Wireless coverage prediction via parametric shortest paths

David Applegate
Google Inc., USA
dapplegate@google.com

Aaron Archer
Google Inc., USA
aarcher@google.com

David S. Johnson
(deceased)

Evdokia Nikolova
University of Texas at Austin, USA
nikolova@austin.utexas.edu

Mikkel Thorup
University of Copenhagen, Denmark
mikkel2thorup@gmail.com

Ger Yang
University of Texas at Austin, USA
geryang@utexas.edu

ABSTRACT

When deciding where to place access points in a wireless network, it is useful to model the signal propagation loss between a proposed antenna location and the areas it may cover. The indoor dominant path (IDP) model, introduced by Wölfle et al., is shown in the literature to have good validation and generalization error, is faster to compute than competing methods, and is used in commercial software such as WinProp, iBwave Design, and CellTrace. The previous algorithms known for computing it involved a worst-case exponential-time tree search, with pruning heuristics for speed.

We prove that the IDP model can be reduced to a parametric shortest path computation on a graph derived from the walls in the floorplan. It therefore admits a quasipolynomial-time (i.e., $n^{O(\log n)}$) algorithm. Moreover, we give a practical approximation algorithm based on running a small constant number of shortest path computations. Its provable worst-case additive error (in dB) can be made arbitrarily small, and is well below 1dB for reasonable choices of parameters. We evaluate this algorithm empirically against the exact IDP model, showing that it consistently beats its theoretical worst-case bounds, solving the model exactly (i.e., no error) in the vast majority of cases.

CCS CONCEPTS

• **Mathematics of computing** → *Paths and connectivity problems*; *Approximation algorithms*; • **Networks** → *Wireless local area networks*; *Network design and planning algorithms*;

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Mobihoc '18, June 26–29, 2018, Los Angeles, CA, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5770-8/18/06.
https://doi.org/10.1145/3209582.3209605

KEYWORDS

dominant path model, parametric shortest paths, wireless coverage prediction

ACM Reference Format:
David Applegate, Aaron Archer, David S. Johnson, Evdokia Nikolova, Mikkel Thorup, and Ger Yang. 2018. Wireless coverage prediction via parametric shortest paths. In *Mobihoc '18: The Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, June 26–29, 2018, Los Angeles, CA, USA.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3209582.3209605

1 INTRODUCTION

When installing a wireless network in an office or other building, it can be difficult to determine the best spots to place access points (APs) in order to achieve the desired signal strength throughout the building. Wireless signal propagation is complicated, especially in an indoor office environment where the numerous walls attenuate the signal as it passes through them or diffracts around corners. Since physical trial-and-error is expensive and time-consuming, we desire an effective model to answer the following question: if we place an access point at location s, how strong of a signal will we receive at various other points t throughout the building? In other words, we want to produce a heat map such as in Figure 1. If we can do this quickly and accurately in simulation, it opens the door to many algorithmic approaches for designing the wireless network to provide the desired coverage, throughput, etc.

Our original motivation for this paper came several years ago, when one of our colleagues told us that he had discovered a beautifully simple but heretical indoor radio signal propagation model in the literature, whose results accorded with reality surprisingly well. The model has two controversial features: it uses only the strongest propagation path to estimate the received signal strength at a point, and it completely ignores paths that rely on reflections off of walls, focusing on diffractions around corners as the only mechanism for a path to change direction. This *indoor dominant path* (IDP) model

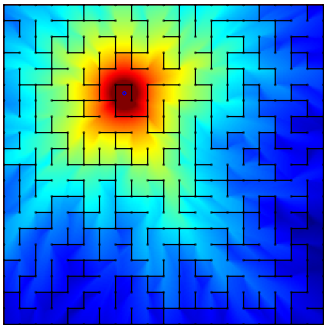

Figure 1: Heat map for a random maze.

lies in stark contrast to the methods blessed by conventional wisdom in the field: sophisticated ray-tracing techniques that expend a large amount of computational effort on tracing individual rays as they bounce off of multiple obstacles, and add up the contributions of multiple rays at each prediction point, accounting for phase shifting of the waves from the differential path lengths and the resulting constructive or destructive interference. Nevertheless, another member of his team was spending his days pushing a cart around the halls, measuring the actual received signal strengths from the WiFi APs in our building, and finding that the IDP model matched reality roughly as well as the much more sophisticated and accepted ray tracing models.

Better yet, the IDP model is fast: the commercial software he was using that featured the IDP model could compute a heat map for our entire office building from a single AP in minutes, many times faster than ray tracing tools. However, O(1min) was still not fast enough to satisfy our colleague. For each possible AP location on a 1m grid, he wanted to compute a heat map for the entire building, also at a 1m resolution, in order to inform his AP placement. For a 60m x 60m building, this would be 3600 heat maps. At 1min apiece, this would take 2.5 days of computation. Obtaining the relevant data to fuel this modeling was a chore unto itself, and he wanted to rerun the models as the input data improved. Could we compute the model faster, he asked?

Sadly, we were not able to do so in time to help our colleague with his project, but we did subsequently design algorithms to compute the IDP model faster, and this paper is the result. On a synthetic 60m x 62m instance meant to model an office building (Section 5), our algorithm takes roughly 1.3sec per heat map, preceded by 2.9sec of pre-computation (which can be amortized across all of the heat maps). Using this algorithm, the full set of heat maps could be computed in under 80min, despite the fact that we have taken no particular care to engineer an optimized implementation.

Our main algorithmic insight is that we can reduce the IDP model to a parametric shortest path computation on an associated graph. By exploiting further structure, we can reduce this to the equivalent of about 2 ordinary, non-parametric shortest path computations on the same graph, while incurring an approximation error that is provably tiny in the worst case and nearly always zero in practice. We call this our *geometric progression* (GP) algorithm, as it involves evaluating a geometric progression of parameter values on geometrically increasing subsets of the graph. Our GP algorithm possesses two benefits over previous algorithms for the IDP model. First, it relies on fast polynomial-time algorithms for shortest path (e.g., Dijkstra's algorithm). Second, it computes path losses for *all measurement points simultaneously*. In contrast, previous algorithms for the IDP model used a worst-case exponential tree search to compute each point-to-point path loss (with pruning heuristics to speed it up), and had to do one such computation per measurement point, rather than handling them all at once.

