Solving IP via Complex Integration on Shortest Paths

Ulf Friedrich
Operations Research, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany, ulf.friedrich@tum.de

Abtstract: Using the weighted geometric series expansion, it is shown how integer programming can be solved by evaluating complex path integrals based on a multi-path version of Cauchy's integral formula. In contrast to existing generating function approaches, the algorithm relies only on complex quadrature and no algebraic techniques are needed. In view of fast implementations of the method, it is demonstrated how preprocessing the path of integration improves the condition number of the quadrature problem. An algorithm that uses the shortest path of integration is discussed and illustrated by computing feasibility instances of knapsack problems.

Keywords: Integer Programming, Generating Function, Cauchy's Integral Formula, Shortest Path of Integration.

1. Introduction We discuss a variant of general Integer Programming (IP) with additional sign restrictions on the coefficients. With $\mathbb{Z}_+$ denoting the set of non-negative integers, the *non-negative integer programming problem* IP_+ is the following optimization problem:

Input: A matrix $A \in \mathbb{Z}_+^{m\times n}$, vectors $b \in \mathbb{Z}_+^m$ and $c \in \mathbb{Z}_+^n$.
Task: Decide that the problem is infeasible or find an optimal solution x^* to the optimization problem $\min \left\{ \sum_{k=1}^n c_k x_k \mid Ax = b, x \in \mathbb{Z}_+^n \right\}$.

The feasibility version of IP_+, i.e., the decision problem whether $Ax = b, x \in \mathbb{Z}_+^n$ has at least one solution, is denoted by f-IP_+.
The defining feature of IP_+ is the non-negativity of the input data A, b and c. While a variable without sign restriction can be modeled with the help of a difference $x = x^p - x^n$ of two non-negative variables x^p, x^n in a general IP, this substitution is not possible in IP_+ since the resulting system matrix would not be non-negative. However, there are several examples of IP problems that are IP_+ problems, e.g., many problems from graph theory, and restricting the coefficients in IP_+ does not reduce the complexity. It is easy to see that f-IP_+ is strongly NP-complete: The 3-partition problem is NP-complete in the strong sense, see Garey and Johnson [20, Theorem 4.3], and it can be formulated as an IP with non-negative input and polynomial size in the input length as in Dell'Amico and Martello [17].
The non-negativity of the input data implies that the weighted geometric series

$$H_A(z) := \prod_{k=1}^{n} \frac{1}{1 - z^{a(k)}},$$

using the columns $a(k)$ of the matrix A as parameters (Definition 1), is analytic on a domain containing the origin (Proposition 1). Our algorithm for integer programming is based on a representation of H_A as a multivariate power series that is the generating function for the number of solutions to the system $Ax = b$, $x \in \mathbb{Z}^n$.

1.1. Related work Generating functions have received considerable attention after the seminal work on *short rational functions* in Barvinok [2, 3] and Barvinok and Pommershein [5], see also Barvinok [4] for an in-depth presentation. The key result of Barvinok [2] that a generating function can be computed in polynomial time when the dimension is fixed inspired several authors to use Barvinok's short rational functions for solving IP (see De Loera [14] and De Loera et al. [16, 15]) integer optimization of polynomials (De Loera et al. [30]), or finding extended formulations (Köppe et al. [25]). In addition, the algorithms to compute short rational functions have been improved in theory by, e.g., Köppe [23] and powerful software is available, e.g., the LattE package [24].

Our approach shares several techniques with the aforementioned work such as binary search for the optimum by sequentially solving feasibility problems as in De Loera et al. [16]. However, the generating function H_A is not of the Barvinok type. As pointed out in Beck [7], the study of H_A can instead be traced back to Euler [18] who already describes the connection to the number of integer solutions to a system of linear equations. A major difference to Barvinok's generating function is immediately apparent: H_A is already given in a "short" form, while in the Barvinok setting the main effort is to compute the short rational function. In addition, H_A is defined in m variables linking it to the dual space instead of the primal, n-dimensional space, cf. Lasserre and Zeron [29] or Lasserre [27] for a detailed discussion.

Similar to Barvinok's short rational functions, the weighted geometric series has been used in the context of counting solutions (see Baldoni-Silva et al. [1], Beck [7], Beck and Robins [8], and Lasserre and Zeron [26, 29]) and solving integer programs via the closely related Z-transform in Lasserre and Zeron [28], but has overall received less attention. The connection of H_A to multivariate residues has already been observed by Brion and Vergne [13, 12] and Beck [6]. Bertozzi and McKenna [9] study multivariate residues and Cauchy's formula for analytic functions in an applied setting and discuss difficulties of the theory in several complex variables in detail. Using an algebraic perspective on path integrals, the authors of Hirai et al. [21] study the algorithmic complexity of counting integer points and apply their results to hypergraph matchings.

While in the above approaches the generating function is studied mainly from an algebraic perspective, our line of argumentation relies on the properties of H_A as an analytic function. The main idea is to compute a certain coefficient in the series representation of H_A via a multivariate Cauchy formula for complex path integrals. This has been observed by Brion and Vergne [13], Beck [6], and Lasserre and Zeron [28] as well, but only for the standard Cauchy formula on circular paths. We generalize Cauchy's formula to arbitrary multi-paths. This way, the path integrals can be evaluated on *shortest* paths inside the unit disk that have a higher numerical stability than simple circular paths and thus facilitate practical computations. The possibility to pre-process the paths of integration in numerical complex analysis was originally observed by Bornemann [10] and Bornemann and Wechslberger [11] for the efficient computation of derivatives.

1.2. Contribution and structure Our contribution includes the following four main aspects: Firstly, we give a short, self-contained presentation of the generating function technique that does not rely on the advanced results by Barvinok and the other authors mentioned above. In particular, we do not need any algebraic reasoning or formal power series arguments. Instead, we apply convergence results for analytic functions to a generalization of the geometric series.

Secondly, we introduce the new concept of multi-paths and use it to prove a multi-path version of Cauchy's integral formula. While this result is a rather straightforward generalization of Cauchy's formula for circular paths, it can to our knowledge not been found in literature.

Thirdly, based on the aforementioned theoretic exposition, we identify the path of integration as an important degree of freedom for improving the numerical stability in practical computations.

Finally, we implement the method, demonstrate the algorithmic ideas in numerical experiments, and verify our findings numerically.

This article is organized in the following way: In Section 2, the technical Proposition 1 carries the generating function idea from formal power series to holomorphic functions. It implies that Cauchy's integral formula is applicable. We give a new, general version of the formula in Section 2.1.

At the beginning of Section 3, we reformulate f-IP$_+$ in the form of an integral inequality. We use this feasibility criterion to state a new algorithm for IP$_+$. Possibilities for the improvement of the numerical performance of the algorithm are in the focus of Section 3.2. In particular, changing the path of integration is recognized as a promising concept.

Subsequently, these ideas are assessed numerically in Section 4. We use two different test sets: An easier set of randomly generated knapsack instances to illustrate the path optimization idea and a set of hard knapsack instances from Pisinger [31] to show its benefits for the computational performance. All problem data are given in the appendix. We conclude by pointing to potential improvements of the method and future research directions in Section 5.

2. The analytic framework The set $D(\mathbf{0},\mathbf{1}) := \bigtimes_{j=1}^{m}\{z_j \in \mathbb{C} \mid |z_j| < 1\} \subseteq \mathbb{C}^m$ is called the (open) *m-dimensional unit polydisk.* The choice of m for the number of variables is no coincidence since the number of rows m in the constraint matrix A of an IP$_+$ instance defines the number of complex variables to be considered for the method.

Definition 1. Let $A = (a(1),\ldots,a(n)) \in \mathbb{Z}_+^{m\times n}$ be a matrix with column vectors $a(1),\ldots,a(n)$ such that $a(k) \neq \mathbf{0}$ for all k. The function $H_A : D(\mathbf{0},\mathbf{1}) \to \mathbb{C}$ in m complex variables defined by

$$H_A(z) := \prod_{k=1}^{n} \frac{1}{1 - z^{a(k)}} \tag{1}$$

is called the (multivariate) *weighted geometric series* with weight A.

In the definition, we have used the multi-index notation $z^{a(k)} = \prod_{j=1}^{m} z_j^{a(k)_j}$. The condition on the columns of A in particular implies that the denominators of the factors in (1) are non-zero. As the name suggests, H_A is closely related to the well-known univariate geometric series: it is the product of n geometric series which are weighted with the columns of the matrix A.

