Research Promotion Workshop
on
Introduction to Graph and Geometric Algorithms

Thapar University, Patiala
October 28-30, 2010

Coordinators

- Prof. Deepak Garg (Co-Convener), Department of Computer and Engineering, Thapar University, Patiala 147004.
- Prof. Subir Kumar Ghosh (Co-Convener), School of Technology & Computer Science, Tata Institute of Fundamental Research, Mumbai 400005.
- Prof. Sandeep Sen, Department of Computer Science & Engineering, Indian Institute of Technology, New Delhi 110 016.
- Prof. Subhas Chandra Nandy, Advanced Computing & Microelectronic Unit, Indian Statistical Institute, Kolkata 700108.

Preface

The study of algorithms is at the very heart of computer science. In the last five decades, a number of significant advances have been made in the field of algorithms. These advances have ranged from the development of faster algorithms to the startling discovery of certain natural problems for which all known algorithms are inefficient. These results kindled considerable interest in the study of algorithms, and the area of algorithm design and analysis has blossomed into a field of interest. Teaching and research in this foundational aspect of computing is therefore a natural and desirable thrust area.

Algorithms are also at the heart of every nontrivial computer application. Therefore, computer scientists and professional programmers are expected to know about the basic algorithmic toolbox: structures that allow efficient organization and retrieval of data, frequently used algorithms and generic techniques for modeling, understanding and solving algorithmic problems. Hence, algorithmic studies form a major component of computer science programs in colleges and universities.

In the last four decades, graph and geometric problems have been studied by computer science researchers using the framework of design and analysis of algorithms. Graph theory is the study of the properties of graphs. Graph algorithms are one of the oldest classes of algorithms and they have been studied for almost 300 years. Graphs provide essential models for many application areas of computer science, and at the same time, they are fascinating objects of study in pure and applied mathematics. There have been a number of exciting developments in graph theory that are important for designers of algorithms. Correspondingly, the algorithmic viewpoint of computer science has stimulated much research in graph theory. Graph theory and graph algorithms are inseparably intertwined subjects.

On the other hand, the main impetus for the development of geometric algorithms came from the progress in computer graphics, computer-aided design and manufacturing. In addition, algorithms are also designed for geometric problems that are classical in nature. The success of the field can be explained from the beauty of the geometry problems studied, the solutions obtained, and by the many application domains – computer graphics, geographic information systems, robotics and others, in which geometric algorithms play a crucial role.

In the last two decades, the number of researchers working in India on graph and geometric algorithms has increased substantially. Presently, research is being carried out in India in several sub-areas of algorithms and results are being published in reputed conferences and journals in computer science. However, the number of active researchers in algorithms is still far less in India compared to the large number of bright students and teachers involved in studying and teaching computer science in India. In order to motivate them towards computer science research in general and algorithmic research in particular, a series of workshops on *'Introduction to graph and geometric algorithms'* for teachers and students (undergraduate, post-graduate and doctoral) is being organized at engineering colleges and universities at different locations in India. The current workshop belongs to this series of research promotion workshops. This program may be viewed as a human resource development program for raising the level of algorithmic knowledge among Indian university students and faculty in computer science and Mathematics.

The workshop will be held at C-Hall, Thapar University, Patiala during October 28-30, 2010. In the workshop, we have fourteen distinguished speakers and around 150 registered participants. The speakers are key figures in their respective research areas and each of them will give an introductory lecture for an hour on a front-line research topic on algorithms and their applications with an idea to expose participants to various key developments in the field. An outline of each lecture with necessary references are included here for the benefit of participants.

The workshop is jointly organized by the Department of Computer Science and Engineering, Thapar University, Patiala, and the School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai. The workshop is funded by the National Board of Higher Mathematics, Department of Atomic Energy, Government of India.

Deepak Garg
Thapar University, Patiala

Subir Kumar Ghosh
TIFR, Mumbai

October 10, 2010

Lectures

Introduction to Computational Geometry

Arijit Bishnu

Advanced Computing and Microelectronics Unit
Indian Statistical Institute, Kolkata 700108
arijit@isical.ac.in

The subject of *Computational Geometry* deals with the study of algorithms for solving geometric problems on a computer. There are several studies in geometry but the term *Computational Geometry* now refers to discrete and combinatorial geometry. This branch of study is around thirty years old if one assumes Michael Ian Shamos's thesis [1] as the starting point. The study of computational geometry as a research discipline owes its success primarily to two issues - on the one hand we have the beauty of theoretical studies in terms of the problems studied and the solutions obtained, on the other hand we have the application domains like robotics, computer graphics, geographic information systems, etc. in which geometric algorithms play a pivotal role. The rapid strides made by the subject is reflected in the number of text books available [2, 3, 4, 5].

In this introductory talk on computational geometry, we will focus on some basic geometric algorithms. We will see that better mathematical characterizations of the geometric problems essentially leads to better algorithms.

The problem of sorting or searching has as its input a set of real numbers. Likewise, problems in Computational Geometry have as input, sets of geometric primitives like points, lines, line segments, polygons, circles, etc. The algorithmic paradigms like divide-and-conquer, induction, dynamic programming, greedy are also applicable in Computational Geometry. An algorithmic paradigm that is unique to Computational Geometry is *plane sweep*. This paradigm talks of sweeping across the plane an infinite horizontal or vertical line. This is easier said than done on a digital computer because plane sweep essentially talks of a phenomenon on the continuous plane. The crux of making plane sweep work is to find certain points or regions in the plane where there are certain combinatorial changes in the geometric structure. This paradigm of plane sweep will be covered in some detail using the problems of *convex hull* and *line segment intersection*.

Apart from these two problems, we will look at some very basic geometric algorithms like *point inclusion in a simple polygon*, *area computation of a simple polygon*. The talk will end by looking at the famous *art gallery theorem* which states the following.

Theorem 1 *For a simple polygon with n vertices, $\lfloor \frac{n}{3} \rfloor$ cameras are occasionally necessary and always sufficient to have every point in the polygon visible from at least one of the cameras.*

References

[1] Michael Ian Shamos, *Computational Geometry*, PhD thesis, Yale University, New Haven.

[2] Franco P. Preparata and Michael Ian Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[3] Joseph O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1998.

[4] Mark de Berg, Marc van Kreveld, Mark Overmars and Otfried Schwarzkof, *Computational Geometry: Algorithms and Applications*, Springer, 1997.

[5] Herbert Edelsbrunner, *Algorithms in Computational Geometry*, Springer, 1987.

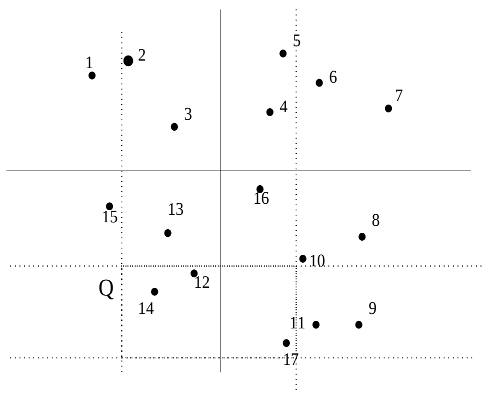
Geometric Data Structures

Swami Sarvottamananda

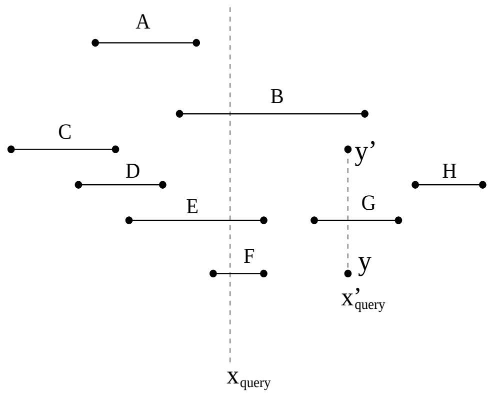
School of Mathematical Sciences
Ramakrishna Mission Vivekananda University
Belur, Howrah, West Bengal 711202
`shreesh@rkmvu.ac.in`

We present a few important data structures used in efficient algorithms for fundamental geometric problems arising in several applications. We concentrate on a few important and fundamental techniques that appear in several geometric searching problems. The common feature in these problems is the *output-sensitivity* of the time complexity required to retrieve data in *queries*; once the entire data is *preprocessed*, the time complexity of each query is partly proportional to the size of the reported output and partly proportional to a sub-linear function of the size of the database.

