearly, if r is pruned then it is not a neighbor of p. Thus r0 has effectively eliminated r as a possible neighbor of p, but because we demand that the path to q be monotonically decreasing in distance, we are not able to visit r0 . In order to get around the pruning problem we exploit a few basic properties about the randomized neighborhood graph and pruning. We state these intuitively here, but they are made precise in the proof of Lemma 2.2 and affect the choice of δ, the angular diameter of the cones. First, because pruning occurs within cones (and not between cones) it is confined locally to points lying relatively near the surface of the sphere (centered at q and of radius dist(q, p)). Before stating the second fact we give a definition. We assume that the sets of cones centered around the points of S are equal up to translation. Each directed edge of the neighborhood graph is naturally associated with a cone centered at its tail, which contains the head of the edge. A path p1, p2,...,pk in the randomized neighborhood graph is said to be pseudo-linear if the associated cones for every edge on the path share a common axis. See Fig. 3(a). Our interest in pseudolinear paths is that they

behave very much like paths that arise in a one-dimensional skiplist because later cones on the path contain a subset of the data points, and hence we can easily measure progress by the number of points eliminated. The second property is that if p is not a (1 +)-nearest neighbor of q, then there exists a pseudo-linear path from p of expected length O(log n) to a point that lies closer to q than any of the pruned points (and this path can be constructed in O(log2 n) expected time). Intuitively, it is this second observation that allows us to circumvent the problem of pruning by "shortcutting" around the pruned points along a path of logarithmic length. We summarize these observations in the following lemma. Lemma 2.2. Given a set of n points S in Ed and a constant > 0, there exists δ > 0 (a function of), such that given any query point q ∈ Ed and any point p ∈ S, the set of points Cl(p) ⊂ S of points closer to q than p, can be partitioned into three subsets, Pr (p) (for "prunable") In(p) (for "inner"), and Im(p) (for "intermediate") such that: (i) the pruned points of Cl(p) lie only in Pr (p), (ii) the points of In(p) are all closer to q than any point of Pr (p) ∪ Im(p), (iii) if p is not a (1+)-nearest neighbor of q, then there exists a pseudo-linear path in NGd(δ, S) from p to a point of In(p), traveling through Im(p), that can be computed in O(log2 n) expected time, (iv) membership in each of these sets can be determined in O(1) time. Proof. (Sketch) Before giving the proof of the lemma, we need to give some terminology and make some observations. Let δ denote the angular diameter of the cones (this value will be determined later). Let c be the index of some cone, and let conec(p) denote the geometric cone whose apex is at p. We shall regard cones with the same index and centered at different points as having the same shape and orientation. Let corec(p) denote the points lying within conec(p) whose angular distance from the central axis of the cone is some small fraction of δ (any fraction less than 1/2 will work). Then given any point r inside corec(p), it can be shown that, for any point s inside conec(p) and sufficiently close to p (relative to the distance between p and r), r lies within the parallel cone centered at s, conec(s). Second observe that we can partition space so that every point lies within the core of some cone, by first covering space with smaller cones having diameter that of the core, and then growing these smaller cones to attain the diameter δ. Now we return to the proof of the lemma. Normalize distances so that p lies on a sphere of radius (1 +) from q, called the outer sphere. Let the base sphere be a sphere of radius 1 centered at q and let the inner sphere be a sphere also centered at q which lies between the base sphere and outer sphere, and whose radius is 1 + αfor some suitably chosen α < 1. We will construct δ sufficiently small so that no point within the inner sphere can be pruned (this also ensures that no point within the base sphere can be pruned). Let C0 denote the subset of cones centered at p whose cores intersect the base sphere. Assuming δ is sufficiently small, each cone in C0 is cut by the inner sphere into a finite cone, called a cap, whose apex is at p and whose base lies on the inner sphere. We choose α close enough to 1 so that, for any point r in the base sphere lying in the core of some cone, and any point s in the cap of this cone, the ratio of distances between p and s, and p and r is sufficiently small. This allows us to use our earlier observations to claim that r lies within the parallel cone centered at any point in the cap. Let Im(p) be the set of points lying in the caps for each cone in C0 , let In(p) be the set of points lying in the inner sphere, and finally let Pr (p) be all remaining points. See Fig. 3(b). Facts (ii) and (iv) follow immediately from our definitions. It is not hard to show that for sufficiently small δ the points in Im(p) cannot be pruned, from which (i) follows. To show (iii), recall that if p is not a (1+)-nearest neighbor of q, then there is a point r inside the base sphere lying within the core of some cone in C0 . Although we do not know which cone it is, we can try them all, since there are only a constant number of cones. For each cone index c we restrict attention to the data points lying inside conec(p) and do the following. First we check if there is an edge from p to any point in the inner sphere and lying inside conec(p). If yes, we are done. Otherwise if there is an edge from p to a point in the cap, then we select such a point s of lowest index and repeat the procedure at point s (for details, see the while loop in the pseudo-code below). If there is no such point we go on to examine the next cone index. The point s of lowest index in the cap is a random point in the cap, and since the parallel cone centered at s is contained within p's cone, we expect at most half of the remaining data points of the cap to lie within s's cone. Thus in the expected case, after O(log n) such steps, each step taking O(log n) time, we terminate. This gives an expected cost