Proposition 1. *For every $A \in \mathbb{Z}_+^{m\times n}$ with columns $a(k) \neq \mathbf{0}$, $k \in \{1,\ldots,n\}$, and every $z \in D(\mathbf{0},\mathbf{1}) \subseteq \mathbb{C}^m$, the series representation*

$$H_A(z) = \sum_{\nu\in\mathbb{Z}_+^n} \prod_{k=1}^{n} \left(z^{a(k)}\right)^{\nu_k} = \sum_{\nu\in\mathbb{Z}_+^n} z^{\nu_1 a(1)+\ldots+\nu_n a(n)} \tag{2}$$

is valid and the series converges uniformly on every compact subset of $D(\mathbf{0},\mathbf{1})$.

In particular, for every $z \in D(\mathbf{0},\mathbf{1})$ the series $H_A(z)$ converges unconditionally of the order of summation and there is a rearrangement of the series in the form

$$H_A(z) = \sum_{\kappa\in\mathbb{Z}_+^m} \eta_\kappa z^\kappa, \tag{3}$$

where

$$\eta_\kappa = \eta_\kappa(A) = \left|\left\{\nu \in \mathbb{Z}_+^n \mid \sum_{k=1}^{n} a(k)\nu_k = \kappa\right\}\right| \tag{4}$$

are non-negative integers.

A proof of Proposition 1 is provided in Appendix A. As we do not consider formal power series, the unconditional convergence is important when dealing with a series whose index set is not $\mathbb{N}$ but some other countable set. For instance, the notation $\sum_{\kappa\in\mathbb{Z}_+^m} \eta_\kappa z^\kappa$ is only reasonable if the value of the series does *not* depend on the order of summation. In this case, a rearrangement can be chosen arbitrarily, e.g., a diagonal scheme. This is why the dimensions of the multi-indices ν in (2) and κ in (3) are not the same. In addition, using the uniform convergence of H_A to interchange differentiation and summation shows that H_A is a holomorphic function in m variables on $D(\mathbf{0},\mathbf{1})$, cf. Hörmander [22] for an introduction on the topic.

It is important to emphasize that the equality in (3) does not only hold in the sense of formal power series, but as an equality for (holomorphic) functions. This is crucial for Section 3 and is a main difference to algebraic solution methods for IP cited in Section 1.1. Also, Proposition 1 does in general not hold if A has negative entries. If negative exponents can occur, H_A is not longer a power series, but a Laurent series. While algebraic arguments apply to Laurent series as well, it is unclear whether

the resulting series converges at all. As pointed out by Bertozzi and McKenna [9], the geometry of singularities in $\mathbb{C}^m$ is considerably more complicated than in $\mathbb{C}$ and makes the multivariate Cauchy formula more technical. Although certain cases involving negative input data can be settled in the framework of analytic functions (e.g., dealing with removable singularities as described in Lasserre and Zeron [26]) we use the non-negativity as a sufficient condition for convergence.
We change the focus back to integer programming and link the above to the feasibility problem f-IP$_+$. We have restricted the analysis of H_A to matrices $A \in \mathbb{Z}_+^{m\times n}$ that do not contain an all-zero column. Fortunately, this (necessary) assumption means no restriction for the study of f-IP$_+$ because the variable x_k can be removed from the problem if the k-th column of the matrix is zero. The following proposition links f-IP$_+$ and Proposition 1 by interpreting the non-negative right-hand side b of the feasibility problem as a multi-index in the series representation.

Proposition 2. *Let $A \in \mathbb{Z}_+^{m\times n}$ with columns $a(k) \neq \mathbf{0}$, $k \in \{1,\dots,n\}$, and $b \in \mathbb{Z}_+^m$ be given. The system $Ax = b$ has a solution $x \in \mathbb{Z}_+^n$ if and only if $\eta_b \geq 1$ in the series representation $H_A(z) = \sum_{\kappa\in\mathbb{Z}_+^m} \eta_\kappa z^\kappa$ given by* (3)*. Moreover, in this case the system has exactly η_b solutions.*

This proposition essentially only rephrases (4) for $\kappa = b \geq 0$. The coefficient η_b is defined as the cardinality of the set of solutions. If $\eta_b = 0$, the set has cardinality zero, i.e., the system is infeasible. In other words, Proposition 2 states that the weighted geometric series is the *generating function* for the number of solutions to the system $Ax = b$, $x \in \mathbb{Z}_+^n$. As pointed out in the introduction, the link between systems of linear diophantine equations and power series is not new. However, we have proved Proposition 2 in the sense of analytic functions and can therefore use powerful tools from the theory of several complex variables to compute the coefficient η_b *without* expanding the series (or partially expanding it using algebraic reasoning).

2.1. Cauchy's integral formula for multi-paths

Holomorphic functions in $\mathbb{C}^m$ can be represented by m-dimensional power series, see Hörmander [22]. As in the one-dimensional case, this observation yields a connection between the coefficients of a power series and the evaluation of path integrals. For the application we have in mind, we first need to generalize the classical multivariate version of Cauchy's formula, see Rudin [32], to *multi-paths*, a new concept tailored specifically to our needs. Following standard notation, a univariate *path* in Definition 2 is a piecewise continuously differentiable complex curve $\gamma : [a,b] \to \mathbb{C}$, with *range* $[\gamma] := \{z \in \mathbb{C};\ z = \gamma(t) \text{ for some } t \in [a,b]\}$, and *index* given by $\operatorname{Ind}_\gamma(z) = \frac{1}{2\pi i}\int_\gamma \frac{1}{\zeta - z}\,\mathrm{d}\zeta$ for all $z \in \mathbb{C}\setminus[\gamma]$, see Rudin [33, Chapter 10] for an introduction to univariate path integrals.

Definition 2 (Multi-Paths). Let Ω be a subset of $\mathbb{C}^m$.

1. A *multi-path* $\Gamma := (\gamma_1,\dots,\gamma_m)$ is a (ordered) tuple of paths $\gamma_j : [a_j,b_j] \to \mathbb{C}$. The paths γ_j are called the *component paths* of Γ for $j \in \{1,\dots,m\}$.
2. A multi-path is called *closed* if $\gamma_j(a_j) = \gamma_j(b_j)$ for all $j \in \{1,\dots,m\}$, i.e., all of its component paths are closed.
3. The *index* $\operatorname{Ind}_\Gamma$ of a closed multi-path Γ is the product of the indices of its component paths

$$\operatorname{Ind}_\Gamma(z) := \prod_{j=1}^m \operatorname{Ind}_{\gamma_j}(z_j)$$

for all $z \in \bigtimes_{j=1}^m \mathbb{C}\setminus[\gamma_j]$.

4. If $\Gamma = (\gamma_1,\dots,\gamma_m)$ is a multi-path and $f : \mathbb{C}^m \to \mathbb{C}$ is a continuous function on $\bigtimes_{j=1}^m [\gamma_j]$, then

$$\int_\Gamma f(z)\,\mathrm{d}z := \int_{\gamma_m}\dots\int_{\gamma_1} f(z_1,\dots,z_m)\,\mathrm{d}z_1\dots\mathrm{d}z_m. \tag{5}$$

Fubini's theorem shows that the value of the integral in (5) is independent of the order of integration and, thus, the integral for multi-paths is well-defined. We are ready to state the technical key result for solving f-IP$_+$ with Cauchy's formula.

PROPOSITION 3 **(Cauchy's integral formula for multi-paths)**. *Let* $\Gamma = (\gamma_1, \ldots, \gamma_m)$ *be a closed multi-path and let convex open sets* $\Omega^j \subseteq \mathbb{C}$ *with* $[\gamma_j] \subseteq \Omega^j$ *for all* $j \in \{1, \ldots, m\}$ *be given. If* f *is a holomorphic function on* $\bigtimes_{j=1}^{m} \Omega^j$*, then*

$$\operatorname{Ind}_\Gamma(z) f(z) = \frac{1}{(2\pi i)^m} \int_\Gamma \frac{f(\zeta)}{\prod_{j=1}^{m}(\zeta_j - z_j)} \,\mathrm{d}\zeta$$

for all $z \in \bigtimes_{j=1}^{m} \Omega^j \setminus [\gamma_j]$.