Orthogonal range queries: Kd-trees and range search trees. We begin with *orthogonal range queries* or *rectangular queries*. Given a set S of n points in the plane, report points inside a query rectangle Q whose sides are parallel to the axes. Such queries are common in databases, computer graphics, and in VLSI design. Here, a set S of n points is given apriori. We *preprocess* S to create some data structure, so that arbitrary 2-dimensional (2-d) *range queries* can be answered (efficiently) subsequently. The query processing for each *query rectangle* $Q[a, b] \times [c, d]$, is expected to scan only a subset of S, necessarily the points that appear inside the query rectangle Q. We may use Kd-trees for 2-dimensional range searching (see [1]). In Figure (a), the points inside Q are 14, 12 and 17. We construct a Kd-tree in $O(n \log n)$ time using $O(n)$ space, so that orthogonal range queries may be answered in $O(\sqrt{n} + k)$ time (see [1]). Here, k is the number of points of S inside the query rectangle Q.



(a)



(b)

Using two 1-d *range queries*, one query along each axis, one can answer 2-d range queries by selecting points common to the outputs of the two queries. The cost thus incurred may however exceed the actual (much smaller) output size of the 2-d *rectangular range query.* Truly *output sensitive* query time $O(\log^2 n + k)$ can be achieved if we use a 2-d *range tree* requiring $O(n \log n)$ space (see [1]). The query time can be further reduced to $O(\log n + k)$ by using the technique of *fractional cascading* for the *range tree* data structure [1], keeping the space requirement within $O(n \log n)$.

The windowing problem and the interval tree data structure. The window *clipping problem* in computer graphics requires displaying those parts of line segments that lie inside an arbitrary window. The set of line segments need to be preprocessed, so that arbitrary *windowing queries* can be answered by reporting portions of such clipped segments. We can use 2-d rectangular range queries as in the previous section, for reporting segments with at least one endpoint inside the query window rectangle. However, segments that cut across the window (with their endpoints outside the window), cannot be detected in this manner. For such segments we need to solve the following problem: given a set S of n horizontal intervals, we need to report those which intersect the vertical segment joining points (x'_{query}, y_{min}) and (x'_{query}, y_{max}) (see Figure (b)). A simpler problem is to consider the infinite vertical line at abscissa x_{query} (Figure (b)). We discuss the solution to these problem using the data structure called *interval trees* as in [1]. The preprocessing time required is $O(n \log n)$, and the query time is $O(\log n + k)$.

Plane sweep and angular sweep algorithms. The plane sweep technique helps in reporting all intersections between n given segments in the plane. The main idea is to sweep a vertical line from left to right, taking steps at endpoints of each of the n segments; the algorithm discovers all intersections and takes a step also at each intersection point. Some actions are taken to maintain the data structures at each step so that the reporting of all intersections to the left of the sweep line is ensured. Using simple data structures, all the k intersection can be reported in $O((n + k) \log n)$ time [3].

The sweep paradigm is useful also in computing the region visible from a point inside a polygon with holes. Typically, the region visible to a robot inside a non-convex room with several obstacles, can be modeled as the visible region from a point inside a polygon with holes. If there are n edges defining the boundary of the polygon and h triangular holes/obstacles, then the visible region can be computed in $O(n \log h)$ time [2].

References

[1] Mark de Berg, Otfried Schwarzkopf, Marc van Kreveld and Mark Overmars, Computational Geometry: Algorithms and Applications, Springer.

[2] S. K. Ghosh, Visibility Algorithms in the Plane, Cambridge University Press, Cambridge, UK, 2007.

[3] F. P. Preparata and M. I. Shamos, Computational Geometry: An Introduction, New York, NY, Springer-Verlag, 1985.

Introduction to Approximation Algorithms

Deepak Garg

Computer Science and Engineering Department
Thapar University
Patiala - 147004
dgarg@thapar.edu

Problems that lie in the set of NP-complete are very important having applications in every area of life. These problems cannot be left unattended only because the optimal solution algorithm takes exponential complexity. Most of the time near perfect solution are equally desirable and have similar usefulness as the actual solution. These near perfect solution which can be achieved in polynomial time are called approximation algorithms.

An approximation scheme for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\epsilon > 0$ such that for any fixed ϵ, the scheme is a $(1 + \epsilon)$-approximation algorithm. We say that an approximation scheme is a polynomial-time approximation scheme if for any fixed $\epsilon > 0$, the scheme runs in time polynomial in the size n of its input instance.

Vertex Cover Problem is NP-complete problem. A vertex cover of an undirected graph $G = (V, E)$ is a subset V such that if (u, v) is an edge of G, then either $u \in V$ or $v \in V$ (or both). The size of a vertex cover is the number of vertices in it. The vertex-cover problem is to find a vertex cover of minimum size in a given undirected graph. Such a vertex cover is called an optimal vertex cover.

Travelling Salesman Problem is also a NP-hard problem. Given a list of cities and their pair wise distances, the task is to find a shortest possible tour that visits each city exactly once. we are given a complete undirected graph $G = (V, E)$ that has a non-negative integer cost $c(u, v)$ associated with each edge $(u, v) \in E$, and we must find a Hamiltonian cycle (a tour) of G with minimum cost.

In this introductory lecture, we present a few constant factor approximation algorithms for some NP-hard problems.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C, Stein, *Introduction to Algorithms*, Third Edition, MIT Press, 2007.

[2] D. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, PWS Publishing, Boston, 1997.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co, 1979.

[4] V. Vazirani, *Approximation Algorithms*, Springer, 2001.

Approximation Algorithms and Linear Programming

Daya Gaur

Department of Computer Science and Engineering
Indian Institute of Technology Ropar
Rupnagar - 140001
`http://www.iitrpr.ac.in/html/faculty/daya.shtml` `mail:daya@iitrpr.ac.in`

Approximation algorithms trade accuracy for time and are a natural and elegant way of coping with $\mathcal{NP}$-completeness. For a minimization optimization problem, a polynomial time algorithm is called an approximation algorithm if for every instance of the problem the algorithm returns a solution with value within a constant multiplicative (or additive) factor of the value of the optimal solution to the instance. Two equally important steps in designing an approximation algorithm are i) a polynomial time algorithm, ii) construction of tight lower bounds on the optimal solution. Linear programming provides a general method for performing both the tasks. Most of the state of art approximation algorithms follow this design.

Linear programming is the problem of minimizing (maximizing) a linear objective function subject to linear constraints (see Vanderbei). Linear programming has found wide spread use in optimization since the invention of the Simplex algorithm by Dantzig. The fundamental theorem of linear programming states that the values for the optimal primal and dual programs are the same (if both are bounded).

By formulating an $\mathcal{NP}$-complete minimization problem as an integer linear program and relaxing the integrality constraints, we get a lower bound on the value of the optimal solution (weak duality). Given a minimization integer program, the ratio of the value of the optimal integral solution to the value of the optimal fractional solution is referred to as the *integrality gap*. Based on the work of Kuhn, [7], Dantzig, Ford and Fulkerson [3] gave a primal-dual algorithm for the linear programming. The primal-dual algorithm for linear programming has been widely used in combinatorial optimization and has led to discovery of the some of the corner stone algorithms in $\mathcal{P}$. A slight modification to the primal-dual algorithm so as to update the primal solutions only integrally gives us an algorithm too. Hence linear programming provides us with a general method (albeit not the only one) for constructing approximation algorithms. However there is an important caveat, if the optimal is bounded by the linear programming relaxation only, then the approximation ratio is at least the integrality gap. Finding integer linear programs with small integrality gap is an art. Two excellent books on approximation algorithms are by Vazirani [9] and Hochbaum [6]. A comprehensive collection of results is in the recent Handbook of Approximation Algorithms and Metaheuristics [4].