Plets et al. published a careful study validating the IDP model, reporting superb agreement between IDP and empirical measurements on four buildings with diverse physical characteristics [15]. Nevertheless, it is fair to say that the model has not reached widespread acceptance within the academic community. It has gained more traction within industry, featuring prominently in at least three commercial software packages for wireless signal propagation: WinProp [2], iBwave Design [10], and CellTrace [7]. WinProp, associated with the original inventors of the indoor, outdoor, and mixed dominant path models, counts many large telecom companies and device manufacturers among its customers, including Alcatel, British Telecom, Ericsson, France Telecom, Fujitsu, Intel, Italtel, Kyocera, Motorola, Nokia, Nokia Networks, Sony, Swisscom, T-Mobile, and Vodafone [17].[1] Qualcomm has proved its capability of modeling femtocell performance in urban neighborhoods [8, 12], and Nokia has used it to model LTE multimedia broadcast systems [5]. One practical selling point is the IDP model's insensitivity to fine details of a building's layout [4]. Most ray tracing tools require CAD drawings or other detailed databases describing the geometry of a building, which are sometimes prohibitively expensive or impossible to obtain. In contrast, tools like WinProp can generate reasonable predictions based on as little as a photocopy of a floorplan, plus information on the materials composing each wall.

Although the IDP model may not yet be fully accepted in the academic community, we take its commercial success and the strong validation results cited above as convincing indicators that it merits

[1]This customer list disappeared from WinProp's website after its acquisition by Altair [1], but we reconstructed it from the raw HTML in an archived snapshot [17].

further study. While the community would probably value further validation of the model, that is not the aim of this paper. Here, we take the quality of the model as given, and our goal is to present a new algorithmic approach that can solve it faster. The practical value of this speedup is to enable new use cases such as the one described above, where our colleague wished to compute a separate heat map for each possible transmitter location in a 1m grid, to serve as input for a WiFi network planning tool.

Our Contribution. This paper provides faster computation of the IDP model, with rigorous approximation guarantees in worst-case polynomial running time (Section 4), in contrast to the worst-case exponential time of the existing approaches. Our *geometric progression* (GP) algorithm is also very fast in practice. While the solution it returns may (rarely) be suboptimal, it is guaranteed to be very close to optimal. The algorithm is based on a careful geometric design and analysis related to the shape of the nonlinear objective function when projected on a 2D subspace spanned by the distance and loss parameters in the model. For reasonable parameter settings, the worst-case additive error is a fraction of 1dB (Theorems 5 and 6), and our computational experiments (Section 5) show that it actually solves the IDP model exactly (i.e., with no error) in the vast majority of cases. Under reasonable assumptions, the total running time of the GP algorithm is dominated by a single (not parametric) shortest path computation on a graph derived from the corner points of the walls in the floorplan. To sum up, GP is a fast, practical algorithm with tiny worst-case error bounds that are essentially zero in practice.

In addition, we provide a (slower but still fast) exact algorithm (Section 3), which we use to evaluate the approximation errors of the GP algorithm (Section 5). Our exact algorithm enumerates feasible paths that correspond to optimal paths in the parametric shortest path problem. The parametric problem takes two weights per edge e, a distance d_e and loss ℓ_e, and asks for the shortest paths w.r.t. edge weights $\ell_e + \lambda d_e$, for all values of the parameter $\lambda \in [0, \infty]$. The *parametric complexity* refers to the number of distinct such shortest paths. The values of λ for which the shortest path switches from one path to another are called *breakpoints*. Our exact algorithm runs in time proportional to the number of breakpoints, so bounds on this quantity are of interest.

Carstensen constructed examples where the number of breakpoints is $n^{\Omega(\log n)}$ [6]. Although our exact algorithm for IDP suffers from this worst-case lower bound, we show that it has polynomial-time smoothed complexity (Theorem 3), meaning that the bad examples with $n^{\Omega(\log n)}$ breakpoints are pathological and fragile. In our experiments, the average number of breakpoints is roughly 4 to 6 (Section 5). This means that our exact algorithm requires about 5 shortest path computations per measurement point that we wish to model, compared to GP running the equivalent of $O(1)$ shortest path computations to closely approximate the model for *all* measurement points *simultaneously*.

Additional Related Work. Wölfle and Landstorfer introduced the dominant path model in 1997 [18]. They suggested solving the model approximately (with no stated guarantees) using a heuristic grid-based method reminiscent of fast marching methods [16]. They later suggested a different algorithm based on searching an exponential-sized tree, with pruning heuristics to speed it up [19]. In 1998, they founded AWE Communications to develop a highly-successful software tool called WinProp, enabling wireless coverage prediction and network design. AWE was acquired by Altair in April 2016 [1]. AWE and affiliated researchers issued a blizzard of similar conference and workshop papers, of which [20] seems to be a canonical one for the IDP model.

Multiple papers [5, 8, 12, 15, 19, 20] compare predictions of the dominant path model against actual physical measurements. Plets et al. [15] stands out for its careful methodology and exposition. They report superb agreement between the IDP model and their empirical measurements on four buildings with diverse physical characteristics, with a mean absolute modeling error of 1.29 dB on the best building and 3.08dB on the worst.

In the algorithms literature, Gusfield gave an upper bound of $n^{O(\log n)}$ on the complexity of the parametric shortest path problem [9]. Nikolova et al. [14] gave an exact algorithm based on parametric shortest paths for a stochastic routing problem of maximizing the probability of arriving on time. Nikolova [13] later gave approximation algorithms for a general combinatorial optimization framework with several concave objective functions. At a high level, the approximation algorithms use geometric progressions similarly to our GP algorithm, but they do not apply to our problem due to a difference in objective functions and desired bounds. To wit, those algorithms provide multiplicative approximations while our algorithm gives an additive approximation requiring a different geometric analysis. In our context, a multiplicative approximation is meaningless, since path losses are measured on a log scale (dB), so the units would not even make sense for a multiplicative error. Our algorithm also provides a better tradeoff between the error and number of shortest path invocations. It remains an open question whether there exists a polynomial-time algorithm to solve the IDP exactly, but Carstensen's lower bound implies that we cannot hope to do so directly via our reduction to parametric shortest paths.

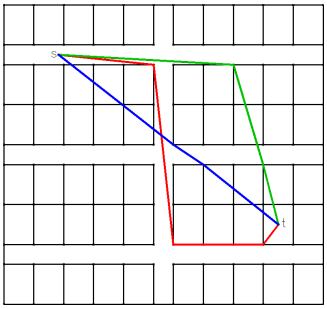

Figure 2: Paths on convex hull for office building

2 DOMINANT PATH MODEL & REDUCTION

Ideal freespace isotropic radio signal propagation is simple: the energy received by an antenna is proportional to $1/d^2$, where d is the distance from the transmitter to the receiving antenna. Real-world radio signals are non-isotropic, diffract around corners, and can penetrate through or reflect off of walls. The dominant path model is motivated by a simple empirical observation: although multiple propagation paths can contribute to the received signal strength at a measurement point t, usually the top 2 or 3 rays account for more than 95% of the energy [20]. Therefore, predicting the single path with smallest propagation loss can drive a good model of the total received signal strength. Even supposing that the dominant path contributes only 50% of the total energy, this would induce only a 3dB error, which is broadly within the range of accuracy that one achieves with more sophisticated models. The study by Plets et al. [15] validates the IDP model using just the single strongest path, reporting average errors within this range.