For the proof of Proposition 3 we can adapt the proof of the polydisk version (see Rudin [32]) of Cauchy's integral formula, which is a special case, see Appendix B.

Finally, we can combine Proposition 2 and Proposition 3 to compute the series coefficients of H_A directly, i.e., without expanding the series:

PROPOSITION 4. *Let* $A \in \mathbb{Z}_+^{m \times n}$ *be a matrix with columns* $a(k) \neq \mathbf{0}$ *and let* Γ *be a closed multi-path in* $D(\mathbf{0}, \mathbf{1}) \subseteq \mathbb{C}^m$ *with* $\operatorname{Ind}_\Gamma(\mathbf{0}) \neq 0$*. Then, for every multi-index* κ*, the coefficient* η_κ *of* z^κ *in* (3) *is given by*

$$\eta_\kappa = \frac{1}{(2\pi i)^m \operatorname{Ind}_\Gamma(\mathbf{0})} \int_\Gamma \frac{H_A(\zeta)}{\zeta^{\kappa + \mathbf{1}}} \,\mathrm{d}\zeta. \tag{6}$$

Proof. On the one hand, differentiating under the integral sign in Proposition 1 for H_A at the origin gives

$$\partial^\kappa H_A(0) = \frac{\kappa!}{(2\pi i)^m \operatorname{Ind}_\Gamma(0)} \int_\Gamma \frac{H_A(\zeta)}{\zeta^{\kappa + \mathbf{1}}} \,\mathrm{d}\zeta.$$

Here, we have used that H_A is holomorphic and thus has continuous (first-order) derivatives.

On the other hand, the uniform convergence of H_A allows to interchange differentiation and summation in (3) to obtain

$$\partial^\kappa H_A(0) = \sum_{\lambda \geq \kappa} \frac{\eta_\lambda \ \lambda!}{(\lambda - \kappa)!} z^{\lambda - \kappa} \bigg|_{z=0} = \sum_{\lambda \in \mathbb{Z}_+^n} \frac{\eta_{\lambda + \kappa} \ (\lambda + \kappa)!}{\lambda!} z^\lambda \bigg|_{z=0} = \kappa! \ \eta_\kappa. \quad \square$$

Propositions 3 and 4 are the multi-path extensions of the similar results in Brion and Vergne [13], Beck [6], Lasserre and Zeron [29], and Hirai et al. [21].

3. Using Cauchy's formula to solve IP$_+$ In the last section, we have connected the solution of f-IP$_+$ to the coefficients of the weighted geometric series. We combine the interpretation of η_b given in Proposition 2 with Proposition 4 to obtain the following result for f-IP$_+$.

PROPOSITION 5. *Let* Γ *be a closed multi-path in* $D(\mathbf{0}, \mathbf{1})$ *with* $\operatorname{Ind}_\Gamma(\mathbf{0}) \neq 0$*. Then for every matrix* $A = (a(1), \ldots, a(n)) \in \mathbb{Z}_+^{m \times n}$ *with* $a(k) \neq \mathbf{0}$, $k \in \{1, \ldots, n\}$*, and every* $b \in \mathbb{Z}_+^m$ *the system* $Ax = b$, $x \in \mathbb{Z}_+^n$ *has a solution if and only if*

$$\int_\Gamma \frac{H_A(\zeta)}{i^m \zeta^{b + \mathbf{1}}} \,\mathrm{d}\zeta \geq (2\pi)^m \operatorname{Ind}_\Gamma(\mathbf{0}). \tag{7}$$

The index of a multi-path is a non-negative integer. Therefore, Proposition 4 implies that the left-hand side of (7) is a real number and, hence, that the $\geq$-relation is well-defined. Independently of the choice of A, both sides of (7) are zero if $\operatorname{Ind}_\Gamma(\mathbf{0}) = 0$ since the integrand is holomorphic in this case. This illustrates why only multi-paths with positive index at zero are suitable for our purposes. We have formulated Proposition 5 in terms of the feasibility problem, but it is based on the counting results of Proposition 4. Counting the number of solutions to certain problems is #P-complete, see Valiant[34]. The perfect matching problem on bipartite graphs is one of these problems and has a formulation as IP$_+$. This characteristic has to be taken into account when analyzing the worst-case complexity of our algorithm and other methods that use counting for feasibility checks.

3.1. An integration algorithm for integer programming The possibility to choose an arbitrary multi-path of integration inside $D(\mathbf{0},\mathbf{1})$ in (7) is why we have bothered to formulate all integral theorems for general multi-paths. We analyze this feature in detail in Section 3.2.

For Algorithm 1 we choose a multi-path Γ_{m+1} in the $m+1$-dimensional unit polydisk and denote its restriction to the first m components by Γ_m. To keep the notations simple, we assume that $\mathrm{Ind}_{\Gamma_{m+1}}(\mathbf{0}) = 1$ in what follows.

A simple choice for Γ_{m+1} is of course the boundary of a smaller polydisk inside $D(\mathbf{0},\mathbf{1})$, i.e., a *multi-circle* $\circlearrowleft = (\circlearrowleft_1, \ldots, \circlearrowleft_{m+1})$ given by $\circlearrowleft_k \colon [0, 2\pi] \to \mathbb{C}$ with $\circlearrowleft_k(t) = r_k e^{it}$, where $0 < r_k < 1$ for all $k \in \{1, \ldots, m+1\}$. This multi-path is a mathematically positive rotation with radius r_k around zero in every component and $\mathrm{Ind}_{\circlearrowleft_k}(0) = 1$ holds for each of the component paths, implying $\mathrm{Ind}_{\circlearrowleft}(\mathbf{0}) = 1$ for the multi-path. Therefore, this factor is one for $\circlearrowleft$. The radii r_k in every component can either be chosen independently of the instance or can be computed from the problem data. For the numerical test on knapsack instances $a^T x = b$ in Section 4 we will use $r_k \in \mathrm{argmax}_{t \in (0,1)} |\frac{H_a(t)}{t^{b+1}}|$, i.e., the radius that (approximately) minimizes the absolute value of the integrand on the real interval $(0, 1)$.

Algorithm 1 uses binary search over a sequence of feasibility instances (Line 7) to solve an optimization problem. We have formulated Algorithm 1 to find the optimal objective value Z of an IP_+ instance. The optimal solution vector x^* can then be found by binary search as well, but we have omitted this step from the presentation to keep the focus on the most relevant and new ideas. The binary search loop needs only polynomially many iterations, so that the worst-case complexity is dominated by the integral evaluations.

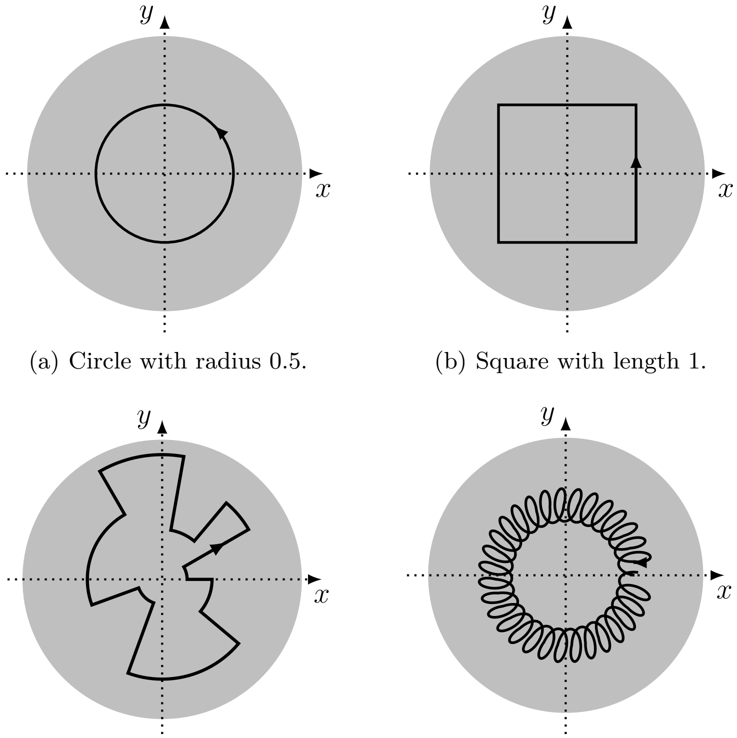
Algorithm 1 Multi-path integral algorithm for IP_+

```
Require: $A \in \mathbb{Z}_+^{m \times n}$, $b \in \mathbb{Z}_+^m$ and $c \in \mathbb{Z}_+^n$, multi-path $\Gamma_{m+1}$ in $D(\mathbf{0},\mathbf{1})$ with $\mathrm{Ind}_{\Gamma_{m+1}}(\mathbf{0}) = 1$
 1: if $\int_{\Gamma_m} \frac{H_A(\zeta)}{i^m \zeta^{b+\mathbf{1}}} \, \mathrm{d}\zeta < (2\pi)^m$ then
 2:     return problem infeasible
 3: end if
 4: set $K \leftarrow \max_{k=1,\ldots,m} b_k$, $U \leftarrow K \sum_{k=1}^n c_k$ and $Z \leftarrow 0$
 5: while $Z < U$ do
 6:     set $M \leftarrow \lfloor \frac{Z+U}{2} \rfloor$
 7:     set $D \leftarrow \begin{pmatrix} A & \mathbf{0} \\ c^\mathsf{T} & 1 \end{pmatrix}$ and $d \leftarrow \begin{pmatrix} b \\ M \end{pmatrix}$
 8:     if $\int_{\Gamma_{m+1}} \frac{H_D(\zeta)}{i^{m+1} \zeta^{d+\mathbf{1}}} \, \mathrm{d}\zeta \geq (2\pi)^{m+1}$ then
 9:         $U \leftarrow M$
10:     else
11:         $Z \leftarrow M + 1$
12:     end if
13: end while
14: return optimal objective $Z$
```