In this talk, aimed at senior undergraduate and graduate students, we will look at two methods for designing approximation algorithms using linear programming: rounding, and the primal-dual schema. We will illustrate these techniques on various problems [1, 5]. For the use of linear programs in the computation of lower bounds on the optimal, we will consider the general method of dual fitting for analyzing greedy approximation algorithms. We will also consider two general methods for reducing the integrality gap; parametric pruning and addition of valid inequalities. For case studies in the reduction of the integrality gap, we will look at parametric pruning on the multiprocessor scheduling problem [8], and the addition of flow-cover inequalities for the minimum knapsack problem [2].

References

[1] R. Bar-Yehuda and S. Even, *A linear-time approximation algorithm for the weighted cover problem.* Journal of Algorithms, 2:198–203, 1981.

[2] T. Carnes and D. B. Shmoys, *Primal-dual schema for capacitated covering problems.* In IPCO, pages 288–302, 2008.

[3] G. B. Dantzig, L. R. Ford, D. R. Fulkerson, *A primal-dual algorithm for linear programs*, In Linear Inequalities and Related Systems, Editors Kuhn and Tucker, pages 171–181, Princeton University Press, 1956.

[4] T. Gonzalez, editor, *Handbook of Approximation Algorithms and Meta Heuristics*, CRC Press, 2007.

[5] D. R. Gaur and T. Ibaraki and R. Krishnamurti, *Constant Ratio Approximation Algorithms for the Rectangle Stabbing Problem and the Rectilinear Partitioning Problem*, Journal of Algorithms 43(1): 138–152, 2002.

[6] D. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, PWS Publishing, Boston, 1997.

[7] H. W. Kuhn, *The Hungarian method for the assignment problem*, Naval Research Logistics Quaterly, 2:83–97, 1955.

[8] J. K. Lenstra, D. B. Shmoys and E. Tardos, *Approximation Algorithms for scheduling unrelated parallel machines*, Math. Prog., 46:259–271, 1990.

[9] V. Vazirani, *Approximation Algorithms*, Springer, 2001.

Introduction to Randomized Algorithms

Subhas C. Nandy

Computing and Micro-Electronics Unit
Indian Statistical Institute, Kolkata 700108.
nandysc@isical.ac.in

The traditional approach for solving a computational problem using the computers is to devise a deterministic approach for solving that problem. But, often it is observed that several problems, in its most practical framework, are sometimes NP-hard, or even if it is polynomial time solvable, the degree of the polynomial in the running time complexity is very high. So, the last two decades have observed a tremendous growth in the area of randomized algorithms. Randomized algorithms find widespread application in many types of algorithms. Two benefits of randomization have spearheaded this growth: simplicity and speed. Naturally, this introduces uncertainty in the correctness and/or efficiency of the algorithm. In spite of this difficulty, the randomized algorithms have become a powerful tool for solving several computational problems.

Needless to say, in a deterministic algorithm, the output and the time of execution become fixed as soon as the input is fixed. But, in a randomized algorithm, the output and/or execution time may vary for the same input if it is executed many times. Here, each execution step is chosen from a selected set of steps in a random manner, guided by pseudo random numbers.

Usually, one uses randomized algorithm where, in addition to the simplicity, one can mathematically show that either small running time or correctness or both can be guaranteed with reasonably high probability. Thus, even though the algorithm may be slow or incorrect, the chances of occurring either of this is very low. If we are ready to accept this low amount of uncertainty, then randomization is really a powerful tool for solving computationally difficult problems.

In this context, we should mention that, randomized algorithms are not the probabilistic analysis of the running time of a deterministic algorithm.

This lecture will present the basic concepts in the design and analysis of randomized algorithms at a level accessible to advanced undergraduates and to graduate students. The aim is to touch upon various branches of the study of randomized algorithms. We will discuss on

- some familiarity with several of the main thrusts of work in randomized algorithms that give an ability to formulate a new algorithmic problem using the known results;
- designing a set of randomized algorithmic tools for some useful algorithmic problems that one often encounters in practical domain.

We will also give some amount of emphasize on some practical problems chosen from the area of graph theory and computational geometry.

References

[1] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[2] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer, 2008.

Introduction to Online Scheduling

Naveen Sivadasan

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad 502205
nsivadasan@iith.ac.in

In this talk, we consider a basic problem of scheduling a sequence of n jobs $j_1, j_2, \ldots, j_n$ arriving in an online fashion to a set of m identical machines. Jobs can have different processing times and each job j_i has to be assigned to one of the m machines without the knowledge of the processing time of any future job j_k with $k > i$. The objective is to minimize the *makespan*, which is the completion time of the last finishing job.

This problem belongs to the framework of *online computation* where the input is a request sequence that is revealed to an algorithm in an online fashion. An online algorithm has to serve the requests in the same order by making decisions based on the requests seen so far without the knowledge of the future requests. The objective is to minimize the total cost incurred in serving the entire request sequence. Such a framework is useful in modeling scenarios where there is an inherent lack of information and the algorithm has to perform well under this uncertainty. Many real world problems such as routing in communication networks, scheduling of jobs to machines, paging in virtual memory etc., can be formulated in this framework.

To analyze the performance of online algorithms, Sleator and Tarjan proposed a comparison measure called *competitive analysis* where cost incurred by the online algorithm ALG is compared against the cost of an *optimal offline algorithm* OPT which knows the complete request sequence in advance. Let $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$ denote the request sequence and $\text{ALG}(\sigma)$ and $\text{OPT}(\sigma)$ denote the cost of serving σ by ALG and OPT respectively. The algorithm ALG is c-competitive if there exists a constant a such that $\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma) + a$ for all request sequences σ. Competitive ratio of ALG is the minimum c such that ALG is c-competitive. The achievable competitive ratios in online settings are not determined by the limited computing power but by the lack of information about the parts of the input that will only be revealed in the future.

We will look at various strategies and their competitiveness for the above scheduling problem. We will also look at the offline version where an optimal offline schedule has to be computed efficiently. We will conclude the talk with an overview of two other important online scheduling problems, namely the Metrical Task Systems and the k-server problem.

References

[1] Allan Borodin and Ran El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge Univ. Press, Cambridge, 2005.

[2] S. Albers, *Online algorithms: A survey*, Mathematical Programming, 97:3-26, 2003.

[3] J. Sgall, *On-line scheduling - A survey*, Online Algorithms: The State of the Art, LNCS. 1442, pages 196-231, Springer, 1998.

[4] D. S. Hochbaum and D. B. Shmoys, *Using dual approximation algorithms for scheduling problems: Theoretical and practical results*, Journal of the ACM, 34:144 –162, 1987.

Sketching Streams

Sumit Ganguly

Department of Computer Science and Engineering
Institute of Technology, Kanpur 208 016
sganguly@cse.iitk.ac.in

Many applications today are characterized by a flow of massive, continuous and rapidly arriving data and a need to perform online analysis on this data . Examples of such applications include network monitoring (e.g., monitoring denial of service attacks), sensor networks (looking for regions in critically w.r.t sensor readings), financial market data (for tracking trends), satellite data (tracking weather phenomena), graph data (monitoring community evolution), etc.. The *Data Streams Model* is a computational model that addresses many common concerns of such applications. In this model, data is viewed as a potentially infinite sequence of records that has to be processed in an online fashion, efficiently and using space that is sub-linear in the size of the stream (often, poly-logarithmic in the size of the stream). Approximation and randomization are used and often necessary, since, for many problems, there are provable linear space lower bounds otherwise.

