Computing Optimized Path Integrals for Knapsack Feasibility

Endric Daues

Fu Foundation School of Engineering and Applied Science, Columbia University, USA, ed2691@columbia.edu

Ulf Friedrich

Operations Research, Technical University of Munich, Germany, ulf.friedrich@tum.de

Abtstract: A generating function technique for solving integer programs via the evaluation of complex path integrals is discussed from a theoretical and computational perspective. Applying the method to solve knapsack feasibility problems, it is demonstrated how the presented numerical integration algorithm benefits from pre-optimizing the path of integration. After discussing the algorithmic set-up in detail, a numerical study is implemented to evaluate the computational performance of the pre-optimized integration method and the algorithmic parameters are tuned to a set of knapsack instances. The goal is to highlight the method's computational advantage for hard knapsack instances with large coefficients and high numbers of feasible solutions.

Keywords: Integer Programming, Cauchy's Integral Formula, Path Optimization, Generating Functions, Computational Study.

1. Introduction

We demonstrate how integer programs (IP), specifically knapsack instances, can be solved with the help of complex analysis by transforming the feasibility problem to a numerical integration problem. While this is certainly not the first generating function approach for counting solutions, feasibility checking, or optimization in the context of integer programming, cf. the references discussed below, our approach is self-contained, streamlined, and fully implemented in Python on `https://github.com/endric-daues/ipcauchy`.

Following the exposition of the theory, we discuss challenges in the implementation and how a careful choice of parameters improves the computational performance. Adapting an idea from numerical complex analysis discussed in Bornemann and Wechslberger (2013), we propose optimizing the path of integration as a pre-solving step. In the following, we demonstrate how even a simple Python code using a standard quadrature library is capable of counting solutions to hard knapsack instances taken from Pisinger (2005) after a careful parameter tuning. Thereby, we prove the capability of the method and the potential for future research in numerical quadrature methods for IP.

This work is structured into four sections: We continue the introduction with a detailed discussion of related work and prerequisites. In Section 2, we present the theoretical framework for the method. For the sake of a condensed presentation, we formulate all results specifically for the knapsack problems considered later in the computational study. Section 3 contains the main improvement over existing methods, namely the comparative discussion of several possible paths of integration and, in particular, a routine to find a *shortest path of integration*. We continue with a detailed numerical study and parameter tuning for the algorithm, which we discuss in Section 4, before we conclude by pointing to future research ideas.

1.1. Related Work

Some of the most influential results on generating functions are due to Barvinok (1994a,b) and Barvinok and Pommersheim (1999) who develop and study the concept of *short rational functions* in a series of papers. In particular, it is shown that the generating function $\sum_{m\in P\cap\mathbb{Z}^n} z^m$ for the integer points of a polyhedron $P \subset \mathbb{R}^n$ can be represented in a compact form. In addition, this representation can be computed in polynomial time when the dimension is fixed, see Barvinok (1994a).

Applying and improving the above results, several authors use generating functions of the Barvinok type for solving integer programs or, more specifically, counting the solutions to IP, see Brion and Vergne (1997a), De Loera et al. (2004, 2005), Köppe et al. (2004), and De Loera (2005). On the computational side, for example the software package LattE, see Köppe (2018), provides algorithms for counting and solving optimization problems based on this theory, see also Köppe (2007) for an extension.

In contrast to the above, we study a generating function H_a which is not of the Barvinok type and is introduced directly via its limit (1). The generating function property of this weighted geometric series for the number of feasible solutions to the system $Ax = b, x \in \mathbb{Z}^n_+$ has already been observed by Euler (1770), cf. also Beck (2004) for a detailed introduction.

While a major theoretical result for the generating function $\sum_{m\in P\cap\mathbb{Z}^n} z^m$ is the Barvinok algorithm for computing the short rational function representation efficiently, we face the opposite problem: H_a is already given in a compact form and we are left with the task of recovering its algebraic structure. In Lasserre (2009) a comprehensive discussion of this dual perspective is given, see also Lasserre and Zeron (2001, 2007). Several authors have studied what we call the weighted geometric series in the context of IP and related fields, such as

Lasserre and Zeron (2002) for solving knapsack problems or Beck and Robins (2002), Beck (2004), and Lasserre and Zeron (2003, 2007) in the context of counting solutions.

Our algorithm and the numerical study for it are based on the idea of computing a certain coefficient in the weighted geometric series via the Cauchy formula for complex path integrals. This connection was observed by Brion and Vergne (1997b), Beck (2000), Lasserre and Zeron (2002) and used for computations in, e.g., Beck and Pixton (2003). In addition, the methods have been applied successfully to special applications, such as networks flows (Baldoni-Silva et al. 2004) and computational number theory (Baldoni et al. 2014). Recently, Hirai et al. (2020) made important contributions to the algorithmic complexity of counting integer points via the evaluation of circular paths integrals and applied their findings to hypergraph matchings.

Previous work, however, does not make use of the potential to tune the parameters of an auxiliary shortest path problem to optimize the complex path integral for a set of similar knapsack instances at hand. The properties that define a similarity between instances are discussed in Section 2.2 and then applied to the practical parameter tuning in Section 4. Our computational study shows how the pre-processing and parameter tuning of the path of integration significantly improves the computational performance of our method and potentially also of the other integration methods mentioned above.

2. Description of the Method

In what follows, we formulate our method for knapsack problems with positive coefficients, which are our object of study in Sections 3 and 4. This makes the presentation considerably easier and self-contained. However, the results can, in principle, be generalized to multi-row IP with non-negative input using several complex variables. The reader is referred to Friedrich (2016, 2020) for a detailed description of the theoretical framework for this extension.

We consider a feasibility instance of an unbounded knapsack problem

$$a^T x = b,\ x \in \mathbb{Z}_+^n,$$

with $a \in \mathbb{N}^n$ and $b \in \mathbb{Z}_+$, i.e., the decision problem of whether there is at least one non-negative, integer solution to the equation $a^T x = b$.

First, we focus only on the coefficient vector. Using a as a parameter, we define the complex function H_a on the open unit disk $D := \{z \in \mathbb{C} : |z| < 1\}$, given by

$$H_a : D \to \mathbb{C},\ z \mapsto \prod_{j=1}^{n} \frac{1}{1 - z^{a_j}}.$$

Because all coefficients a_j are positive, we have $|z^{a_j}| < 1$ for all $j \in \{1, \ldots, n\}$. This shows that H_a is well-defined. Clearly, the function H_a is the product of n geometric series, weighted with the coefficients of the vector a, i.e.,

$$H_a(z) = \prod_{j=1}^{n} \frac{1}{1 - z^{a_j}} = \prod_{j=1}^{n} \sum_{\nu_j=0}^{\infty} z^{a_j \nu_j}. \tag{1}$$

We call H_a the *weighted geometric series* with weight vector a. Note that the weighted geometric series is closely related to the L-transforms studied by Lasserre and Zeron (2007) and others. Verifying basic convergence properties for complex power series, we know that H_a converges absolutely on D and is thus a holomorphic function on the unit disk, see Rudin (1987). Moreover, because of the absolute convergence, we can interchange summation and multiplication in (1) to obtain $H_a(z) = \sum_{\nu \in \mathbb{Z}_+^n} z^{a^T \nu}$ on the unit disk D. This is a sum over a multi-index $\nu \in \mathbb{Z}_+^n$, which does not have a natural order of summation. However, because of the absolute convergence of H_a, the series convergences unconditionally, i.e., we may indeed use an arbitrary order of summation without changing the limit.

In particular, we can re-arrange the series by grouping all monomials with the same value of the exponent $a^T v$.

$$H_a(z) = \sum_{\nu \in \mathbb{Z}_+^n} z^{a^T \nu} = \sum_{k=0}^{\infty} \eta_k z^k, \text{ with } \eta_k = \eta_k(a) = \left| \left\{ \nu \in \mathbb{Z}_+^n : a^T \nu = k \right\} \right|. \tag{2}$$

It follows that η_k are non-negative integers. This proves that H_a is the generating function for the number of solutions of the system $a^T x = k, x \in \mathbb{Z}_+^n$. Setting $k = b$ for the right hand side b of the original knapsack equation, we have the following result.

COROLLARY 1. *For $a \in \mathbb{N}^n$ and $b \in \mathbb{Z}_+$ the problem $a^T x = b, x \in \mathbb{Z}_+^n$, is feasible if and only if $\eta_b \geq 1$ in (2). If the problem is feasible, it has η_b distinct solutions.*

In the general IP setting, we replace the vector $a \in \mathbb{N}^n$ with an integer matrix $A \in \mathbb{Z}_+^{m \times n}$ and consider a generating function in m complex variables $z_1, \ldots, z_m$. Under mild assumptions on A, we can apply multivariate techniques to give an explicit generating function for the solutions to $Ax = b, x \in \mathbb{Z}_+^n$ in the form of a power series in m variables, see Beck (2000), Lasserre and Zeron (2002), and Friedrich (2016).

2.1. Path Integrals

We want to compute the coefficient η_b without expanding the power series in (2). For this task we apply a classic result from complex analysis, known as *Cauchy's integral formula*, see Rudin (1987, Th. 10.15). It expresses the value of a holomorphic function $f(z)$, and the coefficients in its series representation, as a path integral of f that "circles" the point z. Proposition 1 below is just a reformulation of this textbook result for the function H_a.