In Line 1 and Line 8 of the algorithm, an integral along the multi-paths Γ_m and Γ_{m+1}, respectively, has to be evaluated to decide whether the problem under consideration is feasible or not. This does, however, *not* imply that the value of the integral in (7) has to be calculated exactly. To the contrary, stopping the computation early is in particular important in view of the generally higher complexity of counting. Fortunately, the fact that the value of the integral is known to be an integer multiple of $(2\pi)^m$ or $(2\pi)^{m+1}$, respectively, can be used to improve numerical run times.

3.2. Optimizing the path of integration From an analytical perspective the multi-paths in Proposition 5 are equivalent. This observation is generally true for path integrals and their generalization to multi-paths, see Rudin [33, Lemma 10.39]. The value of $\mathrm{Ind}_\Gamma(\mathbf{0})$, and hence the whole integral expression in (7), does not change when the component paths are continuously deformed around zero, i.e., if the paths are homotopic.

Therefore, countless more or less arbitrary choices are possible for the path of integration. Figure 1 depicts the range of four different paths inside the unit disk of $\mathbb{C}$. The circular path in (a) is a smooth path that has the advantage of an easy parametrization. The choice of the radius affects the numerical stability of the computation and the optimal radius of a circle path in numerical evaluations of (Cauchy) integrals is studied extensively from both the theoretical and computational perspective in Bornemann [10].



(a) Circle with radius 0.5.

(b) Square with length 1.

(c) A path with adaptive radii.

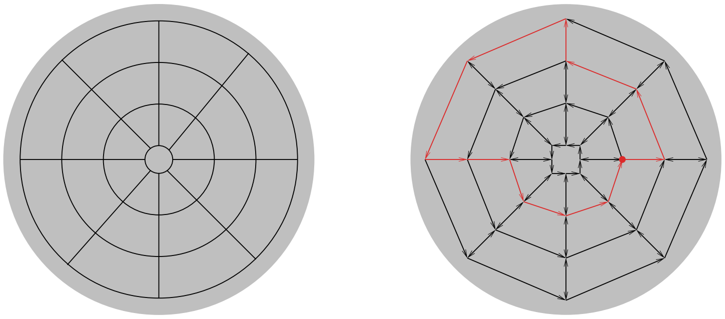
(d) A path with loops.

FIGURE 1. Four paths Γ in the unit disk $\{x+iy \in \mathbb{C} \mid x^2+y^2<1\}$ with $\mathrm{Ind}_\Gamma(0)=1$.

Another obvious choice is a square (or more general: a polygon) inside the domain of holomorphy, see Figure 1(b). The square path is non-smooth, but can be evaluated as a piecewise integral. The square path has the advantage that along each side either the real part or the imaginary part of the variable is constant, which simplifies the calculations. A square path is, for instance, used for a similar integration problem in De Loera et al. [15].

Depending on the integration problem under consideration, other choices for the path may be useful. The two other sketches in Figure 1 exemplify that the path does not have to be the boundary of a convex (c) or connected (d) domain, respectively, i.e, it may have loops.

The path in Figure 1(c) is an example of a simple strategy to optimize the path of integration. The idea of the method is to start a circle around the origin on the real axis and change the radius of the circle whenever the absolute value of the integrand becomes too large. The circular arcs are then connected by straight line segments. This simple path adaption strategy yields a shape that is more



(a) The intersection points $V_{R,P}$ define the nodes of a digraph.

(b) Finding a shortest path on the grid digraph.

FIGURE 2. The shortest path method for determining the path of integration.

flexible than the simple circle (a), but still easy to control. The idea is analyzed in Friedrich [19, Chapter 7].

3.3. The shortest path of integration Choosing an advantageous path of integration improves the numerical quality of the integral computations in Algorithm 1. In what follows, we restrict our analysis to paths in $\mathbb{C}$ instead of multi-paths. However, the method generalizes to multi-paths as the integral along a multi-path is defined as the iterated integral along its component paths. In Bornemann and Wechslberger [11], the possibility to improve the numerical computation of complex integrals is extensively studied in the class of so-called *grid paths*, i.e., paths that consist of the edges of a regular grid. This procedure is combined with a grid adaptation algorithm to allow a local refinement of the integration path. Inspired by this concept, we present a path optimization algorithm that is tailored specifically to the unit disk.

The starting point for optimizing the path of integration is the condition number of the numerical quadrature problem that we use as a measure for the practical computability. For the path integral in Proposition 4, this condition number is given by (see Bornemann and Wechslberger [11])

$$\frac{\int_\Gamma |H_A(\zeta)||\zeta|^{-b-\mathbf{1}}\,\mathrm{d}\zeta}{|\int_\Gamma H_A(\zeta)\zeta^{-b-\mathbf{1}}\,\mathrm{d}\zeta|}. \tag{8}$$

The path independence of holomorphic functions implies that the denominator in (8) is independent of the path of integration Γ as long as it lies inside the unit disk and $\mathrm{Ind}_\Gamma(0)$ remains unchanged. Therefore, we can minimize the condition number by minimizing the numerator integral in (8). We do so by approximating the integral on a grid defined in the following way: Fix two integers $R, P > 1$. First, we consider circles with radii $\frac{r}{R}$ for $r \in \{1, \dots, R-1\}$. Then, we intersect these circles with line segments connecting the origin with the points $\frac{R-1}{R}e^{it_p}$, where $t_p = \frac{p2\pi}{P}$ for $p \in \{1, \dots, P\}$. Thus, all intersection points constructed this way are $V_{R,P} = \{\frac{r}{R}e^{it_p} \mid r = 1, \dots, R-1,\ p = 1, \dots, P\}$. This construction is displayed in Figure 2(a).

Next, we use $V_{R,P}$ as nodes to construct a digraph $(V_{R,P}, A)$. We add the arcs $(\frac{r}{R}e^{it_p}, \frac{r}{R}e^{it_{p+1}})$, $(\frac{r+1}{R}e^{it_p}, \frac{r}{R}e^{it_p})$, and $(\frac{r}{R}e^{it_p}, \frac{r+1}{R}e^{it_p})$ for $r \in \{1, \dots, R-2\}$ and $p \in \{1, \dots, P-1\}$, i.e., all of the connections between two neighboring interception points. However, by construction the spoke arcs of the digraph can be traversed in both directions, whereas the rim arcs can only be traversed in mathematical positive direction, i.e., counterclockwise, see Figure 2(b). We define weights w_a for each arc $a = (u, v)$ of the digraph $(V_{R,P}, A)$ by

$$w_a = \frac{|v-u|}{2}\left(|H_A(v)||v|^{-b-\mathbf{1}} + |H_A(u)||u|^{-b-\mathbf{1}}\right).$$

Thus, w_a approximates the integral along the arc a by the simple rectangular rule. Finally, we choose a node v_1 among the nodes $\{\frac{1}{R}, \ldots, \frac{R-1}{R}\}$ on the real axis in such a way that the value of $|H_A(v_1)||v_1|^{-b-\mathbf{1}}$ is (approximately) minimal and introduce a copy v_1' of the node v_1 that has the same adjacent nodes. We can now use Dijkstra's algorithm to find a shortest $v_1 - v_1'$ path in this digraph and approximately minimize the condition number of the integral. The resulting path circles the origin exactly once in mathematical positive direction, i.e., it has index one at the origin. The idea is summarized in Figure 2(b).