Formally, we can define a stream as a sequence of records where each record is of the form (T, i, v). Here T is the index of the record (or its time stamp), i is the key or the item of the record and is an element from the domain $[n] = \{1, 2, \ldots, n\}$. The value v is designated as the *change in frequency* of i. Associated with a stream σ is an n-dimensional frequency vector $f = f_\sigma$ defined as $f_\sigma(i) = \sum_{(T,i,v)\in\sigma} v$. Corresponding to a new stream record (T, i, v), the frequency vector changes to $f_{\sigma\circ(T,i,v)} := f_\sigma + v \cdot e_i$, where, e_i is the ith elementary vector of $\mathbb{R}^n$.

A basic problem in this model is to compute the pth frequency moment, defined as $F_p = \sum_{i\in[n]} |f_i|^p$ or the L_p norm $\|f\|_p$, for $p > 0$. The moment estimation problem is the basis of "higher-order" sampling, for example, sampling item from the stream proportional to the pth power of its frequency $|f_i|^p$. For $p > 1$, it allows the detection of "heavy-hitters" that cannot be done using regular sampling. A deterministic algorithm that estimates F_p to within a factor of $1 \pm 1/16$ can be shown to require $\Omega(n)$ space, for every $p \geq 0$.

A randomized algorithm for estimating F_2 was presented by Alon, Matias and Szegedy [1] that required space $O(\epsilon^{-2}\log(1/\delta))$ and returned an estimate $\hat{F}_2$ satisfying $|\hat{F}_2 - F_2| \leq \epsilon F_2$ with probability $1 - \delta$. The method constructs a random linear sketch $X = \sum_{i\in[n]} f_i \xi_i$, where, ξ_i's takes random values 1 or -1 and are four-wise independent. Linear sketches can be maintained by processing an update (T, i, v) as $X := X + v\xi_i$. Due to four-wise independence, updates can be processed in $O(1)$ + and cot operations over a field of size $O(n)$ (e.g., computing a cubic polynomial at i). A direct calculation shows that $\mathbb{E}\left[X^2\right] = F_2$ and $\mathsf{Var}\left[X^2\right] = 2F_2^2$. The AMS algorithm keeps $t = O(1/\epsilon^2)$ independent X's, $X_1, \ldots, X_t$ and obtains the average of their squares: $Y = (X_1^2 + \ldots + X_t^2)/t$. This transformation preserves the expectation and reduces the variance by a factor of t, so that $\mathbb{E}\left[Y\right] = F_2$ and $\mathsf{Var}\left[Y\right] = \epsilon^2 F_2^2$. An application of Chebychev's inequality yields that $\mathsf{Pr}\left\{|Y - F_2| < \epsilon F_2\right\} \geq 7/8$. A standard technique is now applied to boost the success probability from $7/8$ to $1-\delta$ by keeping $s = O(\log(1/\delta))$ independent groups of $t = O(1/\epsilon^2)$ sketches and returning the median of the averages of each group. The space requirement is $O(1/\epsilon^2 \log(1/\delta)$ sketches, where, each sketch is $O(\log(mn))$ bits, where, m is an assumed upper bound on $|f_i|$. An $\Omega(1/\epsilon^2)$ space bound for randomized algorithms was shown by Indyk and Woodruff [11] and Woodruff [15]. The time requirement appears to be $O(1)\times$ the number of sketches–

however, it can be reduced to $O(\log(1/\delta))$ using the fact that the sketches in each group need only be pair-wise independent. AMS sketches are fundamentally used in the design of nearly space optimal algorithms for estimating F_p for $p > 2$ [10, 4]. The space requirement of these algorithms is $\tilde{O}(n^{1-2/p})$. [1] A lower bound of $\Omega(n^{1-2/p})$ was established for estimating F_p, for $p > 2$, using a reduction [1] from the communication complexity of multi-party set-disjointness [2, 6].

Indyk [9] presented a solution to the problem of estimating F_p for $p \in [0, 2)$. A p-stable sketch is a linear sketch of the form $X = \sum_{i \in [n]} f_i s_i$, where,the s_i's are independent and are random variable with distribution $S(p, 1)$, the stable distribution with parameter p, scale factor 1 (and skewness 0). By property of p-stable distribution, $X \sim S(p, \|f\|_p)$. Keep $t = K_I/(\epsilon^2 p^2)$ independent p-stable sketches $X_1, \ldots, X_t$, where, K_I is an appropriate constant. Indyk [9] shows that there exists a constant C_I such that $C_I \cdot$ median$(|X_1|^p, \ldots, |X_t|^p)$ is within $(1 \pm \epsilon)F_p$ with probability 7/8. Li shows [13] estimator $C_L \cdot \left(|X_1|^p |X_2|^p \ldots |X_t|^p\right)^{1/t}$ for $t = K_L/(\epsilon^2 p^2)$ also returns an ϵ-approximation of F_p with probability 7/8 and is additionally unbiased. Indyk in [9] shows that Nisan's PRG for fooling space bounded computations can be used in a principled manner to allow these computations to be done in an online fashion and reduce the randomness and the time required.

The problem of estimating F_0 has received additional attention, since F_0 is the number of items in the data stream with non-zero frequency, that is, $F_0 = \sum_{i \in [n]} |f_i|^0 = \sum_{i: f_i \neq 0} 1 = \left|\{i \mid f_i \neq 0\}\right|$. We present the basic idea, due to Flajolet and Martin [8], for the case when item frequencies are non-negative, i.e., $f_i \geq 0$. Obtain a sample S from the set of elements with non-zero frequency with probability $1/T$ and keep $\text{Sum}_S = \sum_{i \in S} f_i$. Then, $\mathsf{Pr}\{\text{Sum}_S > 0\} = 1 - (1 - 1/T)^{F_0}$. Suppose we keep $t = O(1/\epsilon^2 \log(1/\delta))$ independent Sum_S's and count the fraction p_t of these counters that are non-zero. Using Chernoff's bounds and calculus arguments, it can be shown that, (i) if $F_0 \geq T/16$, then, p_t is $O(\epsilon)$-close to $\mathsf{Pr}\{\text{Sum}_S > 0\}$, and, (ii) if $F_0 \leq T/3$ and p_t is $O(\epsilon)$-close to $\mathsf{Pr}\{\text{Sum}_S > 0\}$, then, $\log(1 - p_t)/\log(1 - 1/T)$ is ϵ-close to F_0. One can now keep estimate F_0 by keeping t counters for each value of $T = 2, 2^2, 2^3, \ldots, 2^{\log n}$. The problem has been studied in [1, 3].

Random linear sketches can be viewed as a linear mapping of the frequency vector $x \to Ax$, where, A is a randomized $t \times n$ matrix, where, $m \ll n$. For example, the AMS linear sketch maps the n-dimensional frequency vector x to a t-dimensional vector where $t = O(1/\epsilon^2 \log(1/\delta))$. Such sketches are closely related to dimensionality reduction mappings, such as the Johnson-Lindenstrauss's (J-L) mapping [12]. The classical J-L map is constructed by choosing a random $t \times n$ matrix A whose entries are i.i.d $N(0, 1)$ random variables. If a_j is the jth row of A, then, $a_j x \sim N(0, \|x\|_2)$ and therefore $(a_j x)^2 \sim \text{Gamma}(1/2, 2\|x\|_2^2)$. Therefore, $\|Ax\|^2 = \sum_{j=1}^{t} (a_j x)^2 \sim \text{Gamma}(t/2, 2\|x\|_2^2)$. By probability concentration property for Gamma distribution, it follows that $\left|\|Ax\|^2/t - \|x\|^2\right| \leq \epsilon \|x\|^2$ with prob. $1 - \exp\{-t\epsilon^2/6\}$ or $\epsilon < 1$. If $t = O(1/\epsilon^2 \log m)$, then, for any set of m points, all pair-wise distances and norms are preserved to within a factor of $1 \pm \epsilon$ (also called ϵ-distortion) simultaneously with probability $1 - 1/m^2$. Indyk's application of Nisan's PRG [14] allows the J-L mapping to be applied over streams.