PROPOSITION 1 **(Cauchy's Integral Formula for H_a).** *Let $a \in \mathbb{N}^n$ and let γ be a closed path in D with $\operatorname{Ind}_\gamma(0) \neq 0$. Then, for every $b \in \mathbb{Z}_+$ the coefficient η_b in (2) is given by*

$$\eta_b = \frac{1}{2\pi i \operatorname{Ind}_\gamma(0)} \int_\gamma \frac{H_a(\zeta)}{\zeta^{b+1}} \, \mathrm{d}\zeta. \tag{3}$$

Above, a *path* is a piece-wise continuously differentiable curve $\gamma : [a, b] \to \mathbb{C}$ in the complex plane and i is the complex unit. The *range* $[\gamma] \coloneqq \{z \in \mathbb{C} : z = \gamma(t) \text{ for some } t \in [a, b]\}$ of a path is defined as the set of all points in the image of γ, the *index* (or winding number) is given by $\operatorname{Ind}_\gamma(z) = \frac{1}{2\pi i} \int_\gamma \frac{1}{\zeta - z} \, \mathrm{d}\zeta$ for all $z \in \mathbb{C} \setminus [\gamma]$. The reader is referred to Rudin (1987, Ch. 10) for an introduction to path integrals in the complex plane. In particular, the index $\operatorname{Ind}_\gamma(z)$ is an integer and "counts" how often γ circles the point z in positive direction, see Rudin (1987, Th. 10.10).

We apply Proposition 1 to formulate a feasibility criterion for knapsack instances. By Corollary 1, the system $a^T x = b,\ x \in \mathbb{Z}_+^n$ is feasible if and only if $\eta_b \geq 0$, which is equivalent to

$$\int_\gamma \frac{H_a(\zeta)}{i\zeta^{b+1}} \, \mathrm{d}\zeta \geq 2\pi \operatorname{Ind}_\gamma(0) \tag{4}$$

in the setting of Proposition 1.

While we can use (3) to count the number of solutions to a knapsack instance, thus connecting our work immediately to Baldoni-Silva et al. (2004), Beck (2000), Lasserre and Zeron (2007), and others, the reformulation in (4) focuses on the feasibility setting. In particular, it is not necessary to compute the *exact* value for the integral in (4) to decide whether the problem is feasible or not. This is especially relevant if error bounds for the used numerical integration method are known. This means that, while in the later numerical study we look to maintain the error bounds at the nearest integer, these bounds can be relaxed when checking feasibility. In addition, using the general binary search idea from Grötschel et al.

(1988, Sec. 1.2), equation (4) can be used to solve knapsack *optimization* problems via several feasibility instances.

However, in order to formulate the binary search, we need to generalize (4) for instances with at least two rows of constraints: one for the original knapsack constraint and one to bound the objective function in the search, see Lasserre and Zeron (2002) for this two-dimensional setting without path adaptation. The binary search with path adaptation for general IP instances is discussed in Friedrich (2020). It uses *multi-paths* instead of γ above and Proposition 1 is generalized using a multi-path version of Cauchy's formula.

2.2. Dominant Poles

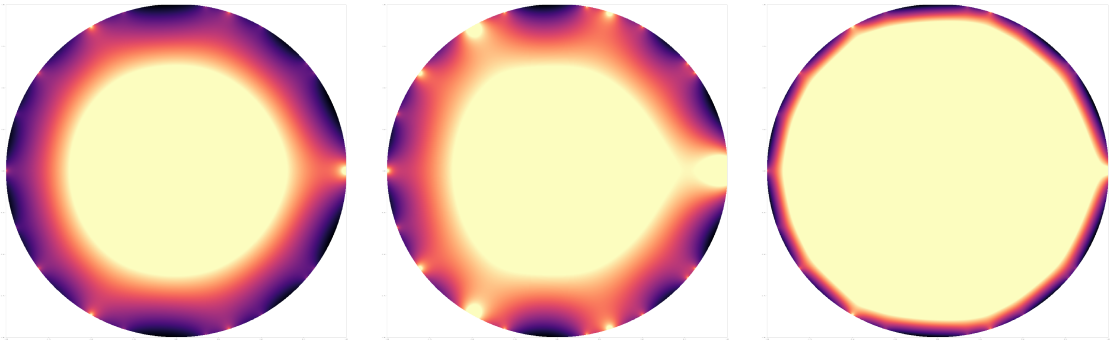
Before we discuss the path of integration, we turn to the properties of the integrand in Proposition 1, specifically its singularities in and near its domain. We know the values of the integrand within the unit disk are dependent on the coefficient vector a and the right hand side value b. Figure 1 visualizes the integrand as a heat map within the unit disk. The poles are located at the centers of the bright areas, where the integrand values tend to infinity.

The integrand in (3) presents singularities at the origin $\zeta = 0$ and wherever $H_a(\zeta) = 0$. The order of the pole at the origin is equal to $b+1$ as seen in (3). In Figure 1 (a) and (b) this pole has order 22, and in Figure 1 (c) it has order 101.

From (1) we see that the singularities of H_a are at $1-\zeta^{a_j} = 0$ and, thus, their locations are given by the a_j-th roots of unity for all a_j in the coefficient vector. We say a pole *dominates* a set of poles when its order is greater than or equal to the order of all other poles in the set. Since for every a_j in the coefficient vector, one of its roots of unity is located at $\zeta = 1$, we know that the pole at $\zeta = 1$ dominates the poles on the boundary of the unit disk. Looking closely at (1), we can say more about the locations of its poles by grouping and multiplying equal terms. An a_j-th root of unity is *unique* in a if it is not also a root of unity for another coefficient a_k in a, $k \neq j$. In particular, if all the coefficients in a are co-prime, the roots are unique in a and the order of all the poles on the boundary of D is one, except for the pole at $\zeta = 1$.

The order of a pole on the boundary of the unit disk is therefore given by the number of elements of a for which the roots of unity coincide. Since $\zeta = 1$ is a root of $1-\zeta^{a_j}$ for all a_j in a, its order is given by $|a| = n$, as stated above.

As an example, let p be a prime number. The order of the poles at the p-th roots of unity are equal to the number of coefficients a_j for which $a_j \equiv 0 \mod p$. If p is not prime, we



(a) (b) (c)

Figure 1 **Change in the size of poles in the unit disk for the η_b integral for instances (a)** $a = [3, 5, 6, 9, 10], b = 21$**, (b)** $a = [3, 3, 3, 5, 5, 5, 6, 6, 6, 9, 9, 9, 10, 10, 10], b = 21$ **(c)** $a = [3, 5, 6, 9, 10], b = 100$

look at its prime factorization and the roots of unity of the elements of this factorization separately. We illustrate these algebraic properties for the example in Figure 1. In (a) all the poles at the 3rd roots of unity, except for the pole at $z = 1$, have order 3 since 3 is a factor of 3, 6, and 9, all of which are coefficients of a. The 6th roots of unity, however, have order 3 wherever they are also 3rd roots of unity, and order 2 wherever they are also 2nd roots of unity. Here, the pole at $\zeta = 1$ has order 5 since $|a| = 5$. In Figure 1 (b) the poles on the 3rd roots of unity have order 9 and the pole at $\zeta = 1$ has order 15.

As this section demonstrates, understanding the locations, orders and symmetries of the poles of (3) requires some deeper algebraic reasoning and structured research in this direction has been undertaken by Beck (2000, 2004), Lasserre and Zeron (2007), and in Hirai et al. (2020). We look to avoid such reasoning in our approach and instead address the problem from a numerical integration perspective. While computing (3) may seem unattractive from a complexity perspective, as discussed in Hirai et al. (2020), we present how it can indeed be mastered in practice. We look to determine a path of integration that is able to avoid the poles in the integrand without algebraically determining their location and size. We define the process of determining such a path as *optimizing the path of integration*. As we demonstrate, this approach works considerably well in many knapsack settings.

3. Optimizing the Path of Integration

We now discuss the path of integration γ, which turns out to be the central degree of freedom for our computational work. In general, a path that permits fast computations of (3) or (4) will be useful in counting solutions, checking feasibility, and therefore ultimately retrieving the optimal solution to the knapsack problem efficiently. Returning to theoretical results

from complex analysis, we observe that the value of a path integral of a holomorphic function is *independent of the path of integration* as long as it stays in the domain of holomorphy, cf. Rudin (1987, Lem. 10.39). Thus, we can use any γ with $[\gamma] \subseteq D \setminus \{0\}$ that cycles the origin at least once in the mathematically positive direction to compute (3).

3.1. Circular and Elliptic Paths

An obvious choice for γ is a circle of radius $r \in (0, 1)$ centered at the origin. The curve is parameterized by the exponential equation for a circle of radius r, i.e.,

$$\eta_b = \frac{1}{2\pi i} \int_0^{2\pi} \frac{H_a(re^{it})}{(re^{it})^{b+1}} re^{it} \, \mathrm{d}t. \tag{5}$$

A second straightforward choice for the path of integration is an ellipse with radii r_1 and r_2. The ellipse has one additional degree of freedom in its parameterization,

$$\eta_b = \frac{1}{2\pi i} \int_0^{2\pi} \frac{H_a(r_1 \cos(t) + r_2 i \sin(t))}{(r_1 \cos(t) + r_2 i \sin(t))^{b+1}} (-r_1 \sin(t) + r_2 i \cos(t)) \, \mathrm{d}t. \tag{6}$$

We maximize the numerical stability of the two paths (5),(6) by choosing the radii carefully. For the circular path, the geometry on the real line segment $(0, 1)$ is the key to the stability of the integration. As discussed above, $\zeta = 1$ is the dominant pole on the boundary of the unit disk and the other poles are distributed symmetrically on the boundary. This behavior can be observed in Figure 2 (a), where the absolute function values on the segment $(0, 1)$ are greater than those on any other radial segment.