The IDP model focuses on RF propagation inside buildings, such as for WiFi. It models the path loss $\mathrm{PL}(P)$ along any path P, finds the path P from source (i.e., transmitter location) s to destination t that minimizes $\mathrm{PL}(P)$, and uses that number for the propagation loss from s to t. In this minimization, it considers only polygonal paths that change direction only at corner points of walls in the floorplan. Figure 2 shows examples of such paths, on a synthetic floorplan representing a generic office building. Other obstacles such as round pillars can be modeled as polygonal walls.

The path loss $\mathrm{PL}(P)$ along path P of length $d(P)$ can be broken into four components: (1) the (constant) unobstructed path loss at a reference distance d_0 (typically 1m), which can include an antenna gain, (2) a distance term based on the ratio $d(P)/d_0$, (3) penetration losses for passing through walls, and (4) diffraction losses for changing directions around obstacle corners. With the terms in this order, the path loss $\mathrm{PL}(P)$ (in dB) is:

$$\mathrm{PL}(P) = \mathrm{PL}_0 + 10\gamma \log_{10} \frac{d(P)}{d_0} + \sum_i \mathrm{WL}_i + \sum_j \mathrm{DL}_j \qquad (1)$$

where path P intersects a sequence of walls i and changes directions at a sequence of corners j. Since PL_0 is a constant (e.g., 40 dB at 1m [15]), we ignore it for the rest of this paper.

We assume a fixed signal frequency. The example parameter values stated next pertain to 2.4 GHz. The wall penetration loss WL_i is modeled as an input parameter for each wall, capturing material and thickness. Plets et al. [15] cite typical values: $\{2, 6, 7, 10\}$dB for thin walls of glass / layered drywall, wood, brick, or concrete (resp.), and $\{4, 15\}$dB for thick walls of glass or concrete (resp.). Justified by Snell's law, the modeled wall penetration loss does not depend on the angle of intersection. The diffraction loss at corner j is modeled as the deflection angle θ_j times a constant δ_j that depends on the wall material at corner j. The linearity follows from a thought experiment comparing diffraction around a sharp vs. beveled corner. Plets et al. use $\delta_j = 5\mathrm{dB}/90^\circ$ for layered drywall, and $17.5\mathrm{dB}/90^\circ$ for concrete [15].

Although earlier works (e.g., [20]) used various $\gamma > 2$ tuned for specific buildings, Plets et al. favor $\gamma = 2$ because it agrees well with their experiments, requires no tuning, and is easy to justify via Gauss's law. Therefore, we use $\gamma = 2$ in our computational experiments, although our theorems work for every choice of γ.

The dominant path model does *not* account for reflections. Some versions of the model incorporate other effects, such as "waveguiding" along tunnels or corridors [20], which would modify edge weights in our graph (Section 2.2). Following Plets et al. again, we eschew these extra knobs in our computational experiments. The heat map in Figure 1 depicts the model solution for a synthetic building representing a maze (Section 5). Discontinuities at each wall are obvious. A closer look reveals "shadows" behind each corner, as the diffraction loss accumulates for points whose dominant path bends around the corner.

2.1 Parametric shortest paths

This section defines the parametric shortest path problem in graphs, and the next section reduces the IDP model to it.

Our input is a graph $G = (V, E)$ where V is a set of nodes and E is a set of edges, along with two non-negative weights on each edge e: a distance d_e and a loss ℓ_e. In our context, d_e represents Euclidean distance and ℓ_e represents penetration and diffraction loss. Given any parameter $\lambda \geq 0$, we define a *hybrid* edge weight $h_\lambda(e) = \ell_e + \lambda d_e$. Given a source node $s \in V$ and target node $t \in V$, let P_λ be the shortest path w.r.t. weights h_λ and let $(d_\lambda, \ell_\lambda)$ denote

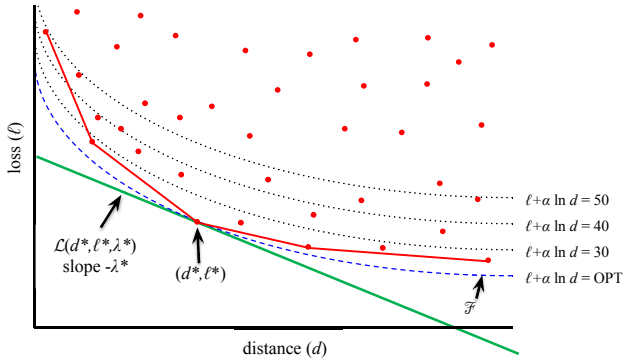

Figure 3: Each red point denotes an s-t path P, and the red solid line is their convex hull. The dotted lines are level sets of the function $\mathrm{PL}(P) = \ell(P) + \alpha \ln d(P)$, while $\mathcal{L}(d^*, \ell^*, \lambda^*)$ is the tangent to level set $\mathcal{F}$ at (d^*, ℓ^*), the dominant path.

its distance and loss. The *parametric s-t shortest path problem* is to compute the set of distinct paths $\{P_\lambda : \lambda \in [0, \infty]\}$, or, if we care only about their weights, then the corresponding set of distance-loss pairs: $\{(d_\lambda, \ell_\lambda) : \lambda \in [0, \infty]\}$. The *parametric single-source shortest path problem* is to solve the parametric s-t shortest path problem for a single s and all $t \in V$.

Figure 3 depicts the geometric meaning of minimizing the function h_λ over all s-t paths. The *extreme points* of the lower-left boundary of the convex hull of feasible paths correspond to the paths output by the parametric shortest path computation. The three paths in Figure 2 are paths corresponding to extreme points for the s-t pair shown. These extreme points partition the λ range $[0, \infty]$ into intervals, where each extreme point represents the shortest path w.r.t. h_λ for all λ in its interval. The endpoints of these intervals are *breakpoints* of λ, which are the (negative of the) slopes of the red line segments in Figure 3. These concepts are important for understanding both our GP algorithm (Section 4) and our experimental results (Section 5). For more intuition on the parametric shortest path problem and its relation to the IDP, see the full paper [3].

2.2 A graph representing all valid paths

We now construct a graph with weights d_e and ℓ_e on each edge e, capturing distance and (penetration + diffraction) loss of paths in the IDP model. We require:

(1) the distance $d(P)$ and loss $\ell(P)$ of path P in the IDP model equal $\sum_{e \in P} d_e$ and $\sum_{e \in P} \ell_e$, and
(2) every valid physical path P in the IDP model corresponds to a path in the graph, and vice versa.