4. Proof-of-concept computations We analyze the solution algorithm with some numerical experiments. For the computational study, we focus on the feasibility version f-IP$_+$. Of course, the proposed solution algorithm for the optimization problem itself needs to re-run multiply feasibility problems, so that our observations can be generalized to the optimization problem. Moreover, we only consider knapsack instances here, which in particular is the only setting that allows us to plot resulting paths in a two-dimensional figure. It is clear that for the general problem, we would have to solve the integration procedure once per row of the constraints matrix. This, however, only makes sense if we have an optimized code for the one-dimensional quadrature problem and is therefore not in the focus of this early numerical experiment.

All computations were run on a MacBook Pro with 8 GB of RAM. The code of Algorithm 1 was implemented in Python and the Python library quad was used for adaptive numerical quadrature within the code.

Our first test set of problems, given in Table 1, consists of a total of 40 knapsack problems. Among them, there are 20 random simple instances s1-s20 with right-hand sides in $[1, 50]$ and 20 random larger instances $l1 - l20$ with right-hand sides in $[1, 500]$. All 40 instances are knapsacks in 5 variables with random coefficients in $[1, 20]$.

In Table 1, we give the right-hand side b for each instance and report the actual number of feasible solutions (sol) in the next column, checked via brute-forcing all possible combinations. The instances' coefficients a are given in Table 4 in the appendix. We then give the optimized radius for the circle $\circlearrowleft$ as explained in Section 3.1. The next two columns show the condition number (cond circ) of the circular path with optimized radius and the total computation time (time circ) when using this path of integration. The results for the shortest path of integration follow in the next four columns: We report the condition number (cond SP) when using this path, the total computation time (time SP), the time for the actual quadrature (int SP), and the time for the Dijkstra pre-solving step (dijk SP) as described in Section 3.3, i.e., the last two columns sum up to the "time SP" column.

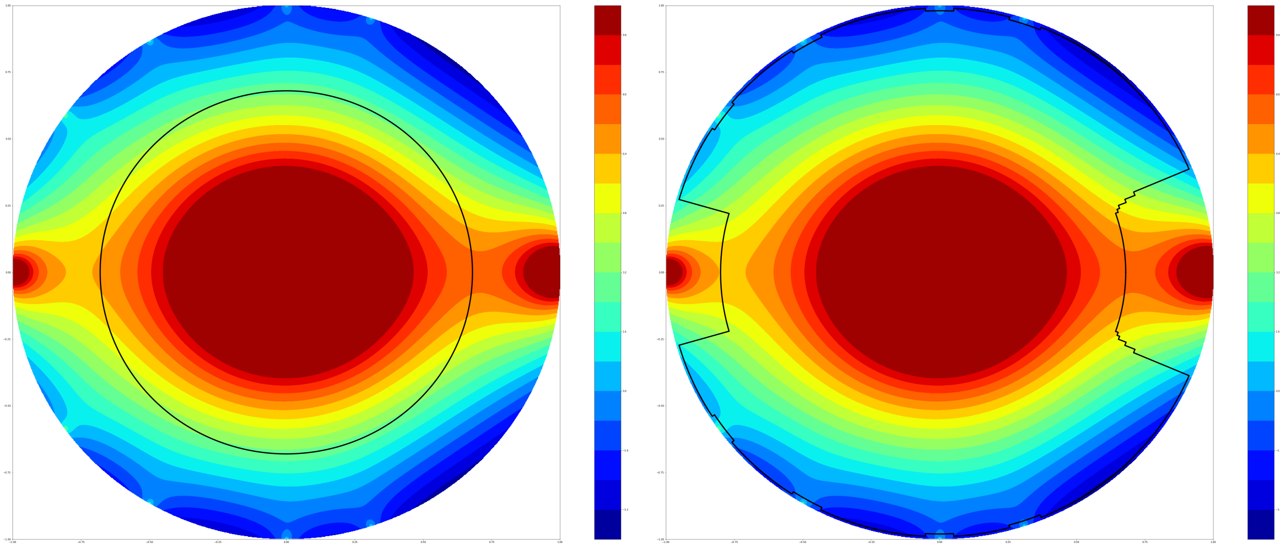
We can state that all instances have been solved quickly via the integration algorithm and we have thus successfully implemented a new algorithm for counting the number of feasible solutions and checking feasibility. First, we compare the condition numbers of the circular paths with the shortest path of integration. For every single instance, the condition number is improved by using the optimized path of integration SP instead of the circular approach. However, for instances with a radius close to one, the difference (almost) vanishes. This shows, in principle, the success of the shortest path pre-solving for achieving better condition numbers.

In addition to the data given above, we can illustrate the concept of path optimization by plotting two instances and the resulting paths of integration. In Figures 3 and 4 we have plotted the absolute value of the integrand function $|\frac{H_a(t)}{t^{b+1}}|$ on a logarithmic scale inside the unit disk. Note that the function values are cut off near the poles for the graphical representation. The color coding ranges from dark blue for small values to dark red for high values. We can clearly see the poles of the functions at the origin (a pole of order $b + 1$ dominating the plots) and poles of smaller order on the boundary of the unit disk. Among the poles on the boundary, the real pole at $z = 1$ is always of highest order because 1 always is a solution to $z^p = 1$, while the distribution of the other roots depend on the value of p.

Table 1. Results for random instances. All times in seconds, computations run on a MacBook Pro with 8GB RAM.

instance	RHS b	sol	radius	cond circ	time circ	cond SP	time SP	int SP	dijk SP
s1	47	14	0.931	51.7	0.015	22.0	0.078	0.017	0.061
s2	2	1	0.618	3.0	0.004	2.1	0.011	0.004	0.007
s3	40	10	0.923	39.0	0.015	14.5	0.015	0.014	0.000
s4	14	0	0.855	9.8	0.009	3.8	0.033	0.009	0.025
s5	48	14	0.933	47.4	0.015	18.7	0.019	0.015	0.003
s6	48	17	0.931	56.5	0.014	24.8	0.022	0.014	0.007
s7	42	7	0.928	33.2	0.015	10.7	0.015	0.015	0.000
s8	19	1	0.881	12.8	0.009	4.5	0.130	0.009	0.121
s9	46	69	0.924	125.4	0.014	85.0	0.035	0.014	0.020
s10	33	26	0.904	57.9	0.011	32.5	0.519	0.011	0.508
s11	32	5	0.912	26.8	0.014	10.0	0.172	0.011	0.161
s12	34	46	0.904	89.3	0.012	56.9	0.157	0.012	0.144
s13	17	0	0.884	9.0	0.009	3.6	0.081	0.009	0.072
s14	2	0	0.668	2.2	0.004	1.6	0.077	0.004	0.073
s15	22	19	0.867	43.8	0.011	24.9	0.017	0.013	0.003
s16	6	1	0.786	4.4	0.007	2.3	0.027	0.006	0.021
s17	50	56	0.928	117.1	0.014	70.0	0.020	0.014	0.006
s18	5	1	0.743	4.4	0.006	2.3	0.013	0.006	0.007
s19	18	4	0.865	17.5	0.008	7.3	0.254	0.009	0.245
s20	21	0	0.901	9.6	0.010	3.3	0.044	0.014	0.030
l1	226	2908	0.981	3767.5	0.041	3307.1	0.178	0.040	0.138
l2	138	1063	0.969	1449.7	0.038	1411.5	0.037	0.037	0.000
l3	279	28483	0.983	37633.3	0.068	34360.2	0.064	0.059	0.005
l4	261	3214	0.983	4072.0	0.061	3606.3	0.194	0.060	0.133
l5	71	175	0.947	281.2	0.022	214.2	0.023	0.022	0.000
l6	489	36584	0.990	44794.6	0.083	43892.5	0.370	0.075	0.295
l7	124	341	0.967	508.9	0.024	486.5	0.507	0.022	0.484
l8	145	689	0.971	1288.1	0.035	1200.7	0.040	0.036	0.003
l9	423	15985	0.989	19705.9	0.067	19085.7	0.127	0.072	0.055
l10	424	35948	0.989	43871.3	0.079	42813.5	0.158	0.102	0.057
l11	160	2065	0.973	2651.7	0.037	2636.0	0.037	0.036	0.000
l12	270	16933	0.983	21138.3	0.092	19065.5	0.121	0.087	0.034
l13	87	670	0.953	1291.6	0.034	1202.5	0.026	0.026	0.001
l14	337	6369	0.986	7904.4	0.063	7538.3	0.069	0.069	0.001
l15	255	29940	0.982	44265.5	0.051	39623.6	0.391	0.048	0.344
l16	375	9547	0.988	11831.3	0.093	11355.1	0.073	0.067	0.006
l17	215	2438	0.979	3941.4	0.037	3471.6	0.095	0.039	0.056
l18	293	4912	0.984	6664.3	0.085	6101.8	0.080	0.078	0.002
l19	157	408	0.973	629.2	0.039	538.4	0.037	0.037	0.000
l20	197	839	0.978	1190.8	0.037	1010.5	0.047	0.039	0.008