We therefore determine the radius r^* for the circular path by minimizing the absolute value of the integrand function with respect to r on the segment $(0, 1)$ of the real axis. While r^* may not minimize the absolute value of the integrand for points on the circular path outside of the real line segment $(0, 1)$, as can be seen in Figure 2, it guarantees that the path bypasses the critical pole on the real axis at the best possible point. In many cases, this approach solely determines whether the numerical integration converges in our study, see Section 4.

This procedure is repeated for the elliptic path to find the second radius in its parameterization on the segment $(0, i)$, i.e., on the positive imaginary axis. It should be clear that the additional degree of freedom presented by the second radius generally improves the stability since the best bypass points on the positive real and imaginary axes are not the same. In particular, an improved stability can be achieved by moving further away from the origin

on points not on the positive real axis because all poles on the unit circle are dominated by the pole at $\zeta = 1$. This is visualized in Figure 2 (b), in which the circular and elliptic paths do not align on the positive imaginary axis. For certain instances, however, the elliptic path moves too close to a pole between the bypass points on the real and imaginary axis, as can be seen in Figure 2 (a), where the red path achieves the largest function values of all paths. This motivates a path with even more degrees of freedom and therefore an improved numerical stability.

3.2. Shortest Paths

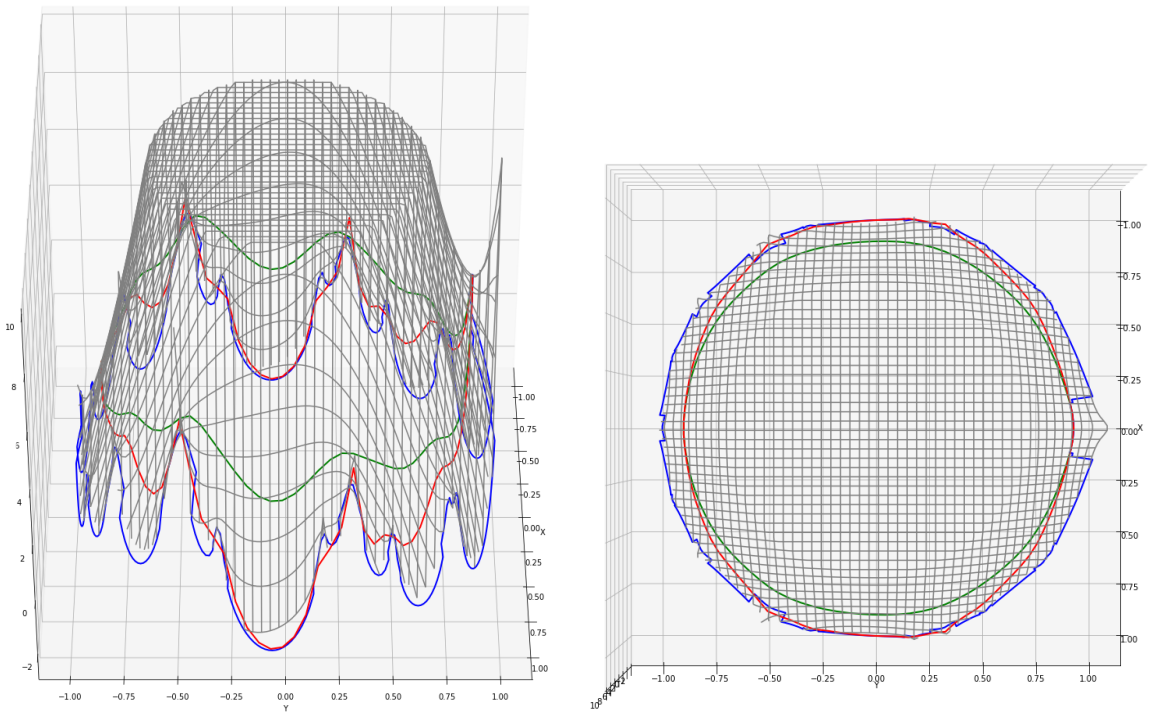
In Bornemann and Wechslberger (2013) a structured approach to finding a stable path of integration is presented, which is based on the condition number for the integration problem, given by

$$\frac{\int_\gamma |H_a(\zeta)||\zeta|^{-b-1}\,\mathrm{d}\zeta}{|\int_\gamma H_a(\zeta)\zeta^{-b-1}\,\mathrm{d}\zeta|}, \tag{7}$$

cf. the introduction to condition numbers in Deuflhard and Hohmann (2008). By path independence, the denominator in (7) is constant for all paths γ inside the unit disk with the same index. Therefore, to minimize the overall condition number, we can focus solely on minimizing the integral in the numerator. The numerator integral is not independent of the path because of the absolute values inside the integral.

Adapting the approach of Bornemann and Wechslberger (2013) to our situation, we construct integration paths of approximately minimal condition by solving a shortest path problem on an auxiliary graph with appropriate edge weights. We consider concentric circles with radii greater than or equal to r^* and symmetric spokes inside the unit disk, as in Figure 3. Nodes at a distance less than r^* from the origin are not considered for the optimal path since the dominant pole on the boundary of the unit disk (located at $\zeta = 1$) defines r^*, and no better path can be found by moving closer to the origin.

Each intersection point of the spokes and circles in the gray area of Figure 3 defines a node of the graph and we consider all edges along the spokes and circles. For two nodes u, v on the same circle, the edge weight w_{uv} is defined as the average of the numerator integrand of (7) at the two end points of the edge, u, v, and the midpoint of the circular arc connecting the two end points, multiplied by the arc length between them. Precisely, let $u = re^{i\theta_1}, v = re^{i\theta_2}$, then



(a) Front view. (b) Top view.

Figure 2 **Paths of integration in D for $a = [3, 5, 6, 9, 10]$ and $b = 21$. The vertical axis displays the absolute function values on a logarithmic scale. Both perspectives show the circle in green, the ellipse in red, and the shortest path in blue. The shortest path places 360 angular nodes on every circular path at a radial distance of 0.001. Note how the shortest path resembles an approximate minimal surface.**

$$w_{uv} = \frac{r|\theta_2 - \theta_1|}{4} \left(\frac{|H_a(re^{i\theta_1})|}{|re^{i\theta_1}|^{b+1}} + 2\frac{|H_a(re^{\frac{i(\theta_1+\theta 2)}{2}})|}{|re^{i\frac{(\theta_1+\theta 2)}{2}}|^{b+1}} + \frac{|H_a(re^{i\theta_2})|}{|re^{i\theta_2}|^{b+1}} \right). \tag{8}$$

Therefore, the weights approximate the integral (7) between the end nodes using the three point trapezoidal rule. For two nodes on the same spoke, the edge weight is defined as the average condition number of the two end points multiplied by the euclidean distance between them. Precisely, let $u = r_1 e^{i\theta}, v = r_2 e^{i\theta}$, then

$$w_{uv} = \frac{|r_2 - r_1|}{2} \left(\frac{|H_a(r_1 e^{i\theta})|}{|r_1 e^{i\theta}|^{b+1}} + \frac{|H_a(r_2 e^{i\theta})|}{|r_2 e^{i\theta}|^{b+1}} \right). \tag{9}$$

Directing all edges between two neighboring spokes in the mathematically positive direction, we compute a shortest path in this graph that is forced to cycle the origin once, starting and ending at r^*. By construction, the sum of the weights along the path approximates the value of the integrand in the numerator of (7), i.e., the condition number multiplied by the

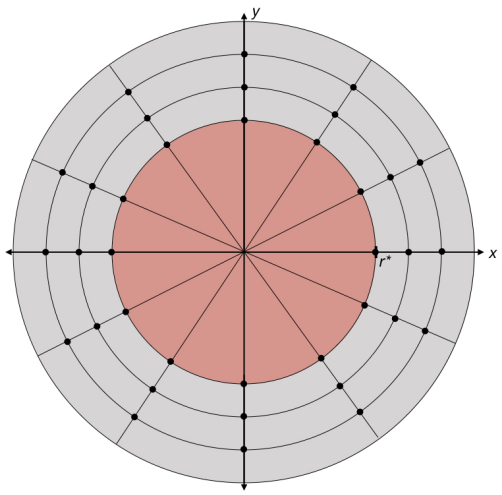


Figure 3 **The shortest path approach: no nodes inside the red circle around 0 are considered (radius r^*). The radial distance, $\triangle r$, is the distance between two neighboring circles, the number of angular nodes, here N=12, is the number of nodes on each circle.**

constant denominator in (7). The optimal path circumvents the poles located on the boundary of the unit disk, as displayed in Figure 2, and balances the cost of moving towards or away from the pole at the origin.