A recent development is that of compressive sensing. A vector is said to be k-sparse if it has at most k non-zero entries. We would like to make a linear measurement Ax of an n-dimensional signal x, where, A is an $m \times n$ matrix and $m \ll n$. From the measured signal Ax, one would like to recover the best k-sparse approximation x^* such that x^* agrees with x on the measurement (i.e., $Ax = Ax^*$). So ideally, we would like to solve the problem: Minimize $\|x^*\|_0$ subject to $Ax = Ax^*$. However, this is quite hard (NP-hard) to solve. So we would like to approximate the optimal k-sparse approximation by a vector x' such that the norm of the difference $x' - x$ is a small constant factor C times the best

[1]The $\tilde{O}$ notation hides polynomial factors of $1/\epsilon$ and poly-logarithmic factors of m and n.

k-sparse approximation to x, that is, $\|x - x'\|_p \leq C\text{Err}$, where, $\text{Err} = \text{Err}_k^p = \min_{\|x''\|_0 \leq k} \|x - x''\|_p$. So the approach [5, 7] is to solve the following problem: (LP) *Minimize* $\|x^*\|_1$ *subject to* $Ax = Ax^*$. If A satisfies certain properties then the solution to (LP) is an approximation to the problem of minimizing $\|x\|_0$. (LP) is a linear programming problem as shown by the following transformation and hence can be solved in time $n^{O(1)}$. *Minimize* $\sum_{i=1}^n t_i$ *subject to* $-t_i \leq x \leq t_i$ and $Ax = Ax^*$. The central property that A must satisfy is the (k, δ) Restricted Isometry Property (RIP). An $m \times n$ matrix is (k, δ) RIP if for any k-sparse vector x, $(1 - \delta)\|x\|_2 \leq \|Ax\|_2 \leq (1 + \delta)\|x\|_2$. It can be shown that if A is $O(k \log(n/k)) \times n$ matrix with *i.i.d.* $N(0, 1)$ random variables, then, A has the $(k, 1/3)$ RIP property. This property can be used to show that, with high probability, the optimal solution x^* to (LP) satisfies $\|x - x_0\|_2 \leq C\text{Err}_k^1(x)/\sqrt{k}$, *for all* $x \in \mathbb{R}^n$, Further, since the joint distribution of independent normal variates is spherical (i.e., invariant of rotation), it follows that this property holds for any orthonormal basis as well.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating frequency moments". *J. Comp. Sys. and Sc.*, 58(1):137–147, 1998.

[2] Z. Bar-Yossef, T.S. Jayram, R. Kumar, and D. Sivakumar. "An information statistics approach to data stream and communication complexity". In *Proceedings of ACM STOC*, pages 209–218, Princeton, NJ, 2002.

[3] Z. Bar-Yossef, T.S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. "Counting distinct elements in a data stream". In *Proceedings of International Workshop on Randomization and Computation (RANDOM)*, Cambridge, MA, 2002.

[4] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. "Simpler algorithm for estimating frequency moments of data streams". In *Proceedings of ACM Symposium on Discrete Algorithms (SODA)*, pages 708–713, 2006.

[5] Emmanuel Candès, Justin Romberg, and Terence Tao. "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information". *IEEE Trans. Inf. Theory*, 52(2):489–509, February 2006.

[6] A. Chakrabarti, S. Khot, and X. Sun. "Near-Optimal Lower Bounds on the Multi-Party Communication Complexity of Set Disjointness". In *Proceedings of International Conference on Computational Complexity (CCC)*, Aarhus, Denmark, 2003.

[7] David Donoho. "Compressed sensing". *IEEE Trans. on Information Theory*, 52(4):1289–1306, April 2006.

[8] P. Flajolet and G.N. Martin. "Probabilistic Counting Algorithms for Database Applications". *J. Comp. Sys. and Sc.*, 31(2):182–209, 1985.

[9] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006. Preliminary Version appeared in Proceedings of IEEE FOCS 2000, pages 189-197.

[10] Piotr Indyk and David Woodruff. "Optimal Approximations of the Frequency Moments". In *Proceedings of ACM Symposium on Theory of Computing STOC*, pages 202–298, Baltimore, Maryland, USA, June 2005.

[11] Piotr Indyk and David P. Woodruff. "Tight Lower Bounds for the Distinct Elements Problem". In *Proceedings of IEEE Foundations of Computer Science (FOCS)*, pages 283–292, 2003.

[12] William B. Johnson and Joram Lindenstrauss. "Extensions of Lipschitz mapping into Hilbert Space". *Contemporary Mathematics*, 26:189–206, 1984.

[13] Ping Li. Estimators and tail bounds for dimension reduction in ℓ_α ($0 < \alpha \leq 2$) using stable random projections. In *Proceedings of ACM Symposium on Discrete Algorithms (SODA)*, pages 10–19, 2008.

[14] N. Nisan. "Pseudo-Random Generators for Space Bounded Computation". In *Proceedings of ACM Symposium on Theory of Computing STOC*, pages 204–212, May 1990.

[15] David P. Woodruff. "Optimal space lower bounds for all frequency moments". In *Proceedings of ACM Symposium on Discrete Algorithms (SODA)*, pages 167–175, 2004.

Probability and Graphs

Arnab Basu

Quantitative Methods and Information Systems
Indian Institute of Management, Bangalore 560076
`arnabb@iimb.ernet.in`

The application of Probability Theory provides one of the most powerful techniques for solving problems from Graph Theory. In this talk, we shall describe some of the tools applied in probabilistic arguments, including the basic techniques that use *expectation* and *variance*. We shall also study a few well-known problems from Graph Theory which have been successfully analyzed using probabilistic techniques. The basic *Probabilistic Method* works as follows: In order to prove the existence of a *graphical structure* with certain properties, we construct an appropriate *probability space* and show that a *randomly chosen* element in this space has the desired properties with strictly positive probability. Our emphasis will be on motivating this deep subject by trying to describe the fundamental ideas, and not always give the best possible results if these are too technical to give a clear presentation. In particular, we shall describe the following problems:

- Ramsey Number (see [1]).
- Crossing Number and the Szemerédi-Trotter Theorem (see [2, 3]).
- Random Graphs (see [4]): (i) Clique Number & (ii) Chromatic Number.
- Discrepancy Methods in Graphs (see, e.g., [5]).

References

[1] Ramsey, F. P. (1929); *On a Problem of Formal Logic*, Proc. Lond. Math. Soc., 30(2): 264-286.

[2] Szemerédi, E. and Trotter (Jr.), W. (1983); *A Combinatorial Distinction between Euclidean and Projective Planes*, European J. Comb., 4: 385-394.

[3] Székely, L. (1997); *Crossing Numbers and Hard Erdös Problems in Discrete Geometry*, Combin. Prob. Comp., 6: 353-358.

[4] Erdös, P. and Rényi, A. (1960); *On the Evolution of Random Graphs*, Magyar Tud. Akad. Mat. Kutató Inz. Közl., 5: 17-61.

[5] Matoušek, J. (1999); *Geometric Discrepancy: An Illustrated Guide*, Springer, Algorithms and Combinatorics, 18.

A New Characterization of Matrices with the Consecutive Ones Property

N.S. Narayanaswamy

Department of Computer Science and Engineering
Indian Institute of Technology, Chennai 600036
swamy@cse.iitm.ac.in

We consider the following constraint satisfaction problem: Given a set F of subsets of a finite set S of cardinality n, and an assignment of intervals of the discrete set $\{1, \dots, n\}$ to each of the subsets, does there exist a bijection $f : S \to \{1, \dots, n\}$ such that for each element of F, its image under f is same as the interval assigned to it. An interval assignment to a given set of subsets is called *feasible* if there exists such a bijection. In this paper, we characterize feasible interval assignments to a given set of subsets. We then use this result to characterize matrices with the Consecutive Ones Property(COP), and to characterize matrices for which there is a permutation of the rows such that the columns are all sorted in ascending order. We also present a characterization of set systems which have a feasible interval assignment.

