Approximate Nearest-Neighbor Search and Rapidly-Exploring Random Trees

Kevin Do
kkd10@duke.edu

Abstract

The rapidly-exploring random tree (RRT) is a randomized data structure used in motion planning, one of the fundamental problems in robotics. The classical RRT uses nearest-neighbor search as one of its subroutines, but practical implementations of the RRT often substitute an approximate nearest-neighbor (ANN) subroutine to achieve drastic performance improvements. Very few theoretical results about the classical RRT are known, and virtually no analysis has been done on the RRT with ANN substituted for nearest-neighbor search. We survey previous work done separately on RRTs and ANN search before stating several new results about RRTs constructed with an ANN subroutine.

1 Introduction

In robotics, motion planning is the process by which the specifics of a movement are computed to satisfy constraints (e.g. avoid collisions) and optimize some feature of the computed path. There exist two natural spaces for describing motions. The *workspace* is the vector space that describes the physical space that the robot occupies; a typical workspace might have the topology[1] $\mathbb{R}^3 \times (S_1)^3$, with three coordinates to describe its (x, y, z) position and three coordinates to describe its orientation in terms of pitch, yaw, and roll. The *configuration space* is the vector space defined by the generalized coordinates of the system. For example, a robotic arm with 7 joints and 3 extendable links might have a natural configuration space that is isomorphic to $(S_1)^7 \times I^3$, where $I \subset \mathbb{R}$ is some closed interval.

A primary difficulty with motion planning is that the state of the robot is often most easily described as a point in configuration space, whereas obstacles are often most easily specified as constraints in

[1] In this paper, $\mathbb{R}$ refers to the real numbers, and S_1 to the topology of the circle, often used in robotics to describe rotations.

the workspace. Transforming a point in configuration space to the workspace (forward kinematics) can be tractable, but the inverse operation is typically infeasible (indeed, there may be many points in configuration space that correspond to the same point in the workspace) [Lat12].

Early motion planners worked directly in the workspace, using techniques like swept volumes to determine whether a path was collision-free [LL05]. These techniques are not easily generalizable, however, as they rely on detailed knowledge about the specifics of the robot and its environment. Planning in configuration space offers the possibility of more general techniques, but it introduces a new question: given that inverse kinematics of a single point in the workspace is difficult, how should entire obstacles be transformed from the workspace to the configuration space?

In low-dimensional spaces, there exist efficient, exact algorithms for computing motions [LaV06]. In high dimensions, however, such algorithms quickly become computationally intractable. Instead, most practical motion planning methods in high dimensions are sampling-based. Sampling-based motion planning algorithms indirectly take advantage of the feasibility of forward kinematics by using a collision detection module as a black box. This sidesteps the difficult problem of obtaining explicit representations of the obstacles in configuration space. In typical practical planning problems, sampling-based motion planners have relatively good performance compared to exact planners, but this comes at the expense of relaxed guarantees about the optimality and existence or non-existence of solutions.

One example of a sampling-based configuration space planner is the the probabilistic roadmap, which constructs a graph by generating random configurations and connecting nearby configurations in a "local planning" step [AW96]. If there are a few simple constraints, then this technique can work well. However, in general there can be many complicated constraints (e.g. differential or nonholonomic ones), which cause the local planning step to be as difficult as the original planning problem.

2 Rapidly-Exploring Random Trees

The rapidly-exploring random tree (RRT) method was first introduced by Steven LaValle in 1998 [LaV98]. The RRT takes a control-theoretic approach to motion planning by defining a state transition equation $\dot{x} = f(x,u)$, where u is a vector of inputs. Integrating f over some time interval is equivalent to applying the input for that time interval. By encoding constraints into the structure of $f(x,u)$ itself, RRTs avoid the need to connect discrete milestones. This is a major advantage of the RRT over other sampling-based planning techniques like the probabilistic roadmap [KŠLO96].