The graph in this construction is parameterized by $(N, \triangle r, r^*)$, i.e., the number of nodes on any concentric circle, the radial distance between nodes on the same spoke, and the minimum radius, i.e., the radius of the red circle in Figure 3. A good choice of these parameters balances the computational cost of the shortest path problem and the computational advantage this provides for the path integral.

As discussed in Section 2.2, the properties of the integrand in (3) depend on the instance at hand. Thus, the optimal parameters for the shortest path algorithm are unique to the instance, and it is impossible to find a parameterization that is optimal for *all* instances. In Figure 4 we see that a parameterization with few angular nodes, thus small N, may be sufficient for instances with a simple distribution of poles on the boundary, such as (a), since it successfully avoids the high order poles on the boundary. Meanwhile, the parameterization with large N in Figure 4 (b) provides little computational advantage for the numerical integration, as it is not necessary to avoid the smaller poles on the boundary for the numerical quadrature to converge. The parameterization with small N causes the path to run into several poles in Figure 4 (c), which has a far more elaborate pole structure on the boundary, and this could prevent the integration from converging. These poles can be avoided using a

larger N, as done in Figure 4 (d). This illustrates that no specific parameterization can be optimal for all instances.

Instead of discussing a theoretical framework for the optimal parameters of the shortest path algorithm, as initiated in Section 2.2, we look to tune the parameters *a posteriori*. Specifically, we focus on a set of knapsack instances with similar properties of the integrand in (3). We select a subset of these instances and carefully tune $(N, \triangle r, r^*)$ for these instances, before testing this parameterization on the entire set.

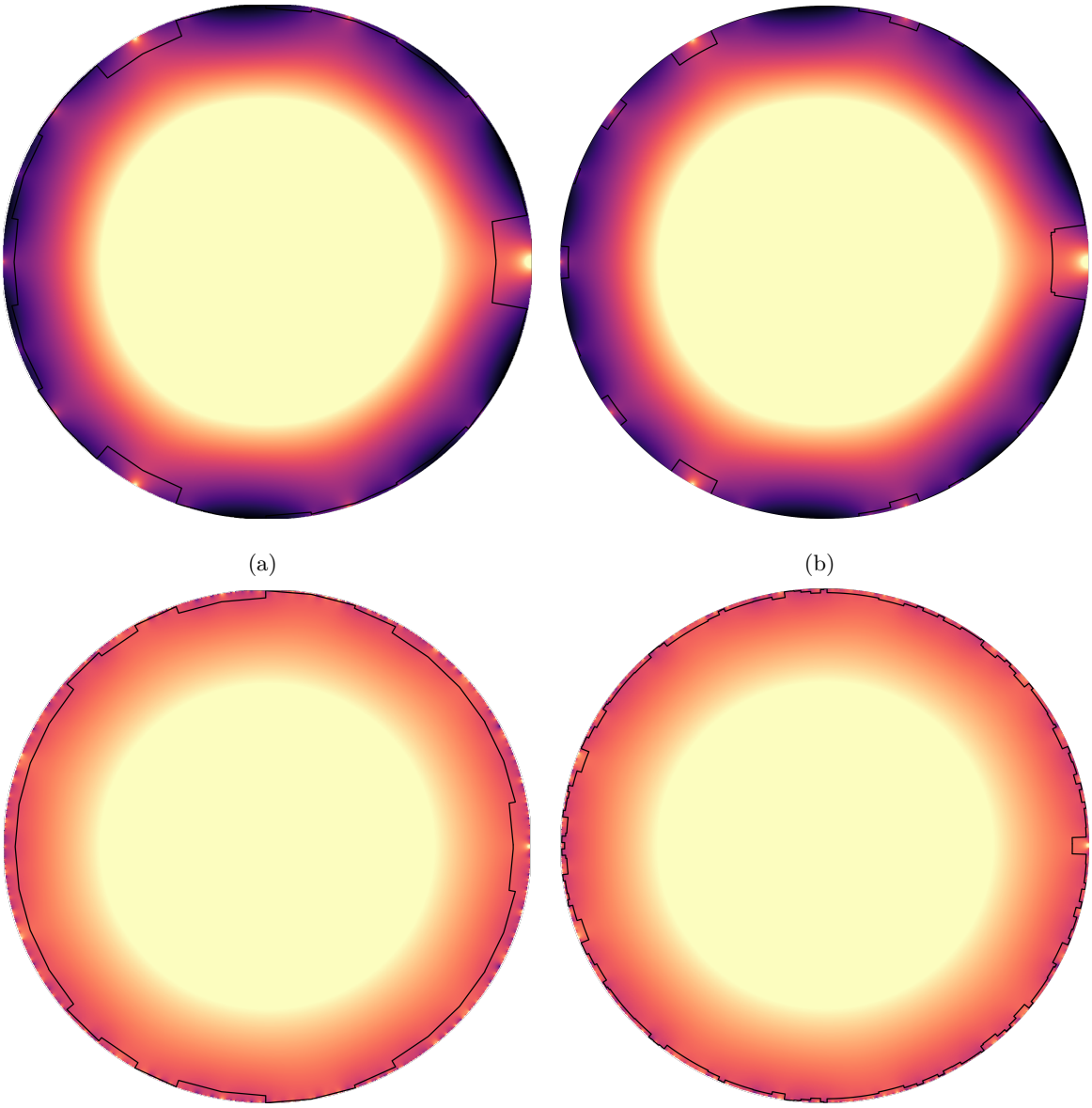
4. Numerical Study

We base our study on a set of hard knapsack instances constructed in Pisinger (2005). These instances all have 50 coefficients in the range $a_n \in (1, 1000)$ and $b \in (900, 6000)$. They are sorted by increasing values of b. Judging from the relative magnitude of the b values and the number of coefficients, we expect the center pole to dominate the condition number within the unit disk for all instances with increasing magnitude, while the distribution and size of the poles on the boundary varies depending on the coefficient vector. Figure 5 visualizes a Pisinger instances and the three paths discussed in Section 2, and confirms the dominance of the pole at the origin.

We begin by investigating the density of path nodes in the unit disk. It is clear that adding more nodes to the graph provides more opportunities for the shortest path to achieve lower condition numbers, while increasing the complexity of the shortest path problem. Without any theoretical analysis of the Pisinger instances, we aim at finding a set of shortest path parameters that outperforms the circular and elliptical parameterizations. We evaluate the performance by looking at the condition numbers achieved by the individual paths, as well as the computation and integration times. The results are tabulated in Tables 2 and 3.

In Section 4.3 we look beyond the Pisinger instances, and evaluate the parameterizations on artificial instances with controlled properties.

The program codes and test data are available online at `https://github.com/endric-daues/ipcauchy`. All integrals are evaluated in Python using the quadpy library, which provides more than 1500 numerical integration schemes. We use a fully vectorized adaptive quadrature method and input an error tolerance of $tol = 1$ since we expect integer values for η_b. All computations are run on a MacBook Pro with 8 GB of RAM. The computation times are reported in seconds with a time limit of 300 seconds for each run.



(a) (b)

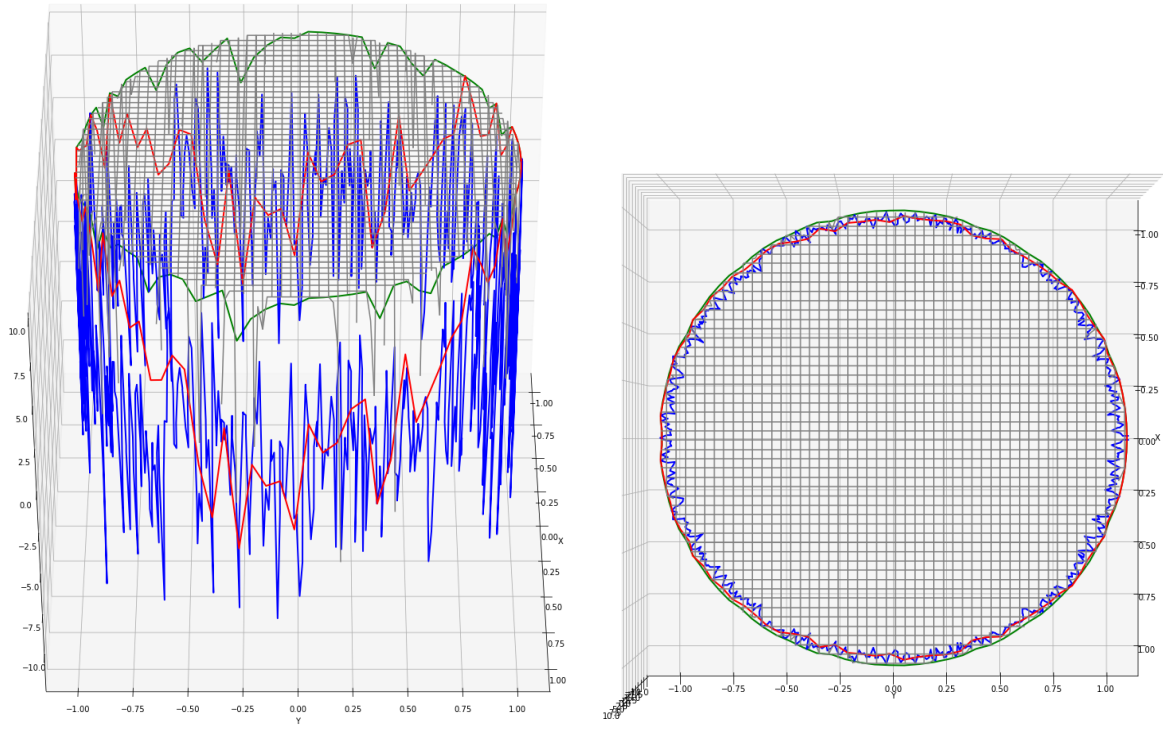
(c) (d)

Figure 4 **The instance in (a) and (b) is the same as in Figure 1 (a), while (c) and (d) use the same b value as the above and a coefficient vector from instance p1 below, taken form the set of hard knapsacks by Pisinger (2005).**

4.1. Tuning the Shortest Path Parameters

4.1.1. Tuning $\triangle r$ First, we vary the radial distance between nodes in the range $\triangle r \in (0.1,\ 0.0001)$ while maintaining $N = 360$ angular nodes. The results are graphed in Figure 6.