The COP is an interesting and fundamental combinatorial property of binary matrices. The COP appears in many applications; data retrieval, DNA physical mapping, sequence assembly, interval graph recognition, and recognizing Hamiltonian cubic graphs. Testing if a given graph is an interval graph, and testing if a given cubic graph is Hamiltonian are applications of algorithms for testing if a given 0-1 matrix has COP. The maximal clique-vertex incidence matrix is tested for COP to check if a given graph is an interval graph [5]. Similarly, from [12] a cubic graph is Hamiltonian if and only if the matrix $A + I$ has a permutation of rows that leaves at most two blocks of consecutive ones in each column. A is the adjacency matrix of the given graph and I is the identity matrix. Testing if a matrix has COP is also applied for constructing physical maps by hybridization (see [9]), and testing if a database has the consecutive retrieval property (see [4]). To ask for a permutation of the rows such that each column is sorted is a natural extension of the COP. For 0-1 matrices this question is studied as the concept of 1-drop matrices in [2].

Previous work. The first mention of COP, according to D.G. Kendall [8], was made by Petrie, an archaeologist, in 1899. Some heuristics were proposed for testing the COP in [11] before the work of Fulkerson and Gross [3] who presented the first polynomial time algorithm. Subsequently Tucker [13] presented a characterization of matrices with the COP based on certain forbidden matrix configurations. Booth and Lueker [1] proposed the first linear time algorithm for the problem using a powerful data structure called the PQ-Tree. This data structure exists if and only if the given matrix has the COP. Hsu [7] presented another linear time algorithm for testing COP without using PQ-trees. More recently in 2001, he introduced [6] a new data structure called PC tree as a generalization of PQ-Tree. This was used to test if a binary matrix has the CiRcular Ones Property (CROP). Another generalization of the PQ-tree is the PQR-tree introduced by Meidanis and Munuera [10]. This generalization was a nice extension of the approach of Booth and Leuker so that PQR-trees are defined even for matrices that do not possess the COP. Further, for matrices that do not have the COP, the PQR-tree points out specific subcollections of columns responsible for the absence of the COP [9]. In 2003, an almost linear time algorithm has been proposed [9] to construct a PQR-tree.

Our Work. Our motivation in this work was to understand the Consecutive Ones Testing (COT) algorithm due to [7] and to extend it to finding a permutation of the rows of matrix such that the columns are all sorted. Clearly, to sort just one column, we can easily identify a family of row permutations that achieves the sorting. So for each column in a given matrix we can associate a set of *sorting permutations*. The question now is whether the intersection of these sets, one per column, is empty or not? In this paper we identify a natural succinct representation of the *sorting permutations* of a column. This leads to the question that we pose in the abstract: Given an interval assignment to a set system, is it feasible? We then present a necessary and sufficient condition for an interval assignment to be feasible. In particular, we show that an interval assignment to a set system is feasible if and only if it preserves the cardinality of the intersection of every pair of sets. While a feasible interval assignment must necessarily satisfy this property, to our surprise we do not find this characterization in the literature, definitely not explicitly to the best of our knowledge. We use this characterization to characterize matrices with the COP, and characterize matrices whose columns can be sorted by a row permutation. We also show a necessary and sufficient condition for a feasible interval assignment to exist. Our proofs are all constructive and can be easily converted into algorithms that run in polynomial time in the input size. An important consequence of this work is what we view as the modularization of COT algorithm due to Hsu [7]. Two essential modules in the COT algorithm are to find a feasible interval assignment for the columns of a 0-1 matrix, and then to find a permutation that is witness to the feasibility of the interval assignment. Our study in this paper can also be seen as a different angle of study, and yet along the line of work initiated by Meidanis et al. [10, 9]. In their work, they study the set system associated with the columns of the matrix. In particular their results find a *closure* of the set system which also has the COP if the given set system has the COP. In this paper, we take another natural approach to study the set system associated with the columns of the matrix. We consider the set of row permutations that yield consecutive ones in the columns of a matrix. We then ask how this set gets pruned when a new column is added to the matrix. In the process of answering this question, we use the decomposition of the given matrix into prime matrices as done in [7]. Our work also opens up natural generalizations of the COP. For example, given a matrix is there a permutation of the rows such that in each column the rows are partitioned into at most two sorted sets of consecutive rows?. This would be an interesting way to classify matrices, and the combinatorics of this seems very interesting and non-trivial. This would also be a natural combinatorial generalization of the k-drop property for 0-1 matrices which is studied in [2] and references therein.

References

[1] K. S. Booth and G. S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs and Graph Planarity Using PQ-Tree Algorithms. *Journal of Computer System Science*, 1976.

[2] R. Chandrasekaran, S.N. Kabadi, and S.Lakshminarayanan. An extension of a theorem of Fulkerson and Gross. *Linear Algebra and its Applications, Vol. 246, 23-29*, 1996.

[3] D. Fulkerson and O.A.Gross. Incidence Matrices and Interval Graphs. *Pacific Journal of Mathematics*, 1965.

[4] S. Ghosh. File organization: the consecutive retrieval property. *Communications of the ACM*, 1979.

[5] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

[6] W.L.Hsu. PC-Trees vs. PQ-Trees. *Proc. of the 7th Annual International Conference on Computing and Combinatorics, 207-217, Lecture Notes in Computer Science 2108*, 2001.

[7] W. L. Hsu. A simple test for the consecutive ones property. *Journal of Algorithms 42, 1-16*, 2002.

[8] D. Kendall. Incidence matrices, Interval graphs and Seriation in Archaeology. *Pacific Journal of Mathematics*, 1969.

[9] O. P. J. Meidanis and G. Telles. On the Consecutive Ones property. *Discrete Applied Mathematics 88,325-354*, 1998.

[10] J. Meidanis and E. Munuera. A theory for the Consecutive Ones Property. *Proceedings of the III South American Workshop on String Processing, 194-202*, August 8-9, 1996.

[11] W.S.Robinson. A method for Chronologically Ordering Archaelogical Deposits. *American Antiquity*, 1951.

[12] R. Wang and F.C.M. Lau and Y.C. Zhao. Hamiltonicity of Regular Graphs and Blocks of Consecutive Ones in Symmetric Matrices. Discrete Applied Mathematics, Vol. 155, Issue 17, 2312-2320, October 2007.

[13] A. Tucker. A Structure theorem for the Consecutive Ones Property. *Journal of Combinatorial Theory, Series B, Vol. 12, No. 2., pp. 153-162*, April 1972.

Geometric Modeling for Shape Classes

Amitabha Mukerjee

Department of Computer Science and Engineering
Institute of Technology, Kanpur 208 016
amit@cse.iitk.ac.in

Geometrical objects are computationally modeled with a set of *primitives* and a set of operations or *compositions* with well-defined semantics. The primitives may be topological features such as vertices and edges, connected by a tree syntax; or they may be volumetric primitives (including curved surfaces) that are combined with boolean operators or with sweep motions.

In this lecture, we will consider some of the mathematical fundamentals by which representations are created for specific rigid geometric shapes. Starting with the fundamentals of vector spaces, orthogonal transformations, and some basic notions of topological spaces, we develop some approaches for representing primitives, primarily as *boundary models*, and and a set of operations on them, such as *boolean operations* and continuous combinations such as *extrusion*. These form the basis for models that are widely used in graphics and CAD modeling. We also consider some fundamental algorithms on boundary models, such as *point membership* for 2D and 3D polyhedra.