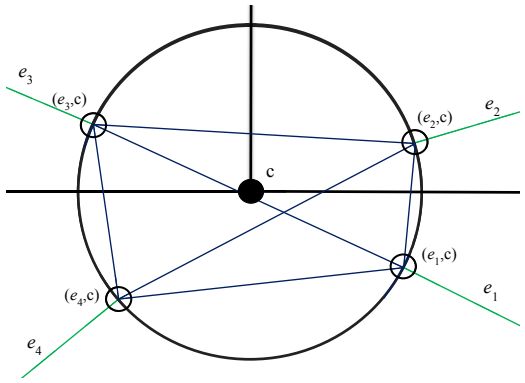

Figure 4: G2 corner

We construct such a graph in two phases: first a graph G_1 that encodes all relevant paths, plus their distances and wall penetration losses, then a related graph G_2 that also encodes diffraction losses.

Defining $G_1 = (V_1, E_1)$ is straightforward. The node set V_1 is the union of three sets: the single source $\{s\}$, the set of *destinations* T (aka *measurement* points), and the set of *corners* C. Here, T means the set of all points for which we wish to compute the dominant path, and C means the set of endpoints of wall segments in the floor plan. The set E_1 is the union of four sets of directed edges: all $s - C$, $s - T$, $C - C$, and $C - T$ pairs. There is no need for $T - T$ edges, since all intermediate points in the polygonal s-t paths considered in the IDP model are corner points. For each $e \in E_1$, let d_e be the Euclidean distance between its endpoints, and ℓ_e be the sum of the penetration losses of all walls it crosses.

There are two defects in G_1 that we must correct in our construction of G_2: it models neither the penetration losses at corners nor the diffraction losses. We correct this by "exploding" each corner node $c \in C$, replacing it with a new set of nodes, one for each incoming and outgoing edge e. These new nodes are *directed sockets* of G_1, i.e., ordered pairs (e, c) where $e \in E_1$ is incident to c in G_1. These new socket nodes are illustrated by the circles in Figure 4, labeled $(e_1, c), \ldots, (e_4, c)$. We now add a directed edge from each incoming socket to each outgoing socket at corner c, represented by the six blue edges inside the big circle in Figure 4.

An intra-corner edge $e \in E_2$ running from incoming socket (e_1, c) to outgoing socket (e_2, c) covers zero physical distance, so $d_e = 0$. The loss ℓ_e is the sum of a diffraction loss $\delta_c \theta_{e_1 e_2}$ (where $\theta_{e_1 e_2}$ is the physical angle between directed edges e_1 and e_2), and a penetration loss term. For the latter, we compute the total penetration loss for the walls incident at corner c encountered as we sweep either clockwise or counterclockwise from e_1 to e_2, and take the minimum.

There are two more subtleties. First, some edges of G_1 run from one end of a wall segment to the other. We represent these as two edges, one on each side of the wall, to enable correct penetration losses on the intra-corner edges of G_2. Second, sets of co-linear corner points are a common occurrence in buildings, so we cannot assume away their existence. It would be problematic to consider edges directly from one end of the line of corners to the other, because we would have to make a decision at each intermediate corner point about which side of the wall the edge lies on for that segment, thereby introducing an exponential number of parallel edges. Instead, we simply delete these edges, keeping only those connecting adjacent pairs along the line of corner points.

It is clear that every valid path P in the IDP model corresponds to a path in G_2, and vice versa. Moreover, $d(P)$ and $\ell(P)$ as defined by G_2 agree with the values assigned by the IDP model. In other words, we have set the edge weights so that (1) becomes

$$\mathrm{PL}(P) = \mathrm{PL}_0 + \ell(P) + 10\gamma \log_{10} d(P). \tag{2}$$

The dominant path is the one that minimizes $\ell(P) + 10\gamma \log_{10} d(P)$. For convenience, we convert to ln, plug in $\gamma = 2$, then define $\alpha = \frac{10\gamma}{\ln 10}\mathrm{dB} \approx 8.686\mathrm{dB}$, $f(d, \ell) = \ell + \alpha \ln d$, and $f(P) = f(d(P), \ell(P))$. Thus, the dominant path P is the one that minimizes $f(P)$.

2.3 Reduction to parametric shortest path

This theorem demonstrates that the dominant s-t path is one of the parametric shortest s-t paths in G_2.

Theorem 1. *Let P^* be the dominant s-t path, and define $d^* = d(P^*)$ and $\lambda^* = \frac{\alpha}{d^*}$. Then P^* also minimizes $h_{\lambda^*}(P) = \ell(P) + \lambda^* d(P)$.*

A proof appears in the full paper [3]. Figure 3 shows the crux: $\mathcal{L}(d^*, \ell^*, \lambda^*)$ lies below the level set $\mathcal{F}$, so (d^*, ℓ^*) minimizes h_{λ^*}.

3 EXACT CONVEX HULL

By the results of Section 2, finding the dominant s-t path reduces to a parametric s-t shortest path computation, or equivalently finding the lower left convex hull of feasible paths (Figure 3). Let $\mathrm{SP}(\lambda)$ denote the shortest path calculation w.r.t. edge weights h_λ.

Theorem 2. *The parametric s-t shortest path problem can be solved using $(2k+1)$ shortest s-t path computations, where k is the number of breakpoints.*

Since k is at most $n^{O(\log n)}$ [9], the parametric s-t shortest path problem can be solved using at most $n^{O(\log n)}$ s-t shortest path computations. The average number of breakpoints in our experiments is only about 5 (Section 5). This divergence between the worst case and practice can be explained by smoothed analysis: a formalization of the idea that the worst-case instances are rare and brittle, and in practice the algorithm encounters "good" instances, for which the number of breakpoints is small.

Theorem 3. *The exact IDP model can be solved in smoothed polynomial time.*

Proofs of Theorems 2 and 3 appear in the full paper [3].

4 GEOMETRIC PROGRESSION ALGORITHM

From Theorem 2, we can compute a single s-t dominant path efficiently if the number of breakpoints is small. However, if we wish to compute dominant paths from a single source s to *all* destinations then we can do much better, especially if we are willing to tolerate a small additive error. Our *geometric progression* (GP) algorithm does precisely this. We devote this section to its definition, then to analyzing its approximation error and practical time complexity.

Given fixed $r > 1$, $\lambda_0 > 0$, define a geometric sequence of values $\lambda_i = \lambda_0 r^i$, for $i \in \mathbb{Z}$. It is safe to think of r as 2. The geometric progression algorithm $GP(r, \lambda_0)$ runs $\mathrm{SP}(\lambda_i)$ for λ_i in some sufficiently wide range (specified later).[2] For each destination t, it outputs the best of the s-t paths it found, according to the real objective function f. Since our algorithm always outputs the path loss of some valid path, it never underestimates the optimal path loss. Let $d_\lambda(t)$ and $\ell_\lambda(t)$ denote the distance and loss of the s-t path $P_\lambda(t)$ computed by $\mathrm{SP}(\lambda)$, omitting the argument t where clear from context. For convenience, define $d_{\min} = d_\infty$ and $d_{\max} = d_0$, the lengths of the straight-line and min-loss s-t paths, noting that d_λ decreases in λ, while ℓ_λ increases.