We see that above a certain threshold, the radial step size $\triangle r$ permits no steps towards or away from the origin without passing outside of the grey area in Figure 3. This can be seen in Figure 6 for all points with zero radial adjustments. Since we aim at parameterizing an



(a) Front view. (b) Top view.

Figure 5 **The paths of integration for the Pisinger instance p1. Both perspectives show the circle in green, the ellipse in red, and the shortest path in blue. The graph for the shortest path places 500 angular nodes on every circular path at a radial distance of 0.001. The absolute function values on the vertical axis are transformed to a logarithmic scale.**

agile shortest path that can circumvent poles, unlike the circular parameterization, we look to stay well below this threshold.

In Figure 7 we see that for all radial distances in the range $\triangle r(0.001,\ 0.01)$ the condition number and computation time are near minimal. A smaller $\triangle r$ value places nodes too close to the poles on the boundary of the unit disk, which can cause a larger approximation error and a suboptimal path to be selected, increasing the condition number and the integration time. We select $\triangle r = 0.001$ since it produces a near minimal condition number in all test instances, while also inducing a large number of radial adjustments at an acceptable computation time.

4.1.2. Tuning N Next, maintaining a radial distance of $\triangle r = 0.001$, we vary the number of angular nodes per concentric circle in the range $N \in (4,\ 1000)$. We observe in Figure 7 that the condition number declines as we increase the number of nodes. More angular nodes provide the shortest path instance with more opportunities to circumvent the poles on the boundary of the unit disk, thereby reducing the condition number.

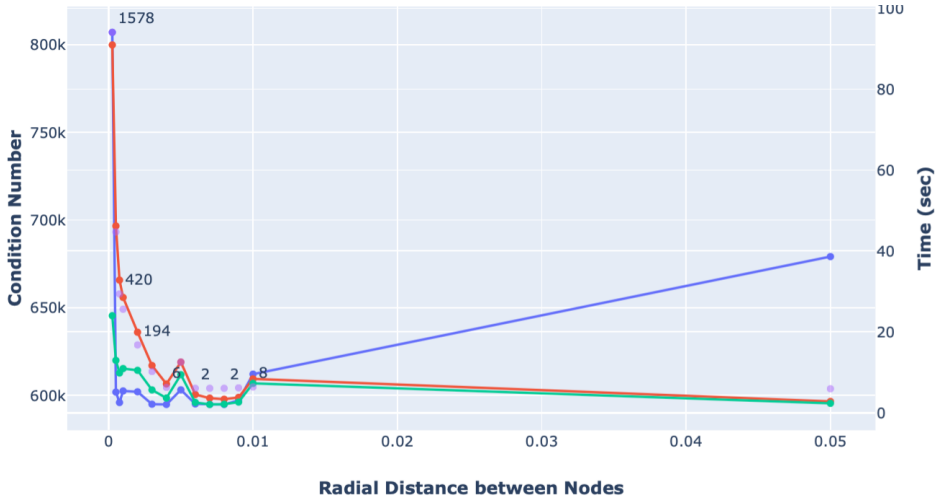
It is interesting to note that small changes in the location and density of nodes cause a varying stability in the condition number. This can be seen in the varying smoothness of the blue line in Figure 7. For instances with poles of large relative order on the boundary, the approximation error in the estimate of the condition number of the path is sensitive to the positioning of the nodes. Varying the arc length between two nodes at the same radial distance and their position with respect to a pole can move the pole into a blindspot for the 3-point trapezoid approximation. This causes the sudden jumps in the condition number in Figure 7 (a). For instances with a dominant pole at the origin, such as p6, the relative size of the poles on the boundary is considerably smaller, and the relative position of nodes with respect to these poles causes smaller relative fluctuations, which are not visible in Figure 7 (c).

The computation time increases linearly with increasing N as expected, while the integration time remains fairly consistent. There are several justified choices for N here. In terms of computation time, Figure 7 indicates that $N = 36$ radial nodes provides a solid low-cost choice, while permitting more than 30 adjustments in all three cases, thereby guaranteeing an improvement in condition number over the circular and elliptic parameterizations.

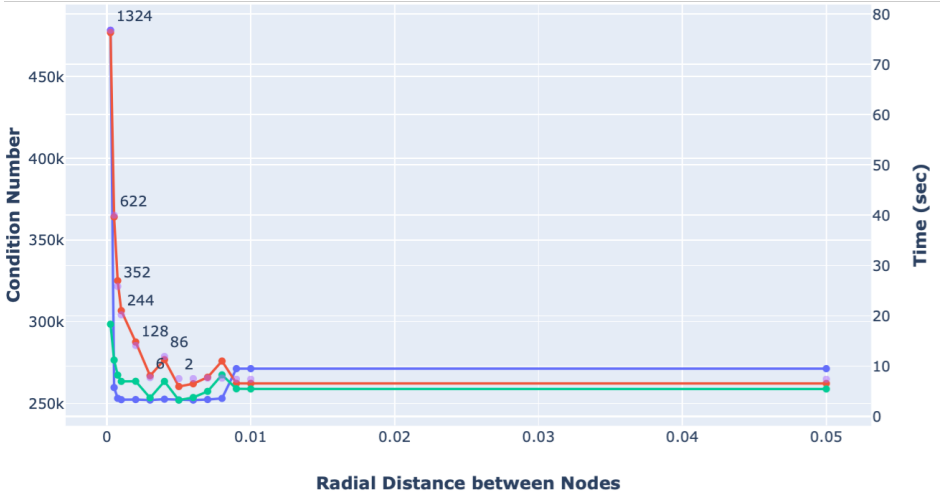
Meanwhile, $N = 500$ provides a more expensive shortest path problem, but generally guarantees an even lower condition number for the path, which could potentially enable convergence for larger instances that cannot be solved with $N = 36$. For reference, and to fulfill our curiosity, we continue our experiments with both parameterizations. Note that r^* is computed for each instance separately as it is a low-cost computation and can drastically reduce the complexity of the shortest path problem.

4.2. Hard Instances

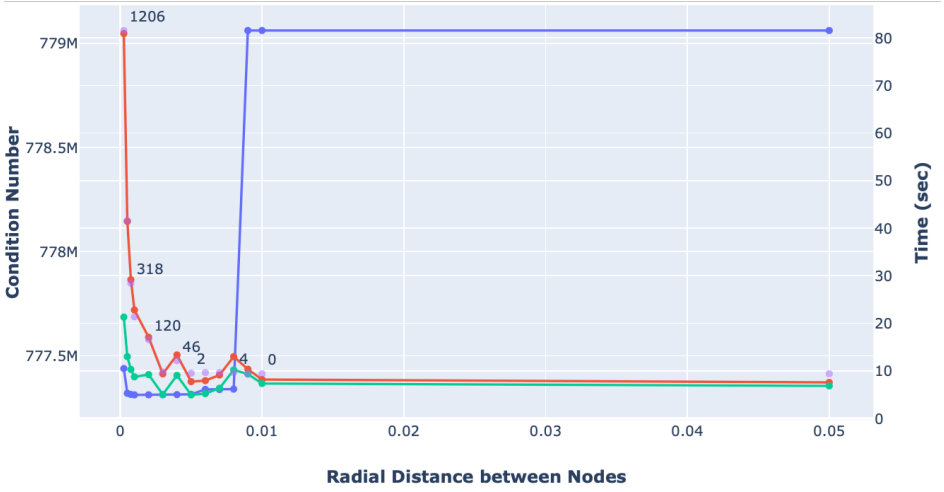
We now apply the results of the parameter tuning to a collection of instances taken from Pisinger (2005). Since we know that the center pole dominates the condition number within the unit disk for all instances, we expect the shortest path with $N = 500$ to offer little if any advantage over the path with $N = 36$ in terms of overall computation time. Nevertheless, the larger parameterization could offer some benefits for instances with repeated factors in the coefficient vector. Furthermore, we aim at displaying the advantage of pre-optimizing the path of integration by comparing the performance of the shortest path parameterizations to the simple geometries of the circle and the ellipse.



(a) Instance p1.