In the last part of the lecture, we consider how these systems may be able to handle *variational shape classes*. In some situations, variation may arise owing to the nature of the input – e.g. curved solids are often defined based on point cloud data which is amenable to one class of approximations. On the other hand, natural shape variations (e.g. the class of "beetles" or "postures of an elephant") are variable in radically different ways. Such variations are related to cognitive models or "image schema" and sometimes involve more than a single prototype.

Two approaches are considered for extending the above representation to variational classes – variations in the primitives, and variations in the semantics (model) of the syntactic operation. We outline how these may be used to construct variational models, resulting in continuously varying shape classes.

References

[1] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, 2002.

[2] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[3] J. McCann and N. Pollard. Responsive characters from motion fragments. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 6, 2007.

[4] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[5] J. B. M. Mizuguchi and T. Calvert. Data driven motion transitions for interactive games. In *Eurographics 2001 Short Presentations*, 2001.

[6] L. Zhao, A. Normoyle, S. Khanna, and A. Safonova. Automatic construction of a minimum size motion graph. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 27–35, 2009.

Projective Geometry for Graphics and Computer Vision

Subhashis Banerjee

Department of Computer Science and Engineering
Indian Institute of Technology, New Delhi 110 016
`Subhashis.Banerjee@cse.iitd.ac.in`

In this tutorial talk we will cover the basics of projective geometry as applied to computer vision and computer graphics. The pin-hole camera is the traditional mathematical tool that has been used to model linear perspective. This model is non-linear and cannot be captured effectively using a linear geometry. In this talk we will introduce the fundamentals of spherical and projective geometries and investigate how the imaging process can be modelled effectively using a projective model.

We will introduce a hierarchy of camera models in terms of Euclidean, affine and projective geometries and analyze the anatomy of a projective camera. We will use the principles of linear perspective and projective geometry to derive theories and constraints for 3D reconstruction of objects from one or more images. To this end we will derive the theory of multiple view geometry and touch upon the algorithmic principles of 3D reconstruction.

We will also illustrate how the theory of projective geometry can be used to generate novel images of an unknown scene from new view-points and demonstrate the use of the principle in designing walk-through and some other applications.

References

[1] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521623049, 2000.

[2] Roger Mohr and Bill Triggs, *Projective Geometry for Image Analysis*, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MOHR_TRIGGS/isprs96.html.

[3] Subhashis Banerjee, *Projective geometry, camera models and calibration*, http://www.cse.iitd.ernet.in/ suban/vision/geometry/geometry.html.

[4] Subhajit Sanyal, *Interactive image based modelling and walk-through planning*, Ph. D. Thesis, Department of Computer Science and Engineering, IIT Delhi, 2006.

Motion Graphs for Interactive Character Animation

Parag Chaudhuri

Department of Computer Science and Engineering
Indian Institute of Technology, Mumbai 400 076
`paragc@cse.iitb.ac.in`

As the quest for realistic character animation has matured, the use of captured motion data for motion synthesis has gained significant support. The challenge lies in searching large collections of motion data at real-time rates in order to find motions that satisfy constraints specified by the user during interaction with the character.

Structured graphs (often called *move trees*) [4, 5] have been used for a long time in computer games. To construct a move tree, game designers first design all the behaviors required by the game and all the logically allowed transitions between the behaviors. Then, they carefully capture the motions according to the pre-planned behaviors so that initial motions have similar start and end postures. Finally, they hand edit the motion capture clips and choose the exact place in the motions for good transitions to appear. The motion clips or frames become the nodes of the graph with the edges representing transitions between these motions [6]. Creating these graph requires substantial amount of manual work.

Methods to automatically construct motion graphs have recently appeared in literature [2, 1, 3]. Given a corpus of motion capture data, we can automatically construct a directed graph called a motion graph that encapsulates connections among the database. The motion graph consists both of pieces of original motion and automatically generated transitions. Novel motions can be generated simply by building walks on the graph. In this talk, we will look at a various techniques for constructing and using motion graphs to synthesize motions that meet a user's specifications at interactive frame rates and the ways in which they can be used to control animated characters.

References

[1] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, 2002.

[2] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[3] J. McCann and N. Pollard. Responsive characters from motion fragments. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 6, 2007.

[4] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[5] J. B. M. Mizuguchi and T. Calvert. Data driven motion transitions for interactive games. In *Eurographics 2001 Short Presentations*, 2001.

[6] L. Zhao, A. Normoyle, S. Khanna, and A. Safonova. Automatic construction of a minimum size motion graph. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 27–35, 2009.

Duality Transformation in Geometry

Partha P. Goswami

Department of Radiophysics and Electronics
University of Calcutta, Kolkata 700009.
ppg.rpe@caluniv.ac.in

Concept of duality is a powerful tool for the description, analysis, and construction of algorithms. A reason for this is that duality allows us to look at a problem from a different angle. Duality has been applied for the design of efficient algorithms for a large number of problems in computational geometry. In this lecture we shall define duality, discuss some of its basic properties and describe, with a few examples, how the concept of duality has been exploited for designing efficient algorithms for problems in computational geometry. Though we shall restrict ourselves in two dimensional cases, the concept of duality is applicable in higher dimensions also.

In the Cartesian plane, a point has two parameters (x- and y-coordinates) and a (non-vertical) line also has two parameters (slope and y-intercept). We can thus *map* a set of points to a set of lines, and vice versa, in an one-to-one manner. This natural duality between points and lines in the Cartesian plane has long been known to geometers.

There are many different point-line duality mappings possible, depending on the conventions of the standard representations of a line. Each such mapping has its advantages and disadvantages in particular contexts. The particular definition we use here is as follows.

Definition 1 *The dual of a point $p(a, b)$ is the line $D_p(y = ax - b)$ and, in a similar manner, the dual of a line $l(y = cx + d)$ is the point $D_l(c, -d)$.*

Observe the asymmetry in the definition. The definition we have given above is known as *m-c* duality. An alternative definition, called *polar duality*, is also used.

Definition 2 *Dual of a point p with coordinates (a, b) in the primal plane is the line T_p with equation $ax + by + 1 = 0$ in the dual plane and vice versa.*

Geometrically this means if d is the distance from the origin O to the point p, the dual T_p of p is the line perpendicular to Op at distance $1/d$ from O and placed on the other side of O.

We now discuss some algorithms which uses duality for their work. One of the most famous problems in computational geometry is the convex hull problem. It can be stated as follows. Given a set $\mathcal{P}$ of n points in the plane, compute the convex hull $CH(\mathcal{P})$ of the point set $\mathcal{P}$. We introduce the concept of convex hull, which is the smallest convex set containing $\mathcal{P}$, and then describe an optimal algorithm which uses duality for computing the convex hull of a point set $\mathcal{P}$. The time and space complexities of the algorithm are $O(n \log n)$ and $O(n)$ respectively.

Let $\mathcal{L}$ be a set of n lines in the plane. The embedding of $\mathcal{L}$ in the plane induces a planner subdivision that consists of *vertices*, *edges*, and *faces*. This subdivision is referred to as *arrangement* induced by $\mathcal{L}$, and is denoted by $A(\mathcal{L})$. Algorithms for a surprising number of problems in computational geometry are based on constructing and analyzing the arrangement of a specific set of lines. We first introduce a simple alternative structure, known as *level structure*, for describing an arrangement of lines and then describe an algorithm for computing the level structure. The algorithm runs in $O(n^2 \log n)$ time using $O(n^2)$ space. We also describe an important paradigm, called *sweep line* paradigm, on which the algorithm is based.

We then consider an algorithm which uses sweep line paradigm and duality in an elegant way to solve the following problem. Let $\mathcal{P}$ be a set of n points in the plane. Determine which of the $\binom{n}{3}$ triangles with vertices in $\mathcal{P}$ has the smallest area. Time and space complexities of the algorithm is $O(n^2 \log n)$ and $O(n)$ respectively.