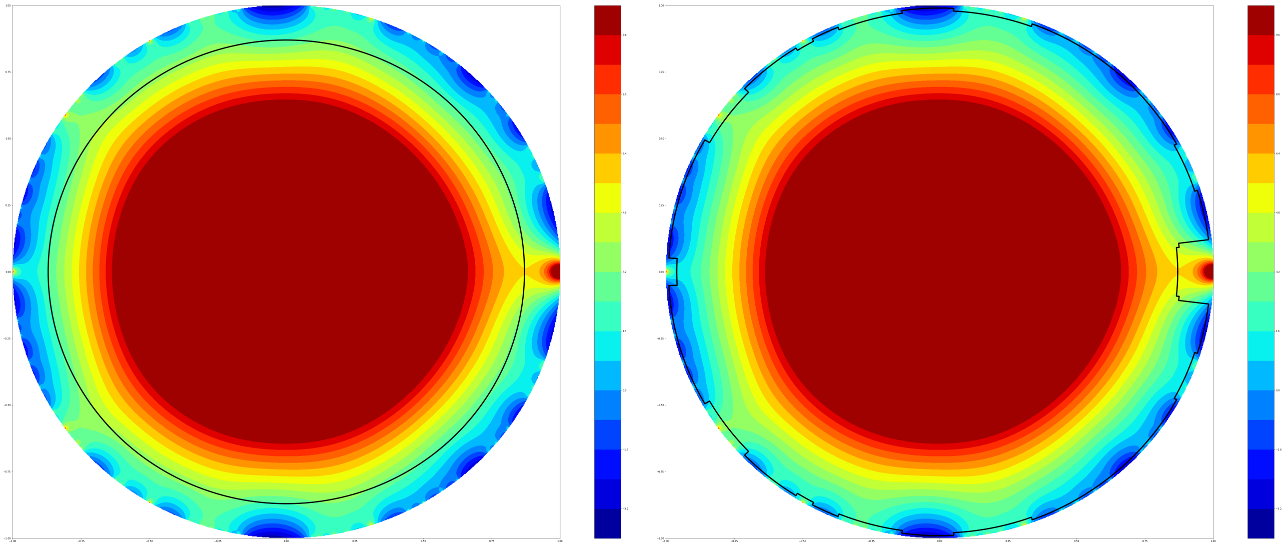
Minimizing the condition number translates to finding a path around the origin that avoids the regions where the (absolute) function values are high. In both Figures 3 and 4 the left image (a) shows the circular path, which finds the best *bypass* point on the positive real axis, but does not consider the remaining values. In contrast, the optimized paths of integration displayed on the right side (b) have a more complex geometry and can find paths on a lower level, i.e., with a better overall condition. Recall that the paths are constructed in such a way that they stay inside the unit disk although the absolute function values are generally smaller outside. We would need algebraic arguments to get the right value for the integral otherwise, as some of the poles on the unit circle would then be “counted” in Cauchy’s formula.



(a) Circular path for the example problem.

(b) Shortest path for the example problem.

FIGURE 3. Paths of integration for the knapsack instance with $a = [2,2,2,2,3,4,5]$ and $b = 10$.



(a) Circular path for the example problem.

(b) Shortest path for the example problem.

FIGURE 4. Paths of integration for knapsack instance with $a = [3,5,10,18,13,17,14]$ and $b = 21$.

Continuing our discussion of the numerical results in Table 1 with the computations times, we observe mixed results: For the smaller instances s1-s20, the computations using the circular path are generally faster. Here the additional computation time for the pre-solving step does not pay off in a better overall run-time. Integration times only along the shortest path (int SP) are similar to the integration times along the circle. For the larger instances l1-l20, we observe that the shortest path pre-solving is more beneficial. In particular, there are some instances (l2, l3, l13, l16, l18, and l19) where the total computation time is smaller than for the circular path.

All in all, we conclude that the shortest path pre-solving is indeed working, but the computational overhead of the shortest path pre-solving is to large on the comparably easy random instances.

We therefore continue by discussing another set of knapsack instances that has been constructed to be hard with the algorithms described in Pisinger [31]. Theses instances, labeled p1-p15 and given in Table 2, can be downloaded from the accompanying online repository given in the reference. We consider the first 15 problems in the file `knapPI_1_50_1000.csv` here, which have 50 variables and coefficients in the interval $[0, 1000]$. The coefficient vectors of the instances are given in Table 3.

TABLE 2. Results for hard knapsack instances from [31]. All times in seconds, computations run on a MacBook Pro with 8GB RAM.

	b	sol	radius	cond circ	time circ	cond SP	time SP	int SP	dijk SP
p1	995	544429	0.9896	679105	1.55	598424	1.91	1.60	0.31
p2	997	28060	0.9913	49093	1.26	35165	1.84	1.59	0.24
p3	970	220560	0.9910	271276	1.54	255341	2.24	1.96	0.28
p4	976	5502	0.9907	22992	1.27	10870	2.41	2.14	0.27
p5	1361	152417	0.9930	208004	1.27	174708	1.83	1.65	0.18
p6	1370	715681393	0.9911	779061215	1.88	778054647	1.93	1.69	0.24
p7	1765	1439298	0.9935	2161961	2.60	1585134	2.35	2.17	0.18
p8	2030	60979453	0.9936	68148154	2.60	65942681	2.50	2.31	0.18
p9	2152	4849303	0.9940	6135258	2.72	5276790	2.22	2.07	0.15
p10	2745	50856154	0.9951	56380123	4.99	54591560	2.74	2.62	0.12
p11	2397	42417243268	0.9937	45224037244	3.85	45188849483	2.73	2.54	0.19
p12	3676	13244852	0.9961	17944002	4.99	14247955	3.45	3.36	0.09
p13	3194	13466832804	0.9950	14292356870	4.95	14246988700	3.42	3.30	0.12
p14	3665	57612978960	0.9955	60858991273	5.77	60791225233	3.46	3.34	0.12
p15	4223	1981736210	0.9961	2122098819	5.74	2080755946	3.61	3.52	0.09

In Table 2, we give the problem identifier, the right-hand side b and the number of feasible solutions. We then report the condition numbers and computation times in the same style as above. First, note that the computations were generally numerically more challenging, which can immediately be seen in the overall computation times.