For each t, Theorem 1 guarantees there is some λ such that $\mathrm{SP}(\lambda)$ finds the dominant s-t path. Although we don't know which λ that is, the $GP(r, \lambda_0)$ algorithm is guaranteed to use a nearby value, and this allows us to bound the error, as shown by Theorems 4, 5 and 6. In each theorem, (d^*, ℓ^*) denotes the optimal distance-loss for t.

Theorem 4. *If $\lambda^* = \alpha/d^*$, then $\mathrm{SP}(\lambda)$ with $\lambda = \beta\lambda^*$ yields a path loss $f(d_\lambda, \ell_\lambda) \le f(d^*, \ell^*) + \alpha(-1 + \beta - \ln \beta)$.*

It turns out that $\hat{\beta} := \frac{r \ln r}{r-1}$ is the worst value of β.

Theorem 5. *Algorithm $GP(r, \lambda_0)$ returns an s-t path P with $f(P) \le f(d^*, \ell^*) + \alpha(-1 + \frac{\ln r}{r-1} + \ln(r-1) - \ln\ln r)$.*

Theorem 6. *Set $\lambda_0 = r^u$ where u is drawn uniformly from $(0, 1)$. For each destination t, algorithm $GP(r, \lambda_0)$ returns an s-t path P with $E[f(P)] \le f(d^*, \ell^*) + \alpha(-\frac{1}{2}\ln r + \ln(r-1) - \ln\ln r)$.*

[2] Recall that Dijkstra's algorithm computes the shortest path from s to *all* targets simultaneously.

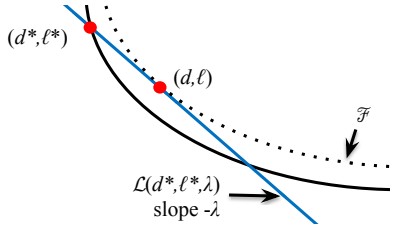

Figure 5: Approximation bound

We assume adversarial input, so the expectation in Theorem 6 is w.r.t. the algorithm's random choice of u. As we prove these theorems, our intermediate results will tell us what range of λ_i we must consider, and also allow us to prune G_2 before running each SP(λ_i) calculation. As a result, each measurement point appears in only a small number of these graphs, $O(1)$ in practice. We defer the proof of Theorem 6 to the full paper [3].

Proof of Theorem 4. Figure 5 illustrates the following geometric argument. SP(λ) minimizes h_λ, so $h_\lambda(d_\lambda, \ell_\lambda) \le h_\lambda(d^*, \ell^*)$. Hence, $(d_\lambda, \ell_\lambda)$ lies on or below the level set for h_λ through (d^*, ℓ^*), denoted $\mathcal{L}(d^*, \ell^*, \lambda)$. Among all such points, the one maximizing f lies at the point of tangency between $\mathcal{L}(d^*, \ell^*, \lambda)$ and some level curve $\mathcal{F}$ of f. This occurs at $d = \alpha/\lambda$, $\ell = \ell^* - \lambda(d - d^*)$. We upper bound $f(d_\lambda, \ell_\lambda)$ by $f(d, \ell)$:

$$\begin{aligned} f(d_\lambda, \ell_\lambda) \le f(d, \ell) &= f(d^*, \ell^*) + (f(d, \ell) - f(d^*, \ell^*)) \\ &= f(d^*, \ell^*) + (\ell - \ell^*) + \alpha \ln(d/d^*) \\ &= f(d^*, \ell^*) - \lambda(d - d^*) - \alpha \ln \beta \\ &= f(d^*, \ell^*) - \lambda(\alpha/\lambda - \alpha/\lambda^*) - \alpha \ln \beta \\ &= f(d^*, \ell^*) + \alpha(-1 + \beta - \ln \beta) \end{aligned}$$

using the fact that $d^* = \alpha/\lambda^*$, from Theorem 1. □

Proof of Theorem 5. Let $\lambda^* = \alpha/d^*$. Because consecutive λ_i are spaced by a multiplicative r, one of them (call it λ_{lo}) must land in the range $[\frac{1}{r}\lambda^*, \lambda^*]$, and one (call it λ_{hi}) must land in the range $[\lambda^*, r\lambda^*]$. Thus, $\lambda_{\text{hi}} = \beta\lambda^*$ and $\lambda_{\text{lo}} = \frac{\beta}{r}\lambda^*$ for some $\beta \in [1, r]$. Applying our upper bound from Theorem 4 twice, we see that the path P returned by $GP(r, \lambda_0)$ has error at most

$$\alpha(-1 + \min(\beta - \ln \beta, \beta/r - \ln \beta/r))$$

This min is maximized at $\beta = \hat{\beta} = \frac{r \ln r}{r-1}$, where the two terms in the min are both equal to $\frac{r \ln r}{r-1} - \ln(\frac{r \ln r}{r-1}) = \frac{\ln r}{r-1} + \ln(r-1) - \ln \ln r$. The desired bound follows. □

These bounds are quite small, even for generous values of r. For instance, when $r = 2$, the worst-case error from Theorem 5 is only 0.5182 dB, and our upper bound on the expected error is a mere 0.1732 dB. These errors are dwarfed by the validation errors of the model itself (around 1dB-5dB, as reported in Plets et al. [15]).

4.1 Practical considerations

We have one piece of unfinished business, which is to define the range of λ_i values for which $GP(r, \lambda_0)$ must run the SP(λ_i) computation. Further, we shall demonstrate some pruning tricks on G_2 that imply, under reasonable assumptions, that the total running time of all of our SP(λ_i) computations is $O(1)$ times the cost of running Dijkstra once on G_2. Finally, we discuss how to save time and memory by running Dijkstra on G_2 implicitly, while explicitly constructing only G_1 (and not G_2) in memory.