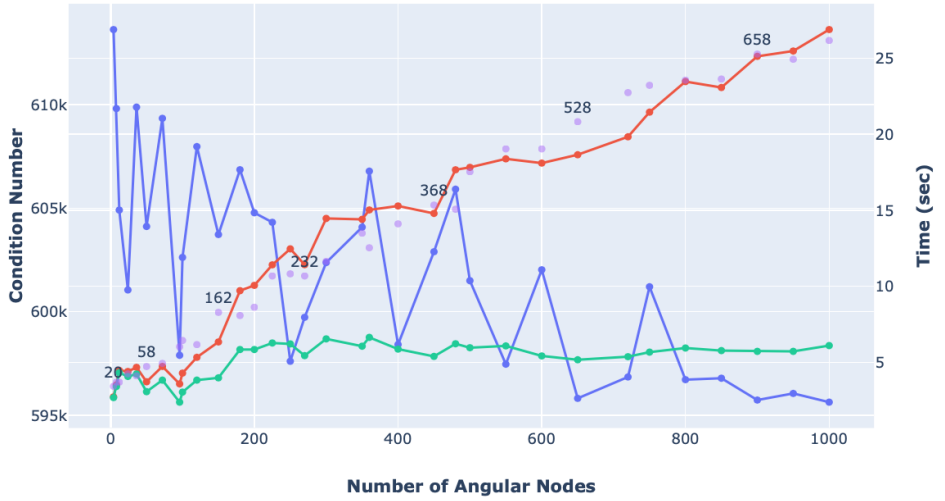


(b) Instance p3.

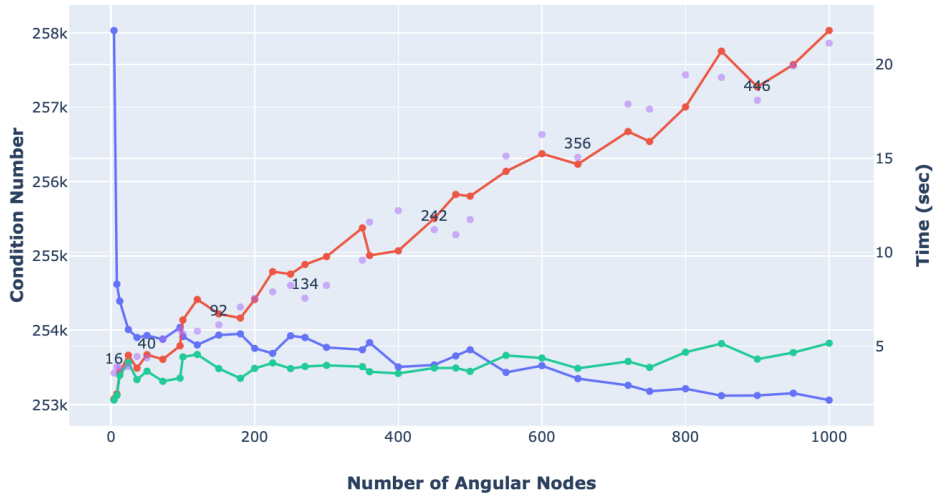


(c) Instance p6.

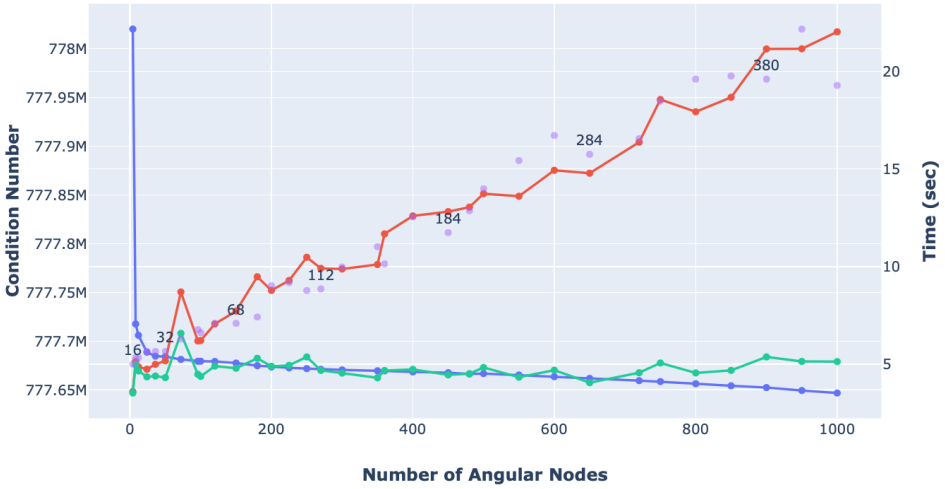
Figure 6 **Change in the condition number (blue), the total computation time (red), and the integration time (green), for a varying radial distance, $\triangle r$, while maintaining $N = 360$. The number of radial adjustments (labeled purple points) tracks how often the shortest path algorithm selects a node at a different radial distance as the previous.**



(a) Instance p1.



(b) Instance p3.



(c) Instance p6.

Figure 7 **Change in the condition number (blue), the total computation time (red), and the integration time (green), for a varying number of angular nodes, N, while maintaining $\triangle r = 0.001$. The number of radial adjustments (labeled purple points) as previously.**

Table 3 confirms that the shortest path with $N = 500$ achieves the minimum condition numbers but requires longer computation times for all instances. There are several instances, such as p3, p5, p7, p8, p9, p14, p19, and p20, where the path with $N = 500$ is able to turn the smaller condition numbers into shorter integration times. This indicates that despite having to compute considerably more integrals on smaller intervals, the integration converges faster on the path with $N = 500$ due to an improved path selection. While this is a successful improvement in itself, it does not pay off when considering the total computation time, at least for these instances. We will explore instances for which such a parameterization can be effective in the next section.

Both paths are able to count solutions to all instances except p21 and p24, while the circular and elliptic parameterizations in Table 2 fail for all instances larger than p17. In Table 2 we also see that the ellipse achieves smaller condition numbers than the circle, and can achieve shorter computation times for the larger instances. For the smaller instances, such as p1, it seems that computation the second radius does not pay off in terms of the computation time.

In general, we can confirm our initial hypothesis that increasing the degrees of freedom in the parameterization of the path decreases the condition number at a cost of increased computation time. As the integrand in (3) grows and becomes more costly to compute, however, the additional path optimization can induce shorter integration times and for harder instances, even shorter total computation times. For example, the circle outperforms all other parameterizations in terms of computation time for smaller instances such as p1. However, as displayed in Table 2 and 3, for harder instances such as p16, the path optimization performed by the shortest path provides an advantage not only in terms of integration time, but also in total computation time. For this instance, the circular path causes the quadrature to get stuck close to a pole on the boundary, which the shortest path can conveniently avoid, and enable the shortest path problem and integral to complete in shorter time.

Furthermore, we have shown that although the optimal shortest path parameterization is unique to each knapsack instance, we can successfully apply it on a set of similar instances, so far as our computational capacity permits. Without invoking any algebraic techniques, a simple analysis of the instance at hand, such as comparing the b value to the size of a, will permit a clustering of instances into groups that can be solved with the same shortest path parameterization.

4.3. Further Study

To solidify the above argument, we now move beyond the Pisinger instances and look to evaluate the two shortest path parameterizations from our numerical study in a more controlled setting. Instead of evaluating the parameters on a set of similar instances, we initialize an artificial instance and vary its properties independently. Keeping in mind the arguments in Section 2.2, we increase the b value and the number of coefficients in a independently, thereby altering the location and size of dominant poles. We are interested in how this affects the computation times and condition numbers of the paths.

4.3.1. Increasing the b Value We fix a vector of 20 coefficients and a starting b value of 50. We increase the b value by 50 in each iteration, iteratively growing the pole at the origin by an order of 50. Table 4 indicates that a shortest path with more angular nodes achieves a smaller condition number for all instances, as before. Again, the increased computational complexity of the path results in a significantly longer computation time.

As the b value increases, the total computation time for both paths decreases, while more significantly for the parameterization with $N = 500$. This is caused by the decreasing complexity of the shortest path problem as r^* is pushed towards the boundary as the pole at the origin grows, shrinking the feasible region for the shortest path. The integration time for the parameterization with $N = 36$ angular nodes remains consistent, whereas the second parameterization displays a decreasing integration time. Again, this is expected as less adjustments are made.

We can conclude that the shortest path with $N = 36$ is sufficient for these instances. With 20 coefficients in a, it seems that 36 angular nodes are sufficient to avoid all the boundary poles induced by the coefficients, and more angular nodes provide no computational advantage.

4.3.2. Increasing the number of coefficients Next, we fix a b value of 200 and the same initial a vector as in Section 4.3.1. We increase the number of coefficients n by generating additional random coefficients in $[1, 50]$ and appending these to a. This increases the density and size of the poles on $\{z : |z| = 1\}$. Table 5 indicates that the computation and integration times increase with growing n for both shortest paths. Again, the parameterization with $N = 36$ is able to achieve considerably faster times while the parameterization with $N = 500$ achieves smaller condition numbers. For the two largest instances we now see that the path

with $N = 500$ permits the numerical integration to converge, while the path with $N = 36$ fails. As suspected, achieving smaller condition numbers can be crucial to whether or not the integration converges. For instances with more than 100 coefficients, the path with $N = 36$ angular nodes is not agile enough to avoid the growing boundary poles, preventing the numerical integration from converging to a solution.