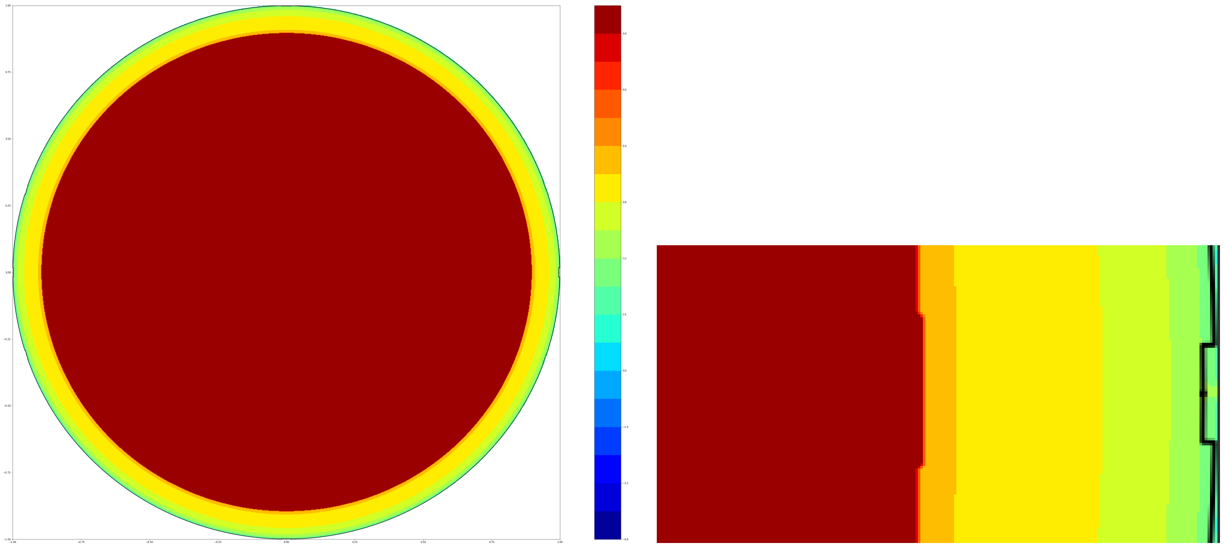
As for the easier instances above, the table shows a reduction of the condition number of every single instance. All but the 6 smallest instances p1-p6 in the table are solved faster on the pre-calculated SP path than on the naive circular path. In addition, the effect becomes more pronounced with growing size of the instances. This comparison includes, of course, the running time for the Dijkstra pre-solving given in the last column of the table. It is also interesting to observe that the times for the Dijkstra run in Tables 1 and 2 are similar, i.e., it scales in a nice way.

We plot instance p10 from the set of hard knapsacks in Figure 5. Although we use a log-log scale for the figure on the left side (a), is hard to recognize any details in the function plot. Moreover, the shortest path of integration is barely visible near the border of the unit disk and almost appears to be a circle with radius near 1. This is, however, not true as the magnification on the right (b) shows. We can state that the exponential growth of the absolute function values near the origin dominates the entire setting.

Clearly, these results show that the concept of path optimization can be beneficial for hard instances, i.e., when the computational overhead of the pre-solve Dijkstra routine is small in comparison to the overall computation time.

5. Conclusion and outlook

Techniques from the field of several complex variables and discrete optimization have been combined in a quadrature algorithm for IP_+ that uses a shortest paths of integration for the Cauchy integrals. The key to the full flexibility of pre-optimized paths is the concept of multi-paths and the extensions of the well-known Cauchy formula. The focus has been mainly on the theory of the general IP_+ setting, which has been supplemented by proof-of-concept computations in one dimension.



(a) The shortest path is almost a circle.

(b) Magnification near the real interval [0.75,1).

Figure 5. The shortest path of integration for instance p10.

Implementing the method involves the use of advanced methods for numerical integration and a careful control of numerical errors. Using the shortest path routine for path optimization, we see promising results when experimenting with the adaptive quadrature implemented in Python, but implementation work is going on. It would be interesting to assess the quality of the algorithm in a broader numerical study. On the theoretical side, both the generalization to arbitrary input data and the worst-case analysis of the algorithm are open research questions.

Appendix A: Proof of Proposition 1 Fix $z \in D(\mathbf{0}, \mathbf{1})$. By the definition of the multi-index and the assumptions on A we have $|z^{a(k)}| < 1$. Thus, each of the factors on the right of (1) can be expanded as a univariate power series at $z^{a(k)} \in \mathbb{C}$. Precisely, we have

$$\frac{1}{1 - z^{a(k)}} = \sum_{\nu_k = 0}^{\infty} \left(z^{a(k)}\right)^{\nu_k}$$

for every $k \in \{1, \ldots, n\}$. As each of the above power series converges absolutely, their product is well-defined and converges to the product of their limits. This yields

$$\prod_{k=1}^{n} \sum_{\nu_k = 0}^{\infty} \left(z^{a(k)}\right)^{\nu_k} = \sum_{\nu \in \mathbb{Z}_+^n} \prod_{k=1}^{n} \left(z^{a(k)}\right)^{\nu_k} = H_A(z).$$

The second equality in (2) is just the definition of the multi-index.

Next, let $K \subseteq D(\mathbf{0}, \mathbf{1})$ be a compact subset of $\mathbb{C}^m$. Because $D(\mathbf{0}, \mathbf{1})$ is open, there is a scalar $r \in (0, 1)$ such that $K \subseteq D(\mathbf{0}, r\mathbf{1})$, the polydisk with radius r in every dimension. This implies $|z^{a(k)}| < r$ for all $z \in K$ and all $k \in \{1, \ldots, n\}$. Using this estimation and the limit of the univariate geometric series, we obtain

$$\sup_{z \in K} \left| \sum_{\nu \in \mathbb{Z}_+^n} \prod_{k=1}^{n} \left(z^{a(k)}\right)^{\nu_k} \right| \leq \sum_{\nu \in \mathbb{Z}_+^n} r^{|\nu|} = \frac{1}{(1-r)^n} < \infty.$$

Turning to the proof of (3), by the absolute convergence we may rearrange the series without changing its value. By setting

$$\eta_\kappa(A) := \left| \left\{ \nu \in \mathbb{Z}_+^n \mid \sum_{k=1}^{n} a(k)\nu_k = \kappa \right\} \right|$$

we obtain (3) from (2) by equating the exponents in both expressions. □

Appendix B: Proof of Proposition 3 We show the proposition by induction on the dimension m. The base case $m = 1$ is settled by the classical univariate Cauchy's formula, see, e.g., Rudin [33, Theorem 10.15].

For the induction step, consider $z \in \bigtimes_{j=1}^{m} \Omega^j \setminus [\gamma_j]$. Fix the m-th variable to $\bar{z}_m \in \Omega^m \setminus [\gamma_m]$ and treat $f(z_1, \ldots, z_{m-1}, \bar{z}_m)$ as a function in $m-1$ variables. Then, the induction hypothesis implies

$$\operatorname{Ind}_\Gamma(z) f(z_1, \ldots, z_{m-1}, \bar{z}_m) = \frac{\operatorname{Ind}_{\gamma_m}(\bar{z}_m)}{(2\pi i)^{m-1}} \int_{\gamma_{m-1}} \cdots \int_{\gamma_1} \frac{f(\zeta_1, \ldots, \zeta_{m-1}, \bar{z}_m)}{\prod_{j=1}^{m-1} (\zeta_j - z_j)} \, \mathrm{d}\zeta_1 \ldots \mathrm{d}\zeta_{m-1}.$$

Using the univariate Cauchy's formula again on the function $\bar{z}_m \mapsto f(\zeta_1, \ldots, \zeta_{m-1}, \bar{z}_m)$ under the integral sign gives

$$\operatorname{Ind}_{\gamma_m}(z) f(\zeta_1, \ldots, \zeta_{m-1}, \bar{z}_m) = \frac{1}{2\pi i} \int_{\gamma_m} \frac{f(\zeta_1, \ldots, \zeta_{m-1}, \bar{\zeta}_m)}{\bar{\zeta}_m - \bar{z}_m} \, \mathrm{d}\bar{\zeta}_m.$$

Now, we can substitute and apply Fubini's theorem to change the order of integration. The application is possible because f is continuous. This completes the induction step. □

Appendix C: Problem data In Tables 3 and 4, we give the coefficients of the hard knapsack instances discussed in 4.