Consider the path planning problem in a metric space X (which we call the configuration space or state space) between the start state x_{init} and the goal state x_{goal}. X can be partitioned into a free space X_{free} and an obstacle region X_{obs}. Assume the existence of a black box subroutine that returns whether a given state x lies in X_{free} or X_{obs}. As presented in LaValle's first paper [LaV98], the method to construct an RRT T with K vertices can be described succinctly:

- Initialize a tree T and insert the initial state x_{init}.
- Repeat K times:
  - Select a random state $x_{\text{rand}} \in X$.
  - Find the nearest neighbor x_{near} of x_{rand} among the vertices of T.
  - Select a control u that minimizes the distance from x_{near} to x_{rand}, while simultaneously ensuring that the state does not collide with any obstacles.
  - Apply the control u for some time Δt to x_{near} obtain a new state x_{new}.
  - Add x_{new} to T, along with an edge from x_{near} to x_{new}.

The algorithm given above can not be used directly to answer motion planning queries, since it merely grows T to explore the state space. There are a variety of techniques to modify the algorithm above so that it can be used to answer motion planning queries. RRT_BIDIRECTIONAL grows two RRTs, one from x_{init} and one from x_{goal}, periodically attempting to connect the trees together. Another alternative is to select $x_{\text{rand}} \leftarrow x_{\text{goal}}$ with a small probability at each iteration [LK01].

The name "rapidly-exploring random tree" stems from the algorithm's growth pattern, as the search tree's growth is strongly biased towards unexplored regions in configuration space. This fact can be seen by drawing the Voronoi diagram of T: at any step, the largest Voronoi cells will be associated with the leaf nodes of T. This rapid exploration is therefore a consequence of the nearest-neighbor search. In fact, it has been shown that an apparently "random" expansion algorithm that grows the tree from a random vertex in a random direction is highly biased towards regions it has already visited [LK01].

3 Approximate nearest-neighbor search

The nearest-neighbor search, crucial as it is to growth pattern of the RRT, is typically the slowest subroutine called by the procedure. A naïve, brute-force solution to the nearest-neighbor search problem takes time linear in the size of the search tree. In low-dimensions, geometric structures like kd-trees may be used to speed up queries [FBF77], but such approaches often perform worse than naïve linear search in high dimensions because many of these data structures have an exponential dependence on the dimensionality of the space [DL76, Cla88, AM93]. However, there exist approximate algorithms and data structures for computing the nearest neighbor that scale better to high dimensions. Indeed, these approximations are often used in practice [MA98, ML09].

Approximate relaxations of nearest-neighbor search come in many flavors. One of the most popular is the $(1+\varepsilon)$-approximate nearest neighbor problem: Given a set S of points, a query point q, and a distance function $d(\cdot,\cdot)$, find a point $p \in S$ such that $\forall p' \in S$, $d(p,q) \leq (1+\varepsilon)d(p',q)$. It is also common to relax the problem further by only requiring that the returned point be an ε-approximate nearest neighbor with some given probability.

There has been a fair amount of previous work on approximate nearest-neighbor (ANN) search. One prominent technique is *locality-sensitive hashing* (LSH) [GIM$^+$99]. The general strategy of locality-sensitive hashing is to hash points from a set in a manner such that the probability of collision is much higher for points that are nearby than for points that are far away. Queries for the nearest neighbors to a point can then be performed by scanning the bucket into which the query point was hashed. These sorts of techniques saw a flurry of research activity in the early 2000s, starting with the first LSH paper in [GIM$^+$99] and finally with a near-optimal algorithm in [AI06]. In fact, an optimal LSH algorithm for ANN has been found [AR15], but it is data-dependent and therefore does not allow for updates to the pointset (and therefore can not be used without modification in the RRT method.)