4.1.1 Pruning G_2. Let us fix a particular destination t, and set $I_{\lambda^*} = [\frac{1}{r}\hat{\beta}\lambda^*, \hat{\beta}\lambda^*]$. The proofs of Theorems 5 and 6 rely only on running SP(λ) for some $\lambda \in I_{\lambda^*}$. Although we do not know d^* or $\lambda^* = \alpha/d^*$, we do know that $\lambda^* \in [\frac{\alpha}{d_{\max}}, \frac{\alpha}{d_{\min}}]$. Therefore, if we run SP(λ_i) for each of the λ_i in

$$I(t) := \bigcup_{\lambda^* \in [\frac{\alpha}{d_{\max}}, \frac{\alpha}{d_{\min}}]} I_{\lambda^*} = \left[\frac{\alpha\hat{\beta}}{r d_{\max}}, \frac{\alpha\hat{\beta}}{d_{\min}}\right] \tag{3}$$

then we satisfy the error bounds in Theorems 5 and 6. Therefore, for each λ_i considered in algorithm $GP(r, \lambda_0)$, we need to include node t in G_2 only for the measurement points

$$M(\lambda_i) := \{t : \lambda_i \in I(t)\} = \left\{t : d_{\min}(t) \le \frac{\alpha\hat{\beta}}{\lambda_i} \text{ and } d_{\max}(t) \ge \frac{\alpha\hat{\beta}}{r\lambda_i}\right\}.$$

If $M(\lambda_i) = \emptyset$, then we do not have to run SP(λ_i) at all.

We can now answer the question that we deferred when first defining algorithm $GP(r, \lambda_0)$, namely: for which λ_i must we run SP(λ_i)? Let us define $D_{\min} = \min_t d_{\min}(t)$ and $D_{\max} = \max_t d_{\max}(t)$. Then we must try all λ_i in $[\frac{\alpha\hat{\beta}}{r D_{\max}}, \frac{\alpha\hat{\beta}}{D_{\min}}]$.

The multiplicative width of $I(t)$ is only $r\frac{d_{\max}}{d_{\min}}$, so in expectation, each destination t is pruned from G_2 for all but $\log_r \frac{r d_{\max}(t)}{d_{\min}(t)}$ values of λ_i. Recall that $d_{\min}(t)$ is merely the straight-line distance from s to t, whereas $d_{\max}(t)$ is the distance along the s-t path $P_0(t)$ with lowest (penetration + diffraction) losses. The diffraction losses are relatively high compared to the penetration losses, e.g., for drywall, a 90° turn costs the same as penetrating 2.5 walls (Section 2). Therefore, we would expect that path P_0 does not bend too much, and therefore $\frac{d_{\max}(t)}{d_{\min}(t)}$ will be fairly small in practice, typically less than 2. In this case, if using $r = 2$, then we include most destinations t in only one or two of the SP(λ) computations.

We can also prune some of the corner points from G_2. If the distance $d(s, c) + d(c, t)$ from s straight to corner point $c \in C$ straight

to t exceeds $d_{\max}(t)$, then we need not consider any s-t paths that go through c, because they will be both longer than $P_0(t)$ and have equal or greater loss than $P_0(t)$ (which has minimum loss over all s-t paths). In this case, we prune the edge (c, t) from G_2. If this condition holds for all $t \in M(\lambda_i)$, we prune c from G_2 entirely.

Conceptually, our sequence of SP(λ_i) computations is performed on a single graph, G_2. However, the pruning operations that we just discussed shrink it greatly for most values of λ_i, because of the geometry. Let us sweep λ downward from $\frac{\alpha\hat{\beta}}{D_{\min}}$ to $\frac{\alpha\hat{\beta}}{rD_{\max}}$. Initially, only the nodes close to the source s remain unpruned. Each time we divide λ by r, the outer radius of the annulus of measurement points defining $M(\lambda)$ grows by a factor of r. Assuming the ratio $\frac{d_{\max}(t)}{d_{\min}(t)}$ is $O(1)$, then the inner radius of this annulus also grows by roughly a factor of r. In the typical case, the set of measurement points is a uniform grid, which means that $|M(\lambda)|$ grows by roughly a factor of r^2. If the corners are also spaced relatively uniformly, then the number of unpruned corners also grows by roughly r^2. Therefore, the set of relevant unpruned $C - C$ and $C - T$ edges grows by roughly r^4. Therefore, the total cost of all of the SP(λ_i) computations is dominated by the ones at the end of the process (small λ_i), where the pruned version of G_2 is the largest, and by SP(0), which is used to compute $d_{\max}(t)$ for all t and so must run on the full G_2.

4.1.2 Running Dijkstra on G_2 implicitly. Recall Dijkstra's shortest path algorithm [11]. We maintain a distance label L_v on each node v, initialized to 0 for s and ∞ for all other nodes, with all labels *active*. At each step, we select the smallest active label, make it inactive (aka *finalize* it), and *relax* each of its outgoing edges $e = (v, w)$, namely $L_w \leftarrow \min(L_w, L_v + w_e)$. Once the last label is finalized, L_v is the weight of the shortest path from s to v.

Recall that most of the nodes in G_2 are sockets (e, c) from G_1, where $c \in C$ is a corner point with incident edge e, and most of the edges of G_2 are the intra-corner edges from each incoming socket to each outgoing socket at c. We can save a huge amount of memory by explicitly constructing and storing only G_1, and running Dijkstra implicitly on G_2. To do this, we maintain our distance labels L on the sockets of G_1. When we finalize the label of an incoming socket (e, c), we relax all of its outgoing intra-corner edges to the outgoing sockets (e', c). The cost of each such edge depends on just three things: (1) the diffraction angle between e and e', (2) the sector of e, and (3) the sector of e', where *sector* refers to the directions between two consecutive walls meeting at c. Figure 4 shows 3 sectors around corner c. The pair of sectors determines the penetration loss and the deflection angle determines the diffraction loss.

This explains how to run Dijkstra without ever storing G_2. Better yet, we can avoid the vast majority of no-op relaxations (i.e., ones that do not actually update their label). This is because, for all outgoing sockets (e', c) within a given sector, the penetration loss from (e, c) is the same, and the diffraction loss, plotted as a function of the angle θ, is a line with constant slope equal to $\pm\delta_c$, the diffraction constant at corner c. Thus, we can picture each of the finalized incoming sockets (e, c) at corner c as inducing either a V-shape (for its own and its opposite sector) or a line (for all other sectors), representing the implied path weight to a hypothetical outgoing socket at angle θ. The actual Dijkstra label will be the minimum of these lines and V-shapes.

To actually perform the relaxation for a newly-finalized incoming socket (e, c), we start in the opposite sector at angle 0, the bottom of the V, and march through the outgoing sockets in clockwise order to 180°, then do it again counterclockwise. If we ever encounter an angle at which our implied label exceeds the existing label (aka a no-op relaxation), we know that our line is dominated for the rest of that sector, since we are increasing at rate δ_c and all other lines are increasing or decreasing at that same rate. We then pick up at the next sector, where we have a chance again because the vertical offset of each line is different (from the differing penetration losses, depending on the sector of the corresponding incoming socket). Therefore, the number of no-op Dijkstra relaxations that we must perform is bounded by the number of sectors + 2 (since the 0° and 180° sectors each count twice).