Lastly, we consider the nearest neighbor of a query line problem. Given a set $\mathcal{P}$ of n points in the plane and a query line l, the problem is to compute the nearest neighbor (in the perpendicular distance sense) of the query line l. Though the problem can be solved optimally in $O(n)$ time in an obvious way, we consider the *multishot query* version of the problem. Here we are allowed to *preprocess* the point set $\mathcal{P}$ in a suitable manner such that queries can be solved efficiently. Our algorithm is simple and uses duality and level structure of an arrangement of lines. With $O(n^2 \log n)$ preprocessing time and $O(n^2)$ space, our algorithm can compute the nearest neighbor of a non-vertical query line in $O(\log^2 n)$ time.

References

[1] Mark de Berg, Marc van Kreveld, Mark Overmars and Otfried Schwarzkof, *Computational Geometry: Algorithms and Applications*, Springer, 1997.

[2] Bernard Chazelle, Leo Guibas, and D. T. Lee, *The power of geometric duality*, Bit, 25:76-90, 1985.

[3] Herbert Edelsbrunner, *Algorithms in Computational Geometry*, Springer, 1987.

[4] D. T. Lee, *The Power of Geometric Duality Revisited*, Information Processing Letters, 21:117-122, 1985.

[5] Subhas C. Nandy, Sandip Das, and Partha P. Goswami, *An efficient k nearest neighbors searching algorithm for a query line*, Theoretical Computer Science, 299:273-288, 2003.

Robot Path Planning:
Off-line and On-line Algorithms

Subir Kumar Ghosh

School of Technology & Computer Science
Tata Institute of Fundamental Research, Mumbai 400005.
`ghosh@tifr.res.in`

In this lecture, we introduce a few basic algorithms for robot path planning in the plane. A basic motion planning problem is to produce a continuous motion connecting starting and destination positions while avoiding collision with obstacles. The robot and obstacle geometry is described in a 2D or 3D workspace, while the motion is represented as a path in (possibly higher-dimensional) configuration space [3, 5]. The configuration space represents all possible placements of robots, and the subset of the configuration space, where the robot can move without collision, represents the free configuration space, which can be computed using Minkowski sums (see Figure 1).

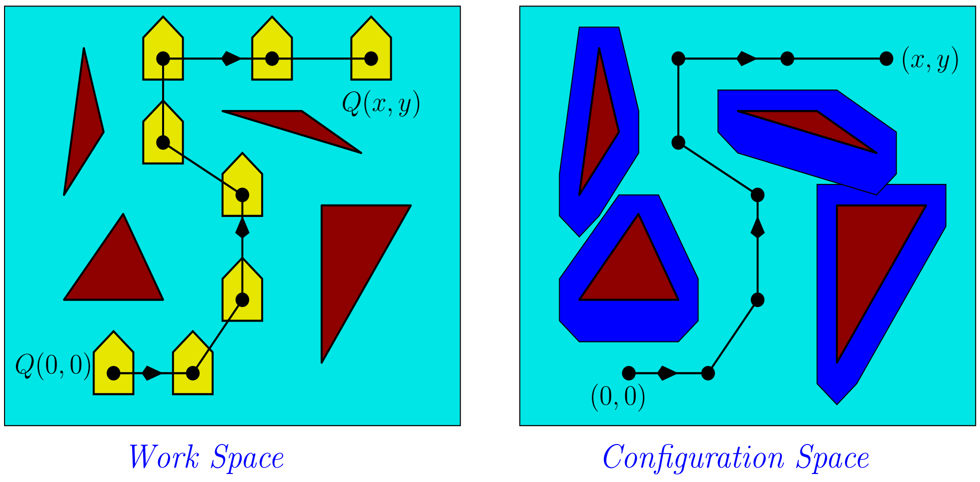


Figure 1: Work space and configuration space of a planar robot.

By constructing a road map using cell decompositions, visibility graphs etc. [1], a collision free path of a robot can be computed. We also discuss how to compute the geometric shortest path and minimum link path inside a free configuration space [4] (see Figure 2).

After presenting off-line algorithms, we present a few basic on-line algorithms that have been designed for mobile robots for searching a target and for exploring a region in the plane (see Figure 2). Consider a natural scenario when the robot does not have the complete knowledge of the geometry of R apriori and also does not know the location of the target t, but the target can be recognized by the robot. In such a situation, the robot is asked to reach t from its starting position s using its sensory input provided by acoustic, visual, or tactile sensors of its on-board sensor system. The problem here is to design an efficient on-line algorithm which a robot can use to search for the target t. Observe that any such algorithm is 'on-line' in the sense that decisions must be made based only on what the robot has

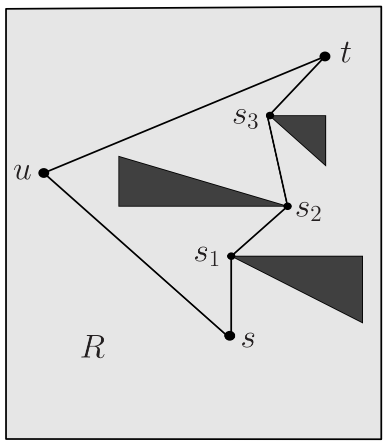


Figure 2: The shortest path and the minimum link path from s to t.

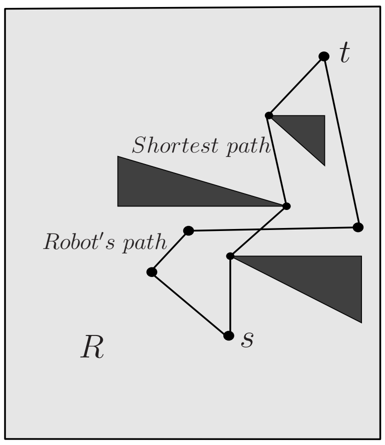


Figure 3: Two paths from s to t computed by on-line and offline algorithms.

received input so far from its sensor system. In an exploration problem, a robot is asked to explore or see all points of an unknown environment rather than searching for a particular point t.

One of the difficulties in working with incomplete information or with only local information is that the path cannot be pre-planned and therefore, its global optimality can hardly be achieved. Instead, one can judge the on-line algorithm performance based on how it stands with respect to other existing or theoretically feasible algorithms, or how 'reasonable' they are from the human watchman point of view. As is the case with other on-line problems [6], the efficiency of on-line algorithms for searching and exploration algorithms is generally measured using their competitive ratios [2]:

$$Competitive\ ratio = \frac{cost\ of\ the\ on-line\ algorithm}{cost\ of\ an\ optimal\ off-line\ algorithm}$$

Let us explain the meaning of the competitive ratio through an example (see Figure 3). Consider the earlier stated problem of moving from s to t in the region R. Let L be the length of the path traversed by a robot using an on-line algorithm A to reach from s to t. On the other hand, an offline algorithm for this problem computes the Euclidean shortest path from s to t of length, say, L', and then this optimal path is followed by the robot to traverse from s to t. Let $c = L/L'$. The ratio c is called the *competitive ratio* of the on-line algorithm A. Intuitively, the competitive ratio tells how good is the path computed by the on-line algorithm in comparison to the best possible path. It implies that if c' is the competitive ratio of another on-line algorithm B for the same problem and $c < c'$, then the on-line algorithm A is more efficient than the on-line algorithm B. An on-line algorithm A with competitive ratio c for a given problem is said to be an *optimal on-line algorithm* for this problem if every on-line algorithm B for the same problem has a competitive ratio $c \leq c'$ in the worst case. Note that an optimal on-line algorithm can have a competitive ratio greater than 1.

References

[1] M. de Berg, M. Van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, 1997.

[2] P. Berman, *On-line searching and navigation*, Lecture Notes in Computer Science 1442, pp. 232-241, Springer, 1996.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, 2005.

[4] S. K. Ghosh, *Visibility Algorithms in the Plane*, Cambridge University Press, Cambridge, United Kingdom, 2007.

[5] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[6] D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, Communication of ACM, 28: 202-208, 1985.