Thus, we have found a case where the additional computational cost of solving a larger shortest path problem pays off. When $|a| = 110, a_i \in [1, 50]$, the shortest path problem with $N = 500$ takes more than 150 seconds to complete, as shown in Table 5, but permits the integration to converge in less than 10 seconds, while $N = 36$ fails. The shortest path method with $N = 500$ counts more than $0.73 \cdot 10^{18}$ feasible solutions with an approximate error of $\pm 1 \cdot 10^4$, which is more than sufficient for a feasibility check. While for the smaller instances the error is well below ± 1, this begins to grow as the function values in (3) grow. Here a customized numerical integration scheme, which can interrupt the computation once a sufficient relative error bound is achieved, would be helpful.

5. Conclusion and Future Work

We have extended the known theory of counting integer points with the help of complex path integrals by carefully analyzing the integration path. In particular, we have demonstrated the practical effectiveness of optimized paths for Cauchy-type integrals of knapsacks. Our numerical study shows that the fine-tuned algorithm can count feasible solutions to hard instances on a standard laptop and check feasibility for even larger instances. Moreover, algorithms based on a simple circular path that have been proposed in earlier works failed to converge. The results are promising enough to be used in the more generalized setting of multi-row IP, thus enabling us to solve optimization problems via a binary search in the future.

Our implementation can certainly be improved by paying more attention to the numerical quadrature. While we rely on the quadpy library in Python, an adaptive integration scheme optimized for the geometry of Cauchy integrals might deliver better results. Further, we did not make use of the error bounds provided by the quadrature routine in a structured way. In view of the decision problem (4), stopping the computations much earlier is advisable when reliable error bounds exist. Of course, from a computational perspective, we assume that a programming language with less overhead than Python would perform better. In addition,

any method to predict the arrangement and dominance of the poles based on the number of coefficients, their size, and the value of b could improve the choice of the shortest path parameterization, $(N, \triangle r, r^*)$, before it is run, thereby pre-optimizing the parameterization for the instance. This could involve an algebraic or a learning approach.

Acknowledgments

This research is supported by the Volkswagen Foundation via the Experiment! initiative and by the Alexander von Humboldt Foundation with funds from the German Federal Ministry of Education and Research (BMBF).

Appendix. Additional Computational results

Table 1 Summary of all variables used in the following numerical analysis.

Abbreviation	*Meaning*
b	Right hand side constraint value
n	Size of the coefficient vector a
r_1	Optimal radius (bypass point) on the positive real axis
r_2	Optimal radius (bypass point) on the positive imaginary axis
r^*	Optimal radius (bypass point) on the positive real axis
sol	Computed number of feasible solutions
sol_{SP}^N	Computed number of feasible solutions using a shortest path with parameters $(N, 0.001, r^*)$
t_{circ}	Total computation time for the circular parameterization
t_{ellip}	Total computation time for the elliptic parameterization
t_{SP}^N	Total computation time for the shortest path parameterization with parameters $(N, 0.001, r^*)$
t_{SPint}^N	Integration time only for the shortest path parameterization with parameters $(N, 0.001, r^*)$
c_{circ}	Condition number for the circular parameterization
c_{ellip}	Condition number for the elliptical parameterization
c_{SP}^N	Condition number for the shortest path parameterization with parameters $(N, 0.001, r^*)$
$\text{adj}_{\text{SPint}}^N$	Number of radial adjustments for the shortest path with parameters $(N, 0.001, r^*)$

Table 2 Overview of the computational results for the simple geometries, the circle and the ellipse, on the instances taken from Pisinger (2005). The ellipse requires longer computation times for smaller instances, but performs faster on the larger instances. The ellipse always guarantees a smaller condition number than the circle, due to the additional degree of freedom provided by the second radius.

	sol	r_1	r_2	t_{circ}	c_{circ}	t_{ellip}	c_{ellip}
p1	544429	0.98962	0.99887	1.49	678655	1.71	612697
p2	28060	0.99131	0.9963	1.62	49176	1.90	36205
p3	220560	0.99096	0.99759	1.34	271239	1.26	257825
p4	5502	0.99074	0.99822	1.34	22999	1.38	10675
p5	152417	0.99303	0.99745	1.50	207614	2.70	183162
p6	715681393	0.99112	0.99821	2.00	779029257	2.06	778130512
p7	1439298	0.99348	0.99809	2.80	2168347	2.53	1683157
p8	60979453	0.99356	0.99861	2.66	68189975	2.83	66345913
p9	4849303	0.99401	0.99733	3.04	6135125	4.26	5417261
p10	50856154	0.99513	0.99773	5.51	56384073	3.43	55023923
p11	42417243268	0.99365	0.99781	3.84	45228184538	3.09	45203883128
p12	13244852	0.99610	0.9985	5.59	17917502	5.00	14771042
p13	13466832804	0.99501	0.99837	5.50	14296550740	4.45	14257710276
p14	57612978960	0.99551	0.99899	7.37	60901637061	5.37	60843896655
p15	1981736210	0.99610	0.99894	5.81	2125025756	5.13	2087465101
p16	2169533751	0.99606	0.9985	19.88	2344359185	11.41	2283324821
p17	324480407213	0.99571	0.99812	9.84	339691087150	8.21	339414418877
p18	-	0.99594	0.99852	>300	-	>300	-
p19	-	0.99553	0.99859	>300	-	>300	-
p20	-	0.99666	0.99877	>300	-	>300	-
p21	-	0.99599	0.99869	>300	-	>300	-
p22	-	0.99616	0.99762	>300	-	>300	-
p23	-	0.99637	0.99797	>300	-	>300	-
p24	-	0.99624	0.99854	>300	-	>300	-

Table 3 **Overview of the computational results for the shortest path method on the instances taken from Pisinger (2005). The table documents the number of unique solutions to the instances for the two shortest path parameterizations, with $N = 36$ and $N = 500$ respectively, as well as their respective condition numbers, computation and integration times, and the number of radial adjustments made on the path. The relative error in the number of solutions is maintained consistently, but exceeds integer values for the larger instances. From Table 2 we see that the paths with $N = 36$ maintain shorter computation times throughout, but the paths with $N = 500$ can frequently convert the smaller condition numbers into shorter integration times.**

	r^*	$\text{sol}_{\text{SP}}^{36}$	t_{SP}^{36}	t_{SPint}^{36}	adj_{SP}^{36}	c_{SP}^{36}	$\text{sol}_{\text{SP}}^{500}$	t_{SP}^{500}	t_{SPint}^{500}	adj_{SP}^{500}	c_{SP}^{500}
p1	0.98962	544429	5.24	4.68	40	608288	544429	11.65	3.75	432	600547
p2	0.99131	28060	3.4	2.95	46	41433	28060	10.1	3.33	366	155863
p3	0.99096	220560	3.17	2.6	44	252396	220560	8.53	2.41	258	252434
p4	0.99074	5501	4.62	4.15	48	197617	5502	11.05	4.38	552	642928
p5	0.99303	152416	2.94	2.61	24	173175	152417	6.97	2.47	208	172550
p6	0.99112	715681393	3.06	2.64	32	777281397	715681393	8.58	2.74	240	777281340
p7	0.99348	1439298	4.15	3.83	30	1718130	1439298	8.24	3.74	246	1618354
p8	0.99356	60979453	4.77	4.46	28	66499849	60979453	8.54	4.04	192	66548927
p9	0.99401	4849303	3	2.74	18	5206232	4849303	6.55	2.74	86	5205048
p10	0.99513	50856154	3.59	3.37	16	54470791	50856154	6.58	3.44	64	54468836
p11	0.99365	42417243269	4.83	4.51	34	45212704840	42417243268	9.17	4.66	206	45212672445
p12	0.99610	13244852	3.83	3.66	8	14037288	13244852	6.17	3.69	16	14036941
p13	0.99501	13466832804	3.48	3.26	10	14250577663	13466832803	6.5	3.36	18	14250575775
p14	0.99551	57612978960	5.29	5.07	12	60832767794	57612978959	8.18	5.02	64	60832758832
p15	0.99610	1981736210	3.9	3.72	8	2081446425	1981736210	6.57	4.08	8	2081445184
p16	0.99606	2169533751	3.81	3.64	8	2269475079	2169533751	6.38	3.89	12	2269472159
p17	0.99571	324480407213	5.55	5.33	28	339366178574	324480407213	9.53	6.36	100	339362115922
p18	0.99594	2938495443637	3.83	3.65	8	3074645993635	2938495443638	6.38	3.91	8	3074645967170
p19	0.99553	333806322641172	5.32	5.1	12	348618396597779	333806322641188	8.01	4.87	24	348618396575776
p20	0.99666	144048903115529	7.17	6.99	8	150376189492629	144048903115544	9.2	6.73	28	150376189489663
p21	0.99599	-	>300	>300	-	-	-	>300	>300	-	-
p22	0.99616	672271093810708	4.47	4.3	10	697432025032480	672271093810763	7.19	4.71	10	697432024320699
p23	0.99637	150556678598701	5.01	4.84	8	156126260246894	150556678598714	7.52	5.05	8	156126260225020
p24	0.99624	-	>300	>300	-	-	-	>300	>300	-	-