5 EXPERIMENTS

Datasets: For our experiments, we consider two types of artificial "buildings." These are not meant to replicate real buildings, but rather just to exhibit some properties of our algorithms. The first type are ten random "maze" buildings, like the one shown in Figure 1. These are formed by taking a 20x20 grid graph, removing a random spanning tree, and then taking the planar dual, leaving a 20x20 connected maze of 3m x 3m cells. This gives 60m x 60m buildings, with 441 corner points, 441 walls, and 3600 measurement points (on a 1m grid). The second building is an artificial office building, like the one shown in Figure 2, to contrast with the random mazes. Although this is not a real office building, it does provide a check that the experimental results are not purely an artifact of the random mazes. It consists of a very regular grid of 3m x 4m offices connected by 2m wide hallways, with 12 rows of 20 offices each (where Figure 2 shows just a portion with 6 rows of 10 offices). This office is 62m x 60m, with 418 corner points, 658 walls, and 3720 measurement points (on a 1m grid). For both types

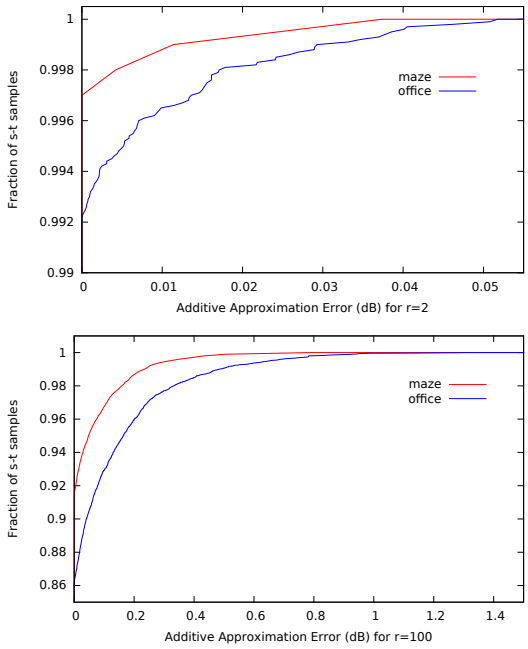

Figure 6: Error distributions for $r = 2$ and $r = 100$

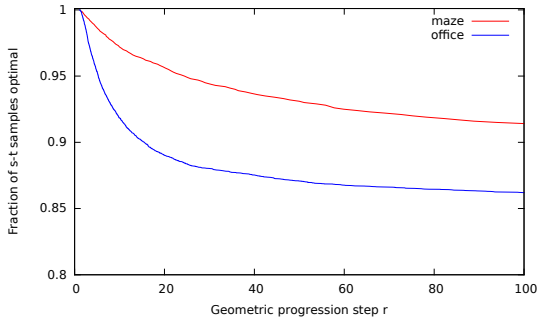

Figure 7: Frequency of identification of dominant *s*-*t* path

of artifical buildings, the exterior walls are concrete (with 15dB penetration loss), the interior walls are layered drywall (with 2dB penetration loss), and the diffraction loss at all corners is 5dB/90° (loss values from Plets et al. [15]). Of course, our algorithms work just fine with different loss parameters for each wall and corner, but uniform values are most easily justified.

Approximation errors: First, we consider the approximation error of the GP algorithm from Section 4. For each of the ten random mazes, we sampled 1000 random source-destination pairs, and for the office building, we sampled 10,000 random pairs. For each pair we computed the full convex hull of parametric shortest paths (Section 3). Even though Carstensen [6] gives a worst-case lower bound of $n^{\Omega(\log n)}$ for the number of extreme points on the convex hull, the worst case we encountered was 19 extreme points, and the mean was only 4.2 for the mazes, and 5.5 for the office building.

Based on these convex hulls, we compute the exact solution to the IDP model, which allows us to compute the expected error for each source/destination pair in the GP approximation algorithm. All expectations are w.r.t. the random choice of λ_0 in $GP(r, \lambda_0)$. Figure 6 shows the error distributions for $r = 2$ and $r = 100$. Recall that our errors are one-sided: we can only overestimate the path loss of the dominant path. The actual expected errors are much better than the bounds from Theorems 5 and 6. For instance, with $r = 2$, the GP algorithm failed to find the exact dominant *s*-*t* path for fewer than 0.8% of the *s*-*t* pairs, and the error exceeded 0.06dB for *none* of them. These approximation errors are insignificant compared to the model validation errors of 1dB to 5dB reported by Plets et al. [15]. Even for the extreme case of $r = 100$, for which Theorem 6 gives an expected error bound of 6.6dB, the worst observed error is only 1.5dB and 99% of the *s*-*t* pairs have error below 0.6dB. There is never any reason to use such a large value of r; we show it just to emphasize that our results are extremely robust to r.

To understand why the geometric progression algorithm actually finds the dominant *s*-*t* path so frequently, consider the two breakpoints λ_{lo} and λ_{hi} corresponding to the two segments of the convex hull adjacent to the extreme point representing the dominant *s*-*t* path (Figure 3). The dominant *s*-*t* path will be returned by $\text{SP}(\lambda)$ for every value of $\lambda \in (\lambda_{\text{lo}}, \lambda_{\text{hi}})$. In particular, if $\lambda_{\text{hi}}/\lambda_{\text{lo}} > r$, then the geometric progression is *guaranteed* to have $\lambda_i \in (\lambda_{\text{lo}}, \lambda_{\text{hi}})$ for some i, and hence find the dominant *s*-*t* path. Figure 7 shows, as a function of r, how often we are guaranteed to find the dominant *s*-*t* path, i.e., what fraction of our sampled *s*-*t* pairs satisfy $\lambda_{\text{hi}}/\lambda_{\text{lo}} > r$.

Pruning G_2 : As observed in Section 4.1.1, a measurement point t needs to be included when running $\text{SP}(\lambda_i)$ only for each $\lambda_i \in I(t)$, so is pruned from G_2 for all but $\log_r \frac{r d_{\max}(t)}{d_{\min}(t)}$ values of λ_i (in expectation). To evaluate how effective this pruning was, we considered $r = 2$ for ten random choices of s and a 1m grid of measurement points for each of ten random maze buildings and the office building. For the random maze buildings, the expected number of $\text{SP}(\lambda_i)$ computations that left the average measurement point unpruned was only 1.06, and the maximum expectation we encountered over all measurement points was only 2.14. For the office building, the long straight hallways result in higher $\frac{d_{\max}(t)}{d_{\min}(t)}$ ratios, but still the average number of $\text{SP}(\lambda_i)$ a measurement point was included in was only 1.29, and the maximum we encountered was only 2.69. Hence, the total complexity of the full $GP(2, \lambda_0)$ algorithm is very close to the complexity of running two Dijkstra computations on

the unpruned version of G_2: one with $\lambda = 0$ to compute $d_{\max}(t)$ for all t, and the sequence of Dijkstra runs on pruned versions of G_2 add up to about one additional Dijkstra on the full G_2.