Other ANN schemes rely not on hashing but on sophisticated data structures. One, called *best-bin first*, is best understood as a variant of the k-d tree search algorithm [BL97]. Yet another ANN technique is known as the balanced box-decomposition tree [AMN$^+$98], which computes a $(1+\varepsilon)$-approximate nearest neighbor in the $O(c_{d,\varepsilon}\log n)$ time, where $c_{d,\varepsilon} = O(d(1+d/\varepsilon)^d)$, with a preprocessing time of $O(dn\log n)$ and requiring $O(dn)$ space. These bounds are asymptotically optimal in the algebraic decision tree model of computation. The balanced box-decomposition tree manages this runtime by subdividing the state space into regions of $O(d)$ complexity by axis-aligned hyperrectangles.

4 RRT-ANN

We now turn to the application of ANN techniques to the RRT. The results discussed here do not rely on the particulars of any one ANN technique and therefore apply to RRTs constructed with any ANN algorithm that satisfies the basic conditions described below. Let RRT-ANN refer to the modified RRT (along with its corresponding construction procedure) where nearest-neighbor search has been replaced with ANN. We now show that several results about RRTs extend naturally to RRT-ANNs.

Claim 4.1. *The growth of an RRT-ANN is biased towards leaf nodes and unexplored regions of state space.*

One of the most desirable properties of a classical RRT is that its expansion is heavily biased towards unexplored portions of the state space X. As noted above, leaf nodes will have the largest Voronoi cells, and will therefore have the largest chances of being selected for expansion. This idea extends naturally to RRT-ANNs. Instead associating each node of the RRT-ANN with its Voronoi cell, we associate each node $v \in T$ with the set of points $\text{ApproxVor}(v) : \{p \in X | v \in \text{ANN}_T(p)\}$, where $\text{ANN}_T(p)$ is the set of points in T that might be returned by the ANN subroutine when queried with p. Because $\text{ApproxVor}(v)$ is a superset of the Voronoi cell of v, leaf nodes of an RRT-ANN will still have a higher likelihood of being selected for expansion as long as the ANN subroutine is reasonably selective (e.g. an ε-approximation). The more exact the ANN, the closer $\text{ApproxVor}(v)$ is to to the Voronoi cell of v, which leads naturally to Claim 4.2.

Claim 4.2. *The exactness of the ANN procedure is a parameter that trades runtime performance for bias towards unexplored regions of the state space.*

We turn now to the analysis of the probabilistic completeness of the RRT-ANN. Let $d_k(x)$ be the distance from x to the closest vertex in the RRT-ANN T when T has k vertices. Furthermore, let us first restrict ourselves to holonomic constraints, so that $\dot{x} = f(x,u) = u$, and to an ANN procedure that has a non-zero probability of returning the nearest neighbor of a query point.

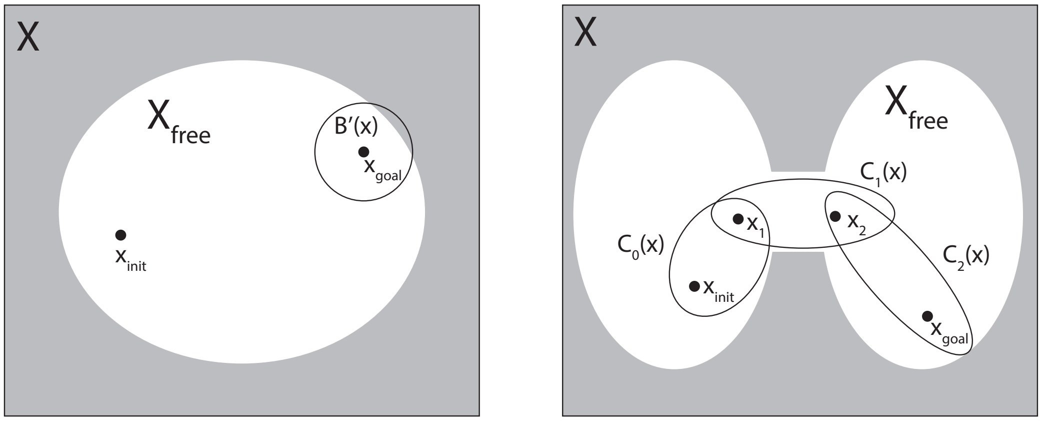

Figure 1: A state space $X \subset \mathbb{R}^2$ with a convex (left, Lemma 4.3) and non-convex (right, Lemma 4.4) collision-free subset.