of O(log2 n) for this procedure. At termination, we are guaranteed to find a point that lies within the inner sphere because if point r inside the base sphere lies within corec(p), then it also lies inside every parallel cone centered at every point inside the cap of conec(p). Thus for cone index c we must finally arrive at a point in the inner sphere. ut The search algorithm operates as follows. We assume that the randomized neighborhood graph NGd(δ, S) has already been computed. This can be done easily in O(n2) expected time. The starting point p can be any point in S initially. Letting p denote the current point being visited, consider p's neighbor of lowest index lying within Cl(p). If this point lies in In(p), then we continue with this point. If not, we apply part (iii) of the previous lemma to find such a point. If the search fails, then we return p as the approximate nearest neighbor. Let us describe the search in greater detail. Let Nc[p] denote the set of neighbors of p in cone c, let N[p] be the set of all p's neighbors, and let NCones denote the total number of cones centered at a point. We index the cones centered at a point from 1 to NCones. Let lowp(B) denote the point with lowest index relative to point p in a set of points B. The while-loop computes the pseudo-linear path described in part (iii) of the previous lemma. function NN (p, q) { Let r := lowp(N[p] ∩ Cl(p)); if (r ∈ In(p)) return(NN (r, q)); for c := 1 to NCones do { r := p; while (Nc[r] ∩ (Im(p) ∪ In(p)) 6= ∅) do { if (Nc[r] ∩ In(p) 6= ∅) { s := any point in Nc[r] ∩ In(p); return(NN (s, q)); } else r := lowr(Nc[r] ∩ Im(p)); } } return(p); }Observe that all the set operations can be performed in O(log n) time by enumerating the elements of either N[p] or Nc[r] and applying the appropriate membership tests for Cl(p), Pr (p), In(p) or Im(p). To verify the correctness of the above procedure, observe that if p is not a (1 +)-nearest neighbor, then Lemma 2.2 implies that there is a pseudo-linear path to a point which is strictly closer to q than p, and hence the search will succeed in finding such a point. To establish the running time of the search procedure we show that the number of recursive calls made to function NN is O(log n) in the expected case. As mentioned before the expectation is computed over all possible choices of random permutations made in the construction of NGd(δ, S), and hence is independent of S, and q. Our basic assertion is that with each successive call to NN , with fixed probability, the number of points that are closer than the current point to q decreases by a constant factor. Informally the argument is based on two cases, |Pr (p) ∪ Im(p)| > |In(p)| and |Pr (p) ∪ Im(p)|≤|In(p)|. In the former case, after O(log2 n) expected effort we either terminate, or make a new recursive call with a remaining set of points of size at most |In(p)| ≤ |Pr (p) ∪ Im(p)| + |In(p)| 2 ≤ |Cl(p)| 2 , and hence at least half of the points have been eliminated from further consideration. In the latter case, with probability at least 1/2, the point of lowest index (with respect to p) in Cl(p) is in In(p), and hence cannot be pruned. In this case, using an argument similar to the one used for the simple randomized search, it follows that we expect at least half of the points of In(p) to be eliminated from consideration by the point of lowest index (along with every point in (Pr (p) ∪ Im(p)) implying that at least half of the points are expected to be eliminated. Summarizing, in the first case we eliminate at least half the points after O(log2 n) effort, and in the second case we eliminate half the points in one step with probability at least 1/2. In the second case the (proportional to the number of neighbors of p). Lemma 2.3. The expected number of recursive calls to NN is O(log n), and hence the expected running time of the search procedure is O(log3 n).