Table 4 **Increasing the b value increases the size of the pole at the origin. This moves the starting radius further to the boundary, and decreases the number of nodes in the shortest path problem, decreasing the computation time. As expected, the condition number for the parameterization with $N = 500$ is considerably smaller. The two parameterizations produce the exact same solutions for the smaller instances, and begin producing small relative discrepancies for the last few instances. The error pattern is similar to what is observed in 3 and results from the error in the numerical integration scheme.**

b	sol	t_{SP}^{36}	t_{SPint}^{36}	c_{SP}^{36}	adj_{SP}^{36}	t_{SP}^{500}	t_{SPint}^{500}	c_{SP}^{500}	adj_{SP}^{500}
50	510	3.64	0.94	634	416	35.90	4.94	604	1962
100	34434	2.92	0.88	38661	298	27.06	4.00	38193	1420
150	875553	2.37	0.97	953080	332	22.53	3.08	949750	986
200	12706352	2.08	0.95	13733403	272	20.32	2.94	13623868	860
250	126508562	1.89	0.91	134828544	228	15.50	2.47	134737635	670
300	956593952	1.82	0.97	1014519018	204	12.39	1.85	1014318588	572
350	5852912692	1.54	0.83	6389699415	180	11.20	1.76	6187498785	488
400	30242661625	1.76	1.02	32910844190	166	10.17	1.69	31900185106	462
450	136049809507	1.39	0.81	152339950309	148	10.94	1.82	143249092647	376
500	545027910355	1.96	1.18	606976426541	114	11.56	1.98	573273368341	370
550	1978312132399	1.21	0.76	2189928633123	104	8.77	1.58	2078739683042	336
600	6595122641660	1.67	1.12	7444688718918	100	9.95	2.14	6924267384509	382
650	20413951150269	1.69	1.11	22833010488655	94	9.28	1.92	21419757867733	284
700	59191214892887	1.77	1.23	66952634340672	98	10.22	2.06	62063790182628	272
750	161955797166090	1.42	1.00	181032537420752	96	7.78	1.95	169725296202847	258
800	420735861015145	1.31	0.95	609365125506520	88	6.57	1.51	440793071785804	240
850	1043161193458690	1.23	0.9	1338341315912620	86	6.01	1.41	1092689749563940	208
900	2479412642859080	1.19	0.88	2955780219429130	78	6.10	1.41	2596004038988620	194
950	5671026462624020	1.13	0.81	6463996048212330	70	5.60	1.46	5937510299177790	166
1000	12523653711398100	1.35	1.05	13875597544999900	76	5.79	1.63	13108782372348000	214
1050	-	>300	>300	-	-	>300	>300	-	-

Table 5 The b value is maintained at $b = 200$ while the number of coefficients in the a vector is increased, where the coefficients are randomly generated in $[1, 50]$. This causes the poles on the boundary of the unit disk to increase in size, while the pole at the origin remains constant. The shortest path problem becomes more costly to compute as the size of the poles on the boundary of the unit disk increases, and avoiding these reduces the condition number severely. The computation time increases for both shortest path parameterizations, but the shortest path with $N = 500$ angular nodes displays longer computation times. Nevertheless, this results in considerably smaller condition numbers, and ultimately permits additional instances to converge in the numerical integration.

n	$\text{sol}_{\text{SP}}^{36}$	t_{SP}^{36}	t_{SPint}^{36}	c_{SP}^{36}	adj_{SP}^{36}	$\text{sol}_{\text{SP}}^{500}$	t_{SP}^{500}	t_{SPint}^{500}	c_{SP}^{500}	adj_{SP}^{500}
20	12706352	1.99	0.92	13733403	272	12706352	15.37	2.22	13623868	860
30	3231494637	3.18	1.38	3416220179	400	3231494637	26.80	2.63	3414894481	982
40	1926080459195	4.60	1.75	2018576130445	672	1926080459195	42.05	3.53	2018339150639	1330
50	112741441577786	5.95	1.92	117363732058470	684	112741441577787	59.68	4.16	117358635936184	1406
60	249378103539524	7.46	2.27	259247694103850	740	249378103539526	72.86	4.41	259238025981771	1470
70	802703537456477	8.24	2.33	834172365560111	754	802703537456480	85.42	5.19	834139231395788	1724
80	3293394167012470	9.45	2.48	3420974571790020	760	3293394167012480	100.74	5.94	3420857654483900	1878
90	10497306954818300	10.79	2.62	10889817400463400	718	10497306954818300	117.89	6.43	10889564880473200	1946
100	-	>300	>300	-	-	442500538243987000	142.45	7.17	457795568727533000	1984
110	-	>300	>300	-	-	731645471006155000	168.69	9.03	756858595535446000	2206
120	-	>300	>300	-	-	-	>300	>300	-	-

References

Baldoni V, Berline N, De Loera JA, Dutra B, Koeppe M, Vergne M (2014) Coefficients of Sylvester's denumerant. *arXiv preprint arXiv:1312.7147* .

Baldoni-Silva MW, De Loera JA, Vergne M (2004) Counting integer flows in networks. *Foundations of Computational Mathematics* 4(3):277–314.

Barvinok A (1994a) A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Mathematics of Operations Research* 19(4):769–779.

Barvinok A (1994b) Computing the Ehrhart polynomial of a convex lattice polytope. *Discrete and Computational Geometry* 48:35–48.

Barvinok A, Pommersheim J (1999) An algorithmic theory of lattice points in polyhedra. Billera L, Björner A, Greens C, Simion R, Stanley R, eds., *New Perspectives in Algebraic Combinatorics*, 91–148 (Cambridge: Cambridge University Press).

Beck M (2000) Counting lattice points by means of the residue theorem. *The Ramanujan Journal* 4:299–310.

Beck M (2004) The partial-fractions method for counting solutions to integral linear systems. *Discrete and Computational Geometry* 32(4):437–446.

Beck M, Pixton D (2003) The Ehrhart polynomial of the Birkhoff polytope. *Discrete & Computational Geometry* 30(4):623–637.

Beck M, Robins S (2002) Explicit and efficient formulas for the lattice point count in rational polygons using dedekind-rademacher sums. *Discrete and Computational Geometry* 27(4):443–460.

Bornemann F, Wechslberger G (2013) Optimal contours for high-order derivatives. *IMA Journal of Numerical Analysis* 33:403–412.

Brion M, Vergne M (1997a) Lattice points in simple polytopes. *Journal of the American Mathematical Society* 10(2):371–392.

Brion M, Vergne M (1997b) Residue formulae, vector partition functions and lattice points in rational polytopes. *Journal of the American Mathematical Society* 10(4):797–833.

De Loera JA (2005) The many aspects of counting lattice points in polytopes. *Mathematische Semesterberichte* 52(2):175–195.

De Loera JA, Haws D, Hemmecke R, Huggins P, Yoshida R (2004) Three kinds of integer programming algorithms based on Barvinok's rational functions. Nemhauser G, Bienstock D, eds., *Proceedings of the 10th International Conference on Integer Programming and Combinatorial Optimization, IPCO, New York, NY, June 7-11, 2004*, volume 3064 of *LNCS*, 244–255 (Springer).

De Loera JA, Haws D, Hemmecke R, Huggins P, Yoshida R (2005) A computational study of integer programming algorithms based on Barvinok's rational functions. *Discrete Optimization* 2(2):135–144.

Deuflhard P, Hohmann A (2008) *Numerische Mathematik 1* (Berlin: de Gruyter), 4 edition.

Euler L (1770) De partitione numerorum in partes tam numero quam specie datas. *Novi Commentarii Academiae Scientiarum Imperialis Petropolitanae* 14:168–187.

Friedrich U (2016) *Discrete Allocation in Survey Sampling and Analytic Algorithms for Integer Programming*. PhD thesis, Trier University.

Friedrich U (2020) Solving IP via complex integration on shortest paths. Preprint available at `http://www.optimization-online.org/DB_HTML/2020/06/7848.html`.

Grötschel M, Lovász L, Schrijver A (1988) *Geometric Algorithms and Combinatorial Optimization* (Berlin: Springer).

Hirai H, Oshiro R, Tanaka K (2020) Counting integral points in polytopes via numerical analysis of contour integration. *Mathematics of Operations Research* 45(2):455–464.

Köppe M (2007) A primal Barvinok algorithm based on irrational decompositions. *SIAM Journal on Discrete Mathematics* 21(1):220–236.

Köppe M (2018) LattE (lattice point enumeration) computer software, version 1.7.5. Available online at `https://github.com/latte-int`.

Köppe M, Louveaux Q, Weismantel R, Wolsey LA (2004) Extended formulations for Gomory corner polyhedra. *Discrete Optimization* 1(2):141–165.

Lasserre J, Zeron E (2003) On counting integral points in a convex rational polytope. *Mathematics of Operations Research* 28(4):853–870.

Lasserre JB (2009) *Linear and Integer Programming vs Linear Integration and Counting: A Duality Viewpoint*. Springer Series in Operations Research and Financial Engineering (Berlin: Springer).

Lasserre JB, Zeron ES (2001) A Laplace transform algorithm for the volume of a convex polytope. *Journal of the ACM (JACM)* 48(6):1126–1140.

Lasserre JB, Zeron ES (2002) Solving the knapsack problem via Z-transform. *Operations Research Letters* 30:394 – 400.

Lasserre JB, Zeron ES (2007) Simple explicit formula for counting lattice points of polyhedra. Fischetti M, Williamson DP, eds., *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization, IPCO, Ithaca, NY, June 25-27, 2007*, volume 4513 of *LNCS*, 367–381 (Springer).

Pisinger D (2005) Where are the hard knapsack problems? *Computers & Operations Research* 32(9):2271–2284.

Rudin W (1987) *Real and Complex Analysis* (New York: McGraw-Hill), 3 edition.