Lemma 4.3. *Suppose X_{free} is a convex, bounded, connected, open subset of a state space X. For any $x_{\text{goal}} \in X_{\text{free}}$ and positive constant $\varepsilon > 0$, $\lim_{k\to\infty} \Pr[d_k(x) < \varepsilon] = 1$.*

Proof. This proof is modified from the one given in [LKJ00] for the classical RRT. Let x_{init} be an initial RRT-ANN vertex, and let $B(x)$ denote a ball of radius ε centered on x, and let $B'(x) = B(x) \cap X_{\text{free}}$. Because $B'(x)$ has positive, finite volume, there is a strictly positive probability at each iteration that x_{rand} lies in $B'(x)$. Assume all RRT-ANN vertices lie outside of $B(x)$, for otherwise the lemma has already been satisfied. Then $E[D_k(x)] - E[D_{k+1}(x)] > b$ for some positive real number $b > 0$, since we assumed that there was a positive probability that the ANN procedure would return the true nearest neighbor and therefore that T would grow from the true nearest neighbor to the randomly chosen point in $B'(x)$. This implies $\lim_{k\to\infty} \Pr[d_k(x) < \varepsilon] = 1$. □

Lemma 4.3 is reassuring, but its result is distinctly weaker than empirical RRT performance. In practice, aggressive RRT-ANNs with a large Δt can make progress towards x_{goal} in a open, convex free space even when the ANN procedure does not return the true nearest neighbor. Nevertheless, we can construct a pathological case in which the ANN routine consistently returns a point other than the true nearest neighbor, which is then extended towards x_{rand} without decreasing $d_k(x)$. We now extend Lemma 4.3 to a nonconvex free space.

Lemma 4.4. *Suppose X_{free} is a nonconvex, bounded, connected, open subset of a state space X. For any $x_{\text{goal}} \in X_{\text{free}}$ and positive constant $\varepsilon > 0$, $\lim_{k\to\infty} \Pr[d_k(x) < \varepsilon] = 1$.*

Proof. Let $x_0 = x_{\text{init}}$ be an initial RRT-ANN vertex. Because x_0 and x lie in the same connected component, there exists a collision-free path P between them, and there exists a sequence, $x_0, x_1, x_2, \ldots, x_n$, of states such that $x_n = x$ and, for any point x_i and its successor x_{i+1}, both x_i and x_{i+1} can be contained by a convex, bounded, open, collision-free subset $C_i(x)$ of X. (Consider sampling P at increasingly fine resolution until the sequence satisfies the desired properties.)

We proceed by induction on the convex subsets $C_i(x)$. Assume as an inductive hypothesis that the RRT contains some node in C_i(x). Because $x_{i+1} \in C_i(x)$, the RRT will eventually come arbitrarily close to x_{i+1} by Lemma 4.3, and so the RRT will have a node in $C_{i+1}(x)$ since $x_{i+1} \in C_{i+1}(x)$. All that remains is to note that $x_0 \in C_0(x)$ so that we may conclude that eventually the RRT will have a node in $C_{n-1}(x)$. One final application of Lemma 4.3 gives the desired result. □

Though Lemmas 4.3 and 4.4 hold for the RRT-ANN as well as they hold for the RRT, we expect the rate of convergence to be slower for the RRT-ANN, since the growth may occur from a node other than the nearest one. Nevertheless, these lemmas show that RRT-ANNs retain the probabilistic completeness of RRTs.