3. A Practical Variant

Although the results of the previous section are theoretically appealing, for practical instances of the nearest neighbor search problem, the algorithm as presented will not be competitive with other practical approaches to the problem. The reason is that as a function of , the number of cones grows asymptotically as **Ω**(1/d−1) (for sufficiently small). In this section we describe a variant of the neighborhood graph method designed to perform well for realistic instances of the problem, even though the formal complexity and performance bounds shown in the previous section are not guaranteed to hold. We feel that these results suggest that this approach holds promise as a practical approach to nearest neighbor searching in higher dimensions. The proposed variant consists of the following principal modifications to the randomized neighborhood graph scheme introduced in the previous section. • To reduce the degree of the graph, we use a pruning scheme similar to the one used in the relative neighborhood graph [13], [21]. As we shall see the resulting graph is significantly

sparser. • To further reduce the degree of the graph by an O(log n) factor we abandon the randomized "skiplist" construction. Our empirical experience suggests that the "long" edges introduced by this part of the construction can be simulated cheaply by simply choosing a better starting point. This can be done by constructing a k-d tree for the point set (as part of the preprocessing), and choosing the starting point from the leaf node of the tree containing the query point. • Given the increased sparseness of the resulting graph, it is not reasonable to assume that the points along the search path will decrease monotonically in distance to the query point. We maintain the list of candidate points consisting of the neighbors of all the visited points. Repeatedly among the unvisited candidates, we select the closest to the query point. The resulting search path may not be monotone, but always attempts to move closer to the query point without repeating points. In summary we decrease the degree of the neighborhood graph, but at an additional cost to the number of steps needed in the search. It is not hard to devise worst case scenarios where this scheme will perform quite poorly. However in most practical situations the search quickly converges to the nearest neighbor. Let us begin by describing the revised neighborhood graph. It is quite similar to the relative neighborhood graph [13, 21]. The relative neighborhood graph of a set of points $S \subset Ed$ is an undirected graph in which two points p and q are adjacent if there is no point that is simultaneously closer to both points. Our modified neighborhood graph is a directed graph based on the following pruning rule. For each point $p \in S$, we consider the remaining points of S in increasing order of distance from p. We remove the closest point r from this sequence, create a directed edge from p to r, and remove from further consideration all points s such that dist(p, s) > dist(r, s). Intuitively, since r is closer to both p and s than they are to one another, the points s are not considered neighbors of p. This process is repeated until all points are pruned. This graph is equivalent to a graph presented by Jaromczyk and Kowaluk [11], which was used as an intermediate result in their construction of the relative neighborhood graph. We call this the sparse neighborhood graph for S, denoted RNG*(S). RNG*(S) can be computed easily in O(n2) time, where n = |S|. An important property of this graph, which explains its intuitive appeal for nearest neighbor searching, is that if the query point happens to be equal to a point of S, then a simple greedy search (at each step visiting any neighbor closer to the query point) will succeed in locating the query point along a path of monotonically decreasing distance to the query point. The reason is that if there is no edge between the current point and the query point, there must be a closer point to which there is an edge that has pruned the query point. Finding an upper bound on the degree of RNG* is closely related to the classical mathematical problem of determining the densest packings of spherical caps on the surface of the d-dimensional sphere. Define the diameter of a spherical cap to be the maximum angle between any two points on the cap. Lemma 3.1. Given a set of points S in Ed in general position (no two points are equidistant from a third point), the degree of any vertex in RNG*(S) does not exceed the number of spherical caps of diameter $\pi/3$ in a densest packing of the surface of the d-dimensional sphere. Proof. Let p, r, $s \in S$ be three points such that 6 $prs \leq \pi/3$. We claim that both p and s cannot be neighbors of r in RNG*(S), since, using elementary geometry, we can easily show that adding a directed edge from r to the closer of the two points p and would prune the point farther away. Thus, if we centrally project the set of neighbors of r onto a unit d-dimensional sphere centered at r and surround each neighbor with a spherical cap of radius $\pi/6$, it follows that no two of these caps can intersect, and hence they form a packing of the surface of the d-dimensional sphere. ut Unfortunately tight bounds for this quantity are not known for arbitrary dimension. It follows from Kabatjanskiĭi and Levenšteĭin's [12] upper bound on spherical packings and Shannon's [17] and Wyner's [22] lower bounds that as dimension goes to infinity the upper bound on the degree of any vertex p of the RNG*(S) lies in the interval [1.15d−1, 1.32d−1] (as $d \rightarrow \infty$). Unfortunately, these bounds are asymptotic in d, and it appears that for the relatively small values of d that we are interested in, these bounds are rather optimistic. For instance, in dimension 24, the worst case degree can be as large as 196,560 [20], while 1.3223 is only 593. However we conjecture that the expected degrees are much smaller than the worst case. To establish a practical bound on the expected degree of vertices in the RNG*(S) we performed two empirical studies. One study measured the expected degree of a vertex of the graph in dimension d on point sets of size 2d, uniformly distributed within a