Table 3. Coefficients of the hard knapsack instances from Pisinger [31].

problem	coefficients a
p1	[485, 326, 248, 421, 322, 795, 43, 845, 955, 252, 9, 901, 122, 94, 738, 574, 715, 882, 367, 984, 299, 433, 682, 72, 874, 138, 856, 145, 995, 529, 199, 277, 97, 719, 242, 107, 122, 70, 98, 600, 645, 267, 972, 895, 213, 748, 487, 923, 29, 674]
p2	[204, 448, 654, 863, 928, 294, 801, 608, 769, 70, 8, 183, 696, 616, 561, 801, 481, 396, 997, 458, 737, 844, 656, 331, 465, 132, 70, 636, 810, 171, 398, 800, 240, 617, 744, 109, 448, 553, 160, 59, 451, 902, 730, 634, 731, 47, 358, 526, 659, 668]
p3	[274, 571, 61, 658, 535, 792, 560, 370, 584, 889, 6, 466, 919, 139, 383, 27, 896, 911, 628, 933, 176, 254, 279, 589, 407, 127, 933, 126, 624, 462, 596, 970, 382, 516, 895, 760, 423, 35, 223, 517, 258, 888, 489, 20, 898, 345, 876, 776, 938, 663]
p4	[345, 693, 467, 100, 141, 291, 670, 781, 46, 59, 357, 100, 493, 661, 206, 254, 310, 425, 258, 407, 614, 313, 901, 496, 350, 121, 795, 617, 439, 104, 795, 141, 525, 414, 45, 762, 749, 518, 637, 976, 416, 523, 247, 759, 416, 644, 395, 379, 216, 657]
p5	[415, 464, 874, 895, 748, 437, 429, 543, 861, 878, 355, 383, 716, 184, 28, 128, 725, 940, 889, 530, 53, 723, 876, 754, 940, 116, 658, 755, 253, 395, 993, 663, 667, 313, 548, 413, 724, 648, 700, 434, 223, 157, 6, 145, 935, 294, 913, 629, 495, 652]
p6	[486, 586, 280, 337, 354, 936, 539, 954, 675, 696, 354, 17, 938, 58, 851, 355, 139, 454, 519, 4, 139, 134, 498, 13, 883, 110, 872, 246, 68, 685, 192, 834, 810, 211, 698, 415, 50, 131, 762, 893, 29, 144, 764, 532, 101, 593, 432, 880, 773, 646]
p7	[556, 709, 687, 132, 961, 434, 298, 716, 490, 515, 352, 300, 513, 581, 25, 581, 906, 969, 150, 479, 578, 544, 121, 919, 825, 105, 735, 736, 882, 328, 742, 356, 952, 110, 201, 66, 25, 613, 825, 351, 188, 778, 523, 270, 620, 891, 302, 482, 404, 641]
p8	[627, 831, 93, 574, 215, 933, 56, 127, 952, 333, 351, 934, 735, 103, 848, 808, 320, 483, 780, 953, 16, 955, 95, 178, 416, 99, 597, 227, 697, 618, 941, 527, 95, 656, 351, 68, 351, 96, 239, 810, 994, 765, 281, 657, 786, 190, 821, 733, 682, 635]
p9	[697, 602, 500, 369, 822, 431, 167, 889, 767, 152, 349, 217, 958, 626, 670, 682, 735, 998, 411, 428, 455, 365, 718, 436, 358, 94, 460, 717, 511, 909, 139, 49, 237, 555, 502, 719, 326, 226, 302, 268, 801, 399, 40, 43, 305, 488, 339, 335, 961, 630]
p10	[416, 724, 906, 811, 428, 930, 925, 300, 581, 970, 700, 851, 532, 148, 493, 909, 149, 512, 41, 902, 893, 776, 340, 343, 949, 736, 322, 208, 326, 551, 338, 220, 380, 453, 4, 721, 652, 709, 364, 727, 607, 386, 798, 782, 471, 139, 858, 586, 239, 624]
p11	[486, 847, 313, 606, 35, 428, 36, 62, 44, 789, 698, 134, 755, 671, 315, 135, 564, 27, 24, 377, 332, 186, 315, 601, 891, 731, 537, 698, 140, 842, 536, 390, 522, 352, 155, 372, 627, 191, 779, 185, 766, 20, 557, 168, 990, 437, 376, 836, 518, 619]
p12	[557, 617, 719, 48, 641, 927, 794, 825, 858, 607, 697, 768, 977, 193, 138, 362, 978, 541, 654, 851, 770, 597, 937, 860, 834, 725, 399, 189, 955, 132, 735, 913, 665, 250, 657, 374, 953, 674, 841, 644, 572, 655, 315, 555, 156, 736, 247, 439, 796, 613]
p13	[627, 740, 126, 843, 248, 425, 553, 235, 673, 426, 695, 51, 552, 716, 960, 236, 745, 56, 285, 326, 209, 655, 560, 766, 424, 720, 262, 327, 769, 775, 933, 83, 807, 149, 808, 25, 928, 156, 904, 102, 379, 641, 426, 293, 675, 34, 765, 689, 427, 608]
p14	[698, 862, 532, 285, 854, 924, 663, 998, 487, 244, 694, 685, 774, 238, 783, 463, 159, 570, 915, 800, 647, 66, 534, 25, 367, 714, 124, 818, 584, 65, 132, 606, 950, 47, 958, 27, 254, 287, 966, 561, 537, 276, 184, 680, 841, 685, 284, 292, 705, 602]
p15	[768, 985, 939, 80, 461, 422, 422, 408, 950, 415, 692, 968, 349, 761, 957, 689, 574, 85, 546, 275, 734, 476, 157, 283, 957, 709, 339, 308, 750, 708, 330, 776, 444, 946, 461, 678, 229, 769, 381, 19, 344,262,943,418,360,983,802,542,984,597]

Table 4. Coefficients of the random instances.

problem	coefficients a	problem	coefficients a
s1	[15, 5, 12, 6, 16]	b1	[17, 2, 16, 19, 6]
s2	[4, 2, 18, 14, 11]	b2	[9, 13, 19, 3, 4]
s3	[7, 19, 9, 13, 5]	b3	[10, 8, 14, 5, 2]
s4	[12, 11, 9, 6, 10]	b4	[3, 13, 11, 15, 14]
s5	[4, 14, 17, 11, 10]	b5	[5, 14, 7, 18, 2]
s6	[17, 5, 14, 7, 8]	b6	[11, 13, 18, 16, 2]
s7	[13, 5, 11, 16, 14]	b7	[16, 15, 7, 9, 4]
s8	[8, 12, 7, 18, 9]	b8	[13, 2, 16, 18, 6]
s9	[8, 10, 2, 16, 5]	b9	[3, 19, 10, 12, 16]
s10	[15, 4, 5, 14, 3]	b10	[14, 16, 10, 7, 3]
s11	[17, 5, 8, 10, 16]	b11	[8, 11, 2, 14, 9]
s12	[5, 4, 2, 15, 12]	b12	[2, 7, 16, 14, 6]
s13	[13, 14, 12, 7, 18]	b13	[4, 6, 14, 2, 9]
s14	[5, 4, 19, 12, 11]	b14	[7, 19, 16, 11, 5]
s15	[2, 9, 10, 8, 3]	b15	[2, 11, 4, 8, 10]
s16	[5, 6, 8, 16, 17]	b16	[7, 10, 15, 12, 9]
s17	[7, 15, 8, 5, 4]	b17	[18, 2, 13, 14, 8]
s18	[9, 7, 5, 16, 4]]	b18	[6, 5, 18, 16, 10]
s19	[9, 4, 17, 8, 6]	b19	[11, 10, 18, 4, 15]
s20	[10, 18, 13, 17, 14]	b20	[10, 9, 7, 11, 18]

Acknowledgments. This research was 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).

References

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

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

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

[4] Barvinok A (2008) *Integer Points in Polyhedra.* Zurich Lectures in Advanced Mathematics 11 (Zürich: European Mathematical Society).

[5] 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).

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

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

[8] 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.

[9] Bertozzi A, McKenna J (1993) Multidimensional residues, generating functions, and their application to queueing networks. *SIAM Review* 35(2):239–268.

[10] Bornemann F (2011) Accuracy and stability of computing high-order derivatives of analytic functions by Cauchy integrals. *Foundations of Computational Mathematics* 11(1):1–63.

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

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

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

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

[15] 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).

[16] 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.

[17] Dell'Amico M, Martello S (1999) Reduction of the three-partition problem. *Journal of Combinatorial Optimization* 3:17–30.

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

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

[20] Garey MR, Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (New York: W. H. Freeman and Company).

[21] 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.

[22] Hörmander L (1991) *An Introduction to Complex Analysis in Several Variables* (Amsterdam: Elsevier), 3 edition.

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

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

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

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

[27] 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).

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

[29] 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).

[30] Loera JAD, Hemmecke R, Köppe M, Weismantel R (2006) An FPTAS for mixed-integer polynomial optimization with a fixed number of variables. *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, Miami, FL, January 22-26, 2006*, 743–748.

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

[32] Rudin W (1969) *Function Theory in Polydiscs* (New York: W. A. Benjamin).

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

[34] Valiant LG (1979) The complexity of enumeration and reliability problems. *SIAM Journal on Computing* 8(3):410–421.