We now characterize the rate of convergence of the RRT-ANN, adapting techniques from [LK01] (with some of the mistakes from that paper corrected). Let $A = \{A_0, A_1, A_2, \ldots, A_k\}$ be a sequence of subsets of X called an *attraction sequence*[2], and let $A_0 = \{x_{\text{init}}\}$ and $A_k = \{x_{\text{goal}}\}$. The attraction sequence must be constructed such that, for each A_i, there exists a *basin* $B_i \subseteq X$ such that

1. If the RRT contains a vertex in A_{i-1} and x_{rand} is selected to be in A_i, then the ANN procedure always returns a point in B_i.
2. For all $x \in A_i$ and $y \in B_i$, there must exist an input u to the state-transition equation that takes y to x.

Intuitively, any valid attraction sequence that we can construct helps give an upper bound on the expected number iterations required to connect x_{init} to x_{goal}. Each basin B_i can be considered to be a potential well: if the RRT-ANN contains a vertex in A_{i-1}, then the existence of the basin B_i ensures that the RRT-ANN will add a vertex to A_i if x_{rand} is selected to lie in A_i.

Let $\mu(\cdot)$ denote the volume (or measure) function on X.

Theorem 4.5. *If an attraction sequence of length k exists and the RRT-ANN is configured with a step size large enough to always attempt to connect to x_{rand}, then the expected number of iterations required to connect x_{init} to x_{goal} is at most $\sum_{i=1}^{k} \frac{\mu(X_{\text{free}})}{\mu(A_i)}$.*

[2] A_k has zero measure, which interferes with the proofs that follow. This technicality is glossed over here, but this can be remedied by either letting A_k be a ball of small radius centered on x_{goal}, or else choosing $x_{\text{rand}} \leftarrow x_{\text{goal}}$ with small probability at each iteration.

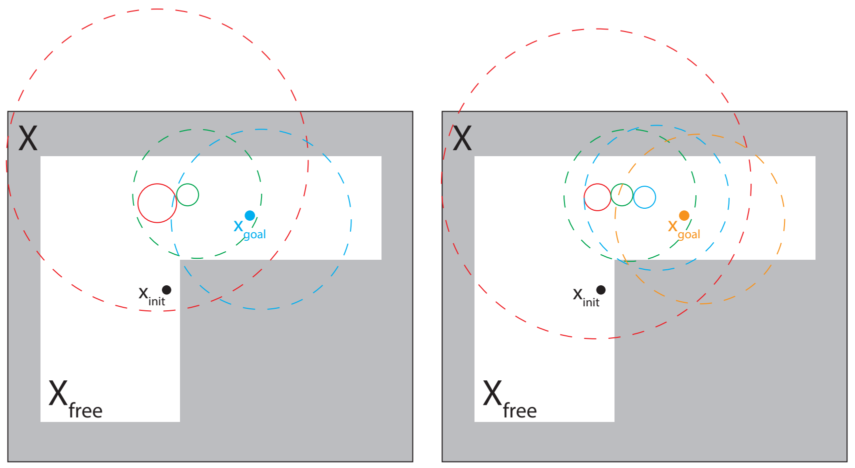

Figure 2: Two attraction sequences from x_{init} to x_{goal} in $\mathbb{R}^2$ with solid A_i and dashed B_i. The attraction sequences were constructed assuming an ANN that returns an $(1+\varepsilon)$-approximate nearest neighbor with $\varepsilon = 0.01$ (left) and $\varepsilon = 0.5$ (right). On the right, note the increased length of the attraction sequence and the smaller A_i, implying a larger upper bound on the expected number of iterations to find a valid path.

Proof. If an RRT-ANN vertex lies in A_{i-1} and x_{rand} lies in A_i, then the RRT-ANN will be connected to x_{rand}. To see this, we can use the first property of the basin to see that the ANN procedure returns a point in B_i. Then, we can use the second property of the basin to see that there exist inputs to get from whichever node the ANN procedure returned and x_{rand}.