unit cube. With such small point sets, boundary effects (the phenomenon in high dimensions that more points tend to be near the convex hull) are quite significant in artificially decreasing the degree of the graph. We ran a second experiment, which attempted to extrapolate this to point sets so large that boundary effects are negligible. In the first experiment we generated 2d uniformly distributed points and computed the degree of a random point. In the second experiment 100,000 points were uniformly distributed inside the hypercube and the degree of a vertex in the center was computed. In both cases, in each dimension the degree was averaged over 100 such trials. The results are presented in Fig. 4. By fitting lines to the logarithm of degrees we conjecture that for the first experiment the degree is 1.46(1.20d) and for the second experiment the degree is 2.9(1.24d) (and a study of residuals suggests the growth rate may be even slower). Even though this is exponential in dimension, it is acceptably small for dimensions in our range of interest. We search the graph using a best-first strategy. The search algorithm begins with a point p selected by choosing a point from a bucket of a k-d tree that contains the query point. We maintain a set of candidates to the nearest neighbor (maintained using a heap) initially containing p. We select the nearest of the candidates that has not already been visited. The algorithm is outlined below. function NN 2(p, q) { C := {p}; nn := p; while (C 6= ∅ and termination condition not yet met) { p := the point of C minimizing dist(q, p); C := C − {p}; for each undiscovered r in N[p] { Mark r discovered; C := C + {r}; if (dist(q, r) < dist(q, nn)) nn := r; } } return(nn); } The choice of termination conditions is somewhat subtle. Since the data structure lacks the structural information provided by other algorithms, it cannot know when it has found the nearest neighbor. In practice termination would be based on a convergence test. For this study we wanted to test the viability of this approach against other practical algorithms, such as the kd tree [9], which was refined and analyzed empirically by Sproull [19]1, and a simple bucketing algorithm, which was analyzed for uniform distributed data by Cleary [7] and independently by Bentley, Weide and Yao [4]. Because the algorithm based on the RNG*(S) does not guarantee finding the nearest neighbor (until all points have been enumerated), we chose as a basis for comparison the number of points considered by each algorithm until coming upon the nearest neighbor (which was precomputed off-line). Note that both the k-d tree algorithm and bucketing algorithm continue to search until establishing firmly that this is the nearest neighbor, but the time until first discovering the nearest neighbor certainly provides a lower bound on their execution times. The point distributions used in the experiments are described below. Some of these were presented by Bentley [3]. Uniform: Each coordinate was chosen uniformly from the interval [0, 1]. Normal: Each coordinate was chosen from the normal distribution with zero mean and unit variance. ClusNorm: Ten points were chosen from the uniform distribution and a normal distribution with standard deviation 0.05 put at each. Laplace: Each coordinate was chosen from the Laplacian distribution with zero mean and unit variance. To model the types of distributions seen in speech