Implicit G_2 savings: A key implementation detail is to avoid no-op Dijkstra edge relaxations on the implicit representation of G_2 (Section 4.1.2). In the run that generated the heat map in Figure 1, roughly 99% of the relaxations were no-ops, so this trick allowed us to perform only 2.18×10^8 relaxation steps rather than 2.87×10^{10}.

Running time: For the mazes, building G_2 took 3.0 CPU sec, and generating a heat map from a single source took 1.0sec. For the office, building G_2 took 2.9sec, and a single heat map took 1.3sec. The experiments were run using a single thread on a 3.6GHz Intel Xeon E5-1650v4 CPU.

6 FUTURE WORK

This paper focuses on algorithm design, not algorithm engineering. Although our prototype implementation is reasonable, it has not been highly engineered for speed. We could accelerate it by pruning edges of G_1 above some loss threshold, handling measurement points outside the main Dijkstra loop and priority queue, tuning data structures, cache optimization, etc. After such improvements, a careful "horserace" running time comparison against tree-search dominant path codes might be appropriate. This paper instead focuses on proving the GP algorithm's viability, owing to its already-fast running time and superb fidelity to the exact IDP model.

For simplicity, we focused on the 2D indoor dominant path model, but the outdoor and mixed models are also important. It would be interesting to apply the GP algorithm to these models, and also to 3D models. Finally, we hope that our methods will be integrated into wireless nework planning tools, to support AP placement optimization as described in Section 1.

7 ACKNOWLEDGMENTS

This paper is dedicated to our late co-author David S. Johnson, an exemplary friend, mentor, and human being. Evdokia Nikolova's and Ger Yang's work was partially supported by NSF grants CCF-1216103, CCF-1331863, CCF-1350823 and CCF-1733832. Mikkel Thorup's research is supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research and by Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.

REFERENCES

[1] Altair. 2016. Altair Acquires WinProp and AWE Communications GmbH. Retrieved March 16, 2017 from http://www.altair.com/newsdetail.aspx?news_id=11249&news_country=en-US

[2] Altair. 2016. WinProp Software for Wave Propagation and Radio Planning. Retrieved March 16, 2017 from http://www.altairhyperworks.com/product/FEKO/WinProp-Propagation-Modeling

[3] David Applegate, Aaron Archer, David S. Johnson, Evdokia Nikolova, Mikkel Thorup, and Ger Yang. 2017. Wireless coverage prediction via parametric shortest paths. *arXiv preprint arXiv:1805.06420* (2017).

[4] Alexander Aschrafi, Philipp Wertz, Michael Layh, Friedrich M. Landstorfer, Gerd Wölfle, and René Wahl. 2006. Impact of Building Database Accuracy on Predictions with Wave Propagation Models in Urban Scenarios. In *Proceedings of IEEE Vehicular Technology Conference, VTC*. 2681–2685.

[5] Ahmad Awada, Ekkehard Lang, Olaf Renner, Karl-Josef Friederichs, Swen Petersen, Kerstin Pfaffinger, Benjamin Lembke, and Roland Brugger. 2016. Field Trial of LTE eMBMS Network for TV Distribution: Experimental Results and Analysis. *IEEE Transactions on Broadcasting* (2016).

[6] Patricia J. Carstensen. 1983. *The complexity of some problems in parametric linear and combinatorial programming*. Ph.D. Dissertation. Mathematics Dept., U. of Michigan, Ann Arbor, Michigan.

[7] CelPlan Wireless Global Solutions. 2017. CellTrace: Indoor Prediction Tools Based on Ray Tracing. Retrieved December 18, 2017 from http://www.celplan.com/products/celltrace.asp

[8] Leonard Grokop, Chirag Patel, Mehmet Yavuz, and Sanjiv Nanda. 2010. SimTown: RF Propagation Models for Urban Femtocell Environments. *Qualcomm White Paper* (2010).

[9] Daniel Mier Gusfield. 1980. *Sensitivity Analysis for Combinatorial Optimization*. Ph.D. Dissertation. University of California, Berkeley. AAI8029414.

[10] iBwave. 2007. iBwave Integrates AWE Communications Propagation Module into iBwave Design. Retrieved December 18, 2017 from http://www.ibwave.com/media-and-events/press-releases/215/ibwave-integrates-awe-communications-propagation-module-into-ibwave-design

[11] Jon Kleinberg and Éva Tardos. 2005. *Algorithm Design*. Pearson.

[12] Farhad Meshkati, Yi Jiang, Lenny Grokop, Sumeeth Nagaraja, Mehmet Yavuz, and Sanjiv Nanda. 2009. Mobility and capacity offload for 3G UMTS femtocells. In *IEEE GLOBECOM 2009*. 1–7.

[13] Evdokia Nikolova. 2010. Approximation algorithms for reliable stochastic combinatorial optimization. In *APPROX-RANDOM*. Springer-Verlag, 338–351.

[14] Evdokia Nikolova, Jonathan A. Kelner, Matthew Brand, and Michael Mitzenmacher. 2006. Stochastic Shortest Paths via Quasi-convex Maximization. In *Lecture Notes in Computer Science 4168 (ESA 2006)*. Springer-Verlag, 552–563.

[15] David Plets, Wout Joseph, Kris Vanhecke, Emmeric Tanghe, and Luc Martens. 2012. Coverage prediction and optimization algorithms for indoor environments. *EURASIP J. Wireless Comm. and Networking* 2012 (2012), 123.

[16] James A. Sethian. 1999. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

[17] The Internet Archive. 2008. WinProp Customers. Retrieved December 14, 2017 from https://web.archive.org/web/20080509165734/http://www.awe-communications.com/Company/CustomersCommercial.html

[18] Gerd Wölfle and Friedrich M. Landstorfer. 1997. Field strength prediction with dominant paths and neural networks for indoor mobile communication. In *MIOP, Mikrowellen und Optronik, 9. Kongreßmesse für Hochfrequenztechnik, Funkkommunikation und elektromagnetische Verträglichkeit,*. 216–220.

[19] Gerd Wölfle and Friedrich M. Landstorfer. 1998. Dominant paths for the field strength prediction. In *Proceedings of the 48th IEEE Vehicular Technology Conference (VTC)*. Ottawa, 552–556.

[20] Gerd Wölfle, Rene Wahl, Philipp Wertz, Pascal Wildbolz, and Friedrich Landstorfer. 2005. Dominant path prediction model for indoor scenarios. In *German Microwave Conference (GeMIC)*. Ulm, Germany, 176–179.