In any iteration, the probability that x_{rand} lies in A_i is $p = \frac{\mu(A_i)}{\mu(X_{\text{free}})}$. If we consider the iterations to be Bernoulli trials with a probability of success p, then the expected number of trials to achieve a successful trial is $1/p = \frac{\mu(X_{\text{free}})}{\mu(A_i)}$. To get from x_{goal} to x_{init}, we can go through $A_1, A_2, ...,$ to A_k, and the main result follows from linearity of expectation. $\square$

Theorem 4.6. *If an attraction sequence of length k exists and the RRT-ANN is configured with a step size large enough to always attempt to connect to* x_{rand}, *then the probability that the RRT-ANN fails to find a path after n iterations is at most* $\exp(\frac{2k-np}{2})$, *where* $p = \min_i \dfrac{\mu(A_i)}{\mu(X_{\text{free}})}$.

Proof. Consider the random variable C that represents the number of successes of the n Bernoulli trials described above in the proof for Theorem 4.5. C has a binomial distribution, so we apply a Chernoff-type bound on its tail probabilities by applying Theorem 4.2 from [MR10]. Noting that $\mu = \mathrm{E}[C] = np$, $\Pr[C \leq (1-\delta)\mu] < \exp(-\mu\delta^2/2)$. The RRT-ANN has found a path once $C \geq k$. Setting $\delta = k/(np) < 1$, we can expand the exponent to $-np/2 + k - k^2/(2np)$. Because $-k^2/(2np)$ is negative, we can safely state $\exp(-\mu\delta^2/2) \leq \exp(\frac{2k-np}{2})$. $\square$

5 Further Studies

Like with RRTs in general, there is much theoretical work to be done on RRT-ANNs, as the practical performance of these algorithms continues to far outstrip theoretical analysis. For example, the probabilistic completeness guarantees shown above hold for the holonomic planning case. Clearly there are nonholonomic robots that can not explore their entire state space even with an infinite amount of time (e.g. a car with its steering wheel fixed at a certain angle), but how much can we relax the holonomic constraint and maintain probabilistic completeness?

It would also be useful to have precise bounds on the rate of exploration of both RRTs and RRT-ANNs, as well as information about the dependence of the rate of exploration on the configurable parameters of the RRT-ANN, like the exactness of the ANN procedure and the step size Δt. The upper bound given in Theorem 4.5 on the expected number of iterations to find a path between two points depends on an attraction sequence between the two points, which is rather unwieldy to construct. A simpler method for characterizing RRT-ANN path planning performance would, for example, allow RRT-ANNs to be integrated into realtime embedded systems with strict time constraints.

References

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[AM93] Pankaj K Agarwal and Jirí Matoušek. Ray shooting and parametric search. *SIAM Journal on Computing*, 22(4):794–806, 1993.

[AMN$^+$98] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 793–801. ACM, 2015.

[AW96] Nancy M Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 113–120. IEEE, 1996.

[BL97] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[Cla88] Kenneth L Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[DL76] David Dobkin and Richard J Lipton. Multidimensional searching problems. *SIAM Journal on Computing*, 5(2):181–186, 1976.

[FBF77] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[GIM$^+$99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[KŠLO96] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[Lat12] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[LaV98] Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. 1998.

[LaV06] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[LK01] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[LKJ00] Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[LL05] Stephen R Lindemann and Steven M LaValle. Current issues in sampling-based motion planning. In *Robotics Research. The Eleventh International Symposium*, pages 36–54. Springer, 2005.

[MA98] David M Mount and Sunil Arya. Ann: library for approximate nearest neighbour searching. 1998.

[ML09] Marius Muja and David G Lowe. Flann, fast library for approximate nearest neighbors. In *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, 2009.

[MR10] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.