processing applications, two more point distributions were formed by grouping the output of autoregressive sources into vectors of length d. An autoregressive source uses the following recurrence to generate successive outputs: Xn = ρXn−1 + Wn where Wn is a sequence of zero mean independent, identically distributed random variables. The correlation coefficient ρ was taken as 0.9 for our experiments. Each point was generated by selecting the first component from the corresponding uncorrelated distribution (either normal or Laplacian) and the remaining components were generated by the equation above. Further details on how to generate these autoregressive processes may be found in Farvardin and Modestino [8]. Co-Normal: Wn was chosen so that the marginal density of Xn is normal with variance unity. Co-Laplace: Wn was chosen so that the marginal density of Xn is Laplacian with variance unity. Speech: From a database consisting of 6.8 million samples formed by sampling speech waveform at 8 kb/s, the consecutive samples were packed in groups to yield vectors in the required dimension. In 16 dimensions, we get 425,000 vectors, from which we choose vectors randomly from the first 400,000 vectors to form the set of data vectors and choose query vectors randomly from the remaining 25,000 vectors. To avoid cluttering the figures, we do not present the results for the ClusNorm and Co-Normal distribution; suffice it is to note that the results for these distributions were quite similar to the Co-Laplace distribution. Figure 5 shows the average number of points examined by the k-d tree algorithm until

termination, for a representative set of these distributions of points and over 1000 query points. For all our experiments, we constructed optimized k-d trees [9] in E16. The cut planes were placed at the median, orthogonal to the coordinate axis with maximum spread. Each leaf node contained one point, which is known to lead to the best performance of the k-d tree algorithm measured in terms of number of points examined until termination. In each case the data and query points are chosen from the same distribution. Table 1 shows average and maximum numbers of points and cells examined until termination by the bucketing algorithm for the case of points uniformly distributed in a 16 dimensional hypercube. The hypercube was partitioned into 216 equal-sized cells which were examined in increasing order of distance from the query point. We restricted our experiments with this technique to the uniform distribution because it is not easy to extend it to unbounded distributions. For 100,000 points, the results are similar to that of the k-d tree. Because our algorithm does not have a termination condition, it is not really fair to compare it against these algorithms which are required to continue searching until they have been assured that the nearest neighbor has been found. For this reason we focused on the question of how many points will be visited until the algorithm first comes across the nearest neighbor (even though the algorithm may not know that it has done so). We computed the true nearest neighbor off-line by brute force. Figure 6 and Table 2 show the number of points examined by the k-d tree and bucketing algorithms until finding the nearest neighbor, for the same set of data and query points for which the results are shown in Figure 5 and Table 1 respectively. For both the k-d tree.

4. CONCLUSION

We have presented a randomized algorithm for computing approximate nearest neighbors in expected polylogarithmic query time and O(n log n) space. Because the constants involved in this algorithm are quite large, we have also presented a more practical variant. Experimental evidence indicates this algorithm is quite effi- cient for many input distributions and on actual speech data in dimensions as high as 16. There are a number of interesting open problems suggested by this work. The most important theoretical question is that of removing the extra logarithmic factors from the space and running time, with the goal of providing O(log n) query time and O(n) space. It would also be nice to know if the results can be made deterministic. Another question is whether our search strategy could be replaced with a simpler greedy search and still guarantee polylogarithmic search time. The most important question from a practical standpoint is whether the constants (depending on d and) involved in the randomized algorithm can be reduced, or whether the efficiency of the RNG∗ search can be established theoretically.

REFERENCES:

[1] S. Arya and D. M. Mount. Algorithms for fast vector quantization. In J. A. Storer and M. Cohn, editors, Proc. of DCC '93: Data Compression Conference, pages 381–390. IEEE Press, 1993.

[2] C.-D. Bei and R. M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. IEEE Transactions on Communications, 33(10):1132–1133, October 1985.

[3] J. L. Bentley. K-d trees for semidynamic point sets. In Proc. 6th Ann. ACM Sympos. Comput. Geom., pages 187–197, 1990.

[4] J. L. Bentley, B. W. Weide, and A. C. Yao. Optimal expected-time algorithms for closest point problems. ACM Transactions on Mathematical Software, 6(4):563–580, 1980.

[5] D. Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham. Fast search algorithms for vector quantization and pattern matching. In Proceedings IEEE ICASSP, volume 1, pages 9.11.1–9.11.4, March 1984.

[6] K. L. Clarkson. A randomized algorithm for closest-point queries. SIAM Journal on Computing, 17(4):830–847, 1988.

[7] J. G. Cleary. Analysis of an algorithm for finding nearest neighbors in euclidean space. ACM Transactions on Mathematical Software, 5(2):183–192, June 1979.

[8] N. Farvardin and J. W. Modestino. Rate-distortion performance of DPCM schemes for autoregressive sources. IEEE Transactions on Information Theory, 31(3):402–418, May 1985.

[9] J. H. Friedman, J. L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, 3(3):209–226, September 1977.

[10] A. Gersho and V. Cuperman. Vector quantization: a pattern-matching technique for speech coding. IEEE Communication Magazine, 21(9):15–21, December 1983.

[11] J. W. Jaromczyk and M. Kowaluk. A note on relative neighborhood graphs. In Proc. 3rd Ann. ACM Sympos. Comput. Geom., pages 233–241, 1987.

[12] G. A. Kabatjanskiĭi and V. I. Levenˇsteˇin. Bounds for packings on the sphere and in space (russian). Problemy Peredaci Informacii, 14:3–25, 1978. Also, Problems of Information Transmission, 1–17.

[13] P. M. Lankford. Regionalization: theory and alternative algorithms. Geographical Analysis, 1(2):196–212, April 1969.

[14] J. Matouˇsek. Reporting points in halfspaces. In Proc. 32nd Ann. Sympos. Foundations of Computer Science, pages 207–215, 1991.

[15] F. P. Preparata and M. I. Shamos. Computational Geometry. Springer-Verlag, 1985.

[16] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. Communications of the ACM, 33(6):668– 676, June 1990.

[17] C. E. Shannon. Probability of error for optimal codes in a gaussian channel. Bell System Technical Journal, 38(3):611–656, 1959.

[18] M. R. Soleymani and S. D. Morgera. An efficient nearest neighbor search method. IEEE Transactions on Communications, 35(6):677–679, June 1987.

[19] R. L. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica, 6, 1991. [20] G. Fejes T´oth. New results in the theory of packing and covering. In Convexity and its applications, pages 318–359. Birkh¨auser Verlag, Basel, 1983.

CONTRIBUTORS: