INDIAN INSTITUTE OF MANAGEMENT AHMEDABAD

IIMA
Working Paper

A General Purpose Exact Solution Method for Mixed Integer Concave Minimization Problems

Ankur Sinha
Arka Das
Guneshwar Anand
Sachin Jayaswal

Research & Publications

Research & Publications
W. P. No. 2021-03-01

A General Purpose Exact Solution Method for Mixed Integer Concave Minimization Problems

Ankur Sinha
Arka Das
Guneshwar Anand
Sachin Jayaswal

March 2021

INDIAN INSTITUTE OF MANAGEMENT AHMEDABAD

A General Purpose Exact Solution Method for Mixed Integer Concave Minimization Problems

Ankur Sinha* Arka Das† Guneshwar Anand‡ Sachin Jayaswal§

August 12, 2021

Abstract

In this article, we discuss an exact algorithm for mixed integer concave minimization problems. A piecewise inner-approximation of the concave function is achieved using an auxiliary linear program that leads to a bilevel program, which provides a lower bound to the original problem. The bilevel program is reduced to a single-level formulation with the help of Karush-Kuhn-Tucker (KKT) conditions. Incorporating the KKT conditions lead to complementary slackness conditions that are linearized using BigM. Multiple bilevel programs, when solved over iterations, guarantee convergence to the exact optimum of the original problem. Though the algorithm is general and can be applied to any optimization problem with concave function(s), in this paper, we solve two common classes of operations and supply chain problems; namely, the concave knapsack problem, and the concave production-transportation problem. The computational experiments indicate that our proposed approach outperforms the customized methods that have been used in the literature to solve the two classes of problems by an order of magnitude in most of the test cases.

Keywords— Concave Minimization, Mixed Integer Non-Linear Programming, Non-Convex Optimization

1 Introduction

The interest in non-convex optimization is motivated by its applications to a wide variety of real-world problems, including concave knapsack problems (Sun et al., 2005; Han et al., 2017), production-transportation problem (Kuno and Utsunomiya, 2000), facility location problems with concave costs (Soland, 1974), concave minimum cost network flow problems (Guisewite and Pardalos, 1990; Fontes and Gonçalves, 2007), etc. Non-convexities often arise, in the above problems, due to the presence of concave functions either in the objective or in the constraints. It is difficult to solve these optimization problems exactly, and hence the problem has been of interest to the optimization community since the 1960s.

One of the earliest studies on non-convex optimization problems is by Tuy (1964), where the author proposed a cutting plane algorithm for solving concave minimization problems over a polyhedron. The proposed algorithm was based on the partitioning of the feasible region, where the partitions were successively eliminated using Tuy cuts. However, the algorithm had a drawback that it was not guaranteed to be finite, which was tackled later by Zwart (1974); Majthay and Whinston (1974). Apart from cutting plane approaches, researchers also used relaxation-based ideas. For instance, Falk and Hoffman (1976); Carrillo (1977) computed successive underestimations of the concave function to solve

*Production and Quantitative Methods, Indian Institute of Management Ahmedabad, Gujarat, India, 380015, asinha@iima.ac.in

†Production and Quantitative Methods, Indian Institute of Management Ahmedabad, Gujarat, India, 380015, phd17arkadas@iima.ac.in

‡Production and Operations Management, Indian Institute of Management Visakhapatnam, Andhra Pradesh, India, 530003, guneshwar.anand-phd19@iimv.ac.in

§Production and Quantitative Methods, Indian Institute of Management Ahmedabad, Gujarat, India, 380015,sachin@iima.ac.in

the problem optimally. Branch-and-bound based approaches are also common to solve these problems, where the feasible region is partitioned into smaller parts using branching (Falk and Soland, 1969; Horst, 1976; Ryoo and Sahinidis, 1996; Tawarmalani and Sahinidis, 2004). Other ideas for handling concavities are based on extreme point ranking (Murty, 1968; Taha, 1973) or generalized Benders decomposition (Floudas et al., 1989; Li et al., 2011). The limitations of some of the above studies are one or more of the following: applicable only to a specific class of concave minimization problems; strong regularity assumptions; non-finite convergence; and/or convergence to a local optimum. Due to these limitations, there is also a plethora of specialized heuristics and meta-heuristics in the literature to obtain good quality or approximate solutions in *less* time. However, most of the heuristics and meta-heuristics do not guarantee convergence. Therefore, the idea of obtaining good quality solutions is often questioned as there is no way to ascertain how far the solution is from the global optimum.

In this paper, we discuss an algorithm for minimization problems with concave functions, which requires few assumptions about the problem structure. The problem studied in this paper is also studied in the area of difference-of-convex (DC) programming, for instance, the work by Strekalovsky (2015) comes close to our study. However, there have been challenges in directly implementing many of the DC programming approaches on operations and supply chain problems, as the problems considered are often large dimensional mixed integer problems for which obtaining the exact solution in reasonable time is difficult. We design and implement a piecewise-linear inner approximation method that is able to solve large dimensional operations and supply chain problems that involve mixed integers and concavities. The method relies on the piecewise-linear inner approximation (Rockafellar, 1970) approach, which replaces the concave function to arrive at a bilevel formulation that leads to the lower bound of the original problem. The bilevel optimization problem is solvable using the Karush-Kuhn-Tucker (KKT) approach. Through an iterative procedure, wherein multiple bilevel programs are solved, the method converges to the global optimum of the original problem. The method can be used to solve concave minimization problems with continuous or discrete variables exactly, as long as the concavities in the optimization problem are known. We solve two classes of optimization problems in this paper to demonstrate the efficacy of our method: (i) concave knapsack problem; and (ii) concave production-transportation problem. For the concave production-transportation problem, we further consider two sub-classes: (a) single sourcing; and (b) multiple sourcing that have quite different formulations. We show that the proposed exact method, which is general, beats the existing specialized methods for solving the application problems by a large margin.

The rest of the paper is organized as follows. We provide the algorithm description, followed by the convergence theorems and proofs in Section 2. The concave knapsack problems and production-transportation problems are discussed in Section 3 and Section 4, respectively. Each of these sections contains a brief survey, problem description, and computational results for its respective problem. Finally, we conclude in Section 5. The paper also has an Appendix, where we show the working of the algorithm on two sample problems.

2 Algorithm Description

We consider optimization problems of the following kind

$$\min_x \; f(x) + \phi(x) \tag{1}$$

$$\text{subject to } g_i(x) \le 0, \qquad i = 1, \dots, I \tag{2}$$

$$x_k^l \le x_k \le x_k^u, \qquad k = 1, \dots, n \tag{3}$$

where $f(x)$ and $g(x)$ are convex, $\phi(x)$ is strictly concave and $x \in \mathbb{R}^n$. Note that there is no restriction on x, which may be combinatorial, integer or continuous. The functions are assumed to be Lipschitz continuous. For a given set of points $S_c = \{z^1, z^2, \dots, z^\tau\}$ (let $c = 1$), the function $\phi(x)$ can be approximated as follows (see Section 12 in Rockafellar (1970)):

$$\hat{\phi}(x|S_c) = \max_\mu \left\{ \sum_{j=1}^{\tau} \mu_j \phi(z^j) : \sum_{j=1}^{\tau} \mu_j = 1, \sum_{j=1}^{\tau} \mu_j z_k^j = x_k, k = 1, \dots, n, \mu_j \ge 0, j = 1, \dots, \tau \right\} \tag{4}$$

which is a linear program with x as a parameter and μ as a decision vector. For brevity, we will represent the approximation $\hat{\phi}(x|S_c)$ as $\hat{\phi}(x)$. Figures 1 and 2 represent the approximation graphically and show how the approximation improves with addition of a new point in the approximation set. Note that the

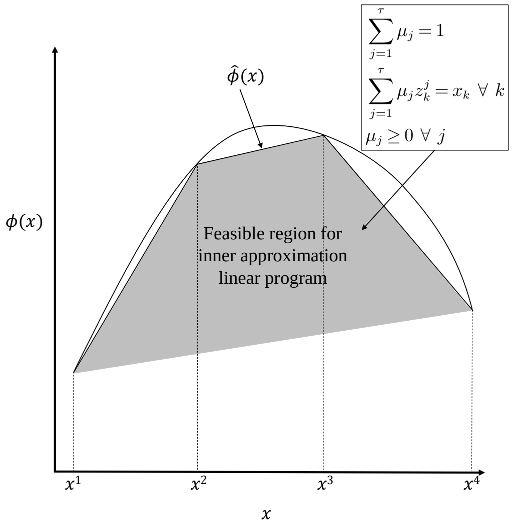

Figure 1: Inner-approximation of $\phi(x)$ with a set of points ($\tau = 4, n = 1$).

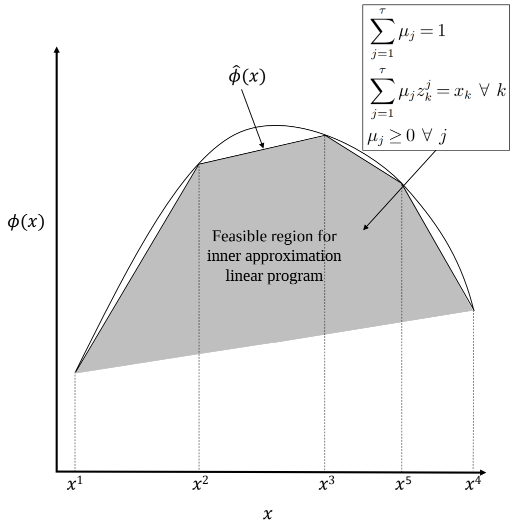

Figure 2: Inner-approximation of $\phi(x)$ with an additional point ($\tau = 5, n = 1$).

feasible region of (4) in Figure 2 is a superset of the feasible region in Figure 1. We will use this property later while discussing the convergence properties of the algorithm. The above approximation converts the concave minimization problem (1)-(3) into the following lower bound program, as $\hat{\phi}(x)$ is the inner piecewise linear approximation of $\phi(x)$.

$$\min_{x} \ f(x) + \hat{\phi}(x) \tag{5}$$

$$\text{subject to } g_i(x) \leq 0, \qquad i = 1, \ldots, I \tag{6}$$

$$x_k^l \leq x_k \leq x_k^u, \qquad k = 1, \ldots, n \tag{7}$$

Theorem 1. *The formulation* (5)-(7) *provides a lower bound for the formulation* (1)-(3).

Proof. Given that $\hat{\phi}(x)$ is a piecewise inner-approximation of $\phi(x)$, the function $\hat{\phi}(x)$ always bounds $\phi(x)$ from below. Therefore, at any given x, $\hat{\phi}(x)$ will always take a smaller value than $\phi(x)$. This implies the following:

$$f(x) + \hat{\phi}(x) \leq f(x) + \phi(x)$$

$\square$

Formulation (5)-(7) is a bilevel program, which can be written as follows:

$$\begin{aligned}
&\min_{x,\zeta} \ f(x) + \zeta \\
&\text{subject to } \mu \in \operatorname*{argmax}_{\mu} \left\{ \sum_{j=1}^{\tau} \mu_j \phi(z^j) : \sum_{j=1}^{\tau} \mu_j = 1, \sum_{j=1}^{\tau} \mu_j z_k^j = x_k, k = 1, \ldots, n, \mu_j \geq 0, j = 1, \ldots, \tau \right\} \\
&\qquad \sum_{j=1}^{\tau} \mu_j \phi(z^j) \leq \zeta \\
&\qquad g_i(x) \leq 0, \quad i = 1, \ldots, I \\
&\qquad x_k^l \leq x_k \leq x_k^u, \quad k = 1, \ldots, n
\end{aligned}$$

A bilevel problem, where the lower level is a linear program, is often solved by replacing the lower level with its KKT conditions. Substituting the KKT conditions for the lower level program using α as the Lagrange multiplier for $\sum_{j=1}^{\tau} \mu_j = 1$, β as the Lagrange multiplier for $\sum_{j=1}^{\tau} \mu_j z_k^j = x_k$ and γ as the

Lagrange multiplier for $\mu_j \geq 0$, the above formulation reduces to the following.

$$\textbf{Mod-}S_c \qquad \min_{x,\alpha,\beta,\gamma,\zeta} f(x) + \zeta \tag{8}$$

$$\text{subject to} \tag{9}$$

$$g_i(x) \leq 0 \qquad i = 1, \dots, I \tag{10}$$

$$\sum_{j=1}^{\tau} \mu_j \phi(z^j) \leq \zeta \tag{11}$$

$$\sum_{j=1}^{\tau} \mu_j = 1 \tag{12}$$

$$\sum_{j=1}^{\tau} \mu_j z_k^j = x_k \qquad k = 1, \dots, n \tag{13}$$

$$\phi(z^j) - \alpha - \sum_{k=1}^{n} \beta_j z_j^k + \gamma_j = 0 \qquad j = 1, \dots, \tau \tag{14}$$

$$\mu_j \gamma_j = 0 \qquad j = 1, \dots \tau \tag{15}$$

$$\mu_j \geq 0 \qquad j = 1, \dots \tau \tag{16}$$

$$\gamma_j \geq 0 \qquad j = 1, \dots \tau \tag{17}$$

$$x_k^l \leq x_k \leq x_k^u \qquad k = 1, \dots, n \tag{18}$$

Note that the above program contains product terms in the complementary slackness conditions ($\mu_j \gamma_j = 0$), which can be linearized using binary variables (u) and BigM (M_1 and M_2) as follows:

$$\gamma_j \leq M_1 u_j, \quad j = 1, \dots \tau \tag{19}$$

$$\mu_j \leq M_2(1 - u_j), \quad j = 1, \dots, \tau \tag{20}$$

From constraints (12) and (16), we observe that the maximum value that μ_j can take is 1. Hence, $M_2 = 1$ is acceptable. We may choose M_1 to be any big number.

After linearization of the complimentary slackness conditions, (8)-(14), (16)-(20) is a convex mixed integer program (MIP), which represents a lower bound for (1)-(3). Solving (8)-(14), (16)-(20) leads to $z^{\tau+1}$ as an optimal point, which is a feasible point for the original problem (1)-(3). Therefore, substituting $z^{\tau+1}$ in (1) provides an upper bound for the original problem. The optimal point $z^{\tau+1}$ to the convex MIP is used to create a new set $S_{c+1} = S_c \cup z^{\tau+1}$ corresponding to which a new convex MIP is formulated. The new convex MIP formulated with an additional point is expected to provide improved lower and upper bounds in the next iteration of the algorithm. This algorithm is referred to as the Inner-Approximation (IA) algorithm in the rest of the paper. A pseudo-code of IA algorithm is provided in Algorithm 1. The

Algorithm 1 IA Algorithm for solving concave problem

Begin
$UB_{\mathcal{A}} \leftarrow +\infty, LB_{\mathcal{A}} \leftarrow -\infty, c \leftarrow 1$
Choose an initial set of τ points $S_c = \{z^1, z^2, \dots, z^\tau\}$
while $((UB_{\mathcal{A}} - LB_{\mathcal{A}})/LB_{\mathcal{A}} > \epsilon)$ begin **do**
  Solve Mod-S_c ((8)-(18)) with an MIP solver after linearizing (15)
  Let the optimal solution for Mod-S_c be $z^{\tau+c}$
  $LB_{\mathcal{A}} \leftarrow f(z^{\tau+c}) + \hat{\phi}(z^{\tau+c})$
  $UB_{\mathcal{A}} \leftarrow f(z^{\tau+c}) + \phi(z^{\tau+c})$
  $\mathcal{S}_{c+1} \leftarrow \mathcal{S}_c \cup z^{\tau+c}$
  $c \leftarrow c + 1;$
End

algorithm starts with an initial set of points $S_1 = \{z^1, \dots, z^\tau\}$, such that dom $\phi(x) \subseteq$ conv S_1.

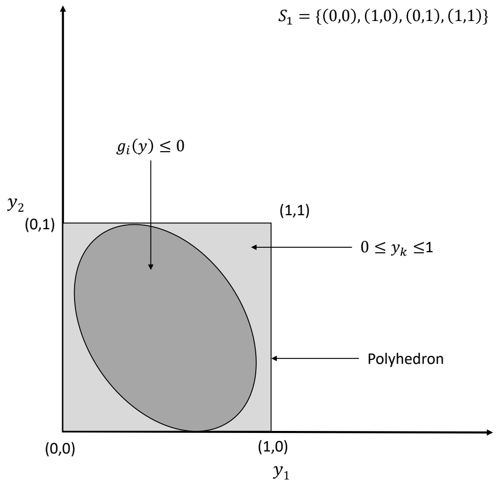

Figure 3: Smaller polyhedron with larger number of points.

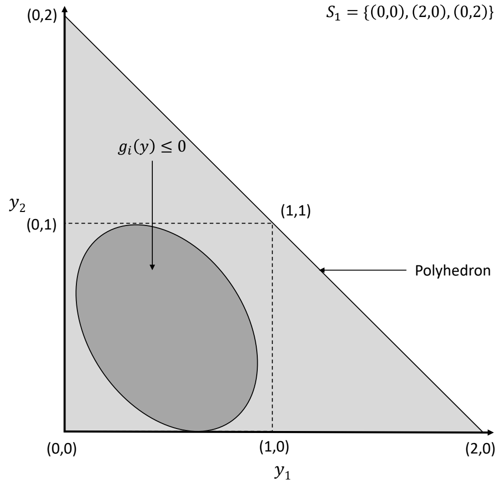

Figure 4: Larger polyhedron with smaller number of points.

2.1 The Initial Set

In this section, we discuss the choice of the initial set $S_1 = \{z^1, \dots, z^\tau\}$, such that dom $\phi(x) \subseteq$ conv S_1. The bound constraints in (1)-(3) are important so that the initial set S_1 may be chosen easily. One of the ways to initialize S_1 would be to choose the corner points of the box constraints $x_k^l \le x_k \le x_k^u$, $k = 1, \dots, n$. Additional points may be sampled randomly between the lower and upper bounds at the start of the algorithm for a better initial approximation of $\phi(x)$, but are not necessary. However, note that for a problem with n variables, choosing the corner points of the box constraints, amounts to starting the algorithm with the cardinality of S_1 as 2^n. For large dimensional problems, the size of the set may be very large, and therefore the approach would be intractable. For large dimensional problem we propose an alternative technique to choose S_1, such that dom $\phi(x) \subseteq$ conv S_1, but the number of points in S_1 is only $n + 1$.

Without loss of generality, assume that the lower bound is 0 and the upper bound is 1, as one can always normalize the variables by replacing variables x_k with $y_k(x_k^u - x_k^l) + x_k^l$ such that $0 \le y_k \le 1$. In such a case, Figure 3 shows the feasible region $g_i(y) \le 0 \; \forall \, i$, enclosed in the polyhedron $0 \le y_k \le 1 \; \forall \, k$. Another polyhedron that encloses $g_i(y) \le 0 \; \forall \, i$ completely is shown in Figure 4. While the polyhedron in Figure 3 is smaller in terms of the area (or volume), the polyhedron in Figure 4 is comparatively larger. However, the number of points required to form the polyhedron in Figure 3 for an n dimensional problem would be 2^n, whereas the polyhedron in Figure 4 will require only $n + 1$ points for an n dimensional problem. For the second case, in an n dimensional problem the points can be chosen as follows, $(0, 0, \dots, 0), (n, 0, \dots, 0), (0, n, \dots, 0), \dots, (0, 0, \dots, n)$. These points from the y space can be transformed to the corresponding x space by the following substitution $x_k = y_k(x_k^u - x_k^l) + x_k^l$. One may of course choose any other polyhedron that completely encloses $g_i(x) \le 0 \; \forall \, i$.

2.2 Convergence Results

Next, we discuss the convergence results for the proposed algorithm. First we prove that if the algorithm provides the same solution in two consecutive iterations, then the solution is an optimal solution to the original concave minimization problem (1)-(3).

Theorem 2. *If two consecutive iterations i and $i + 1$ lead to the same solution then the solution is optimal for* (1)-(3).

Proof. Say that $z^{\tau+i}$ is the solution at iteration i. Note that $S_{i+1} = S_i \cup z^{\tau+i}$, which implies that at iteration $i+1$, $\hat{\phi}(z^{\tau+i}|S_{i+1}) = \phi(z^{\tau+i})$. From Theorem 1, at iteration $i+1$, $f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1}) \le f(z^{\tau+i+1}) + \phi(z^{\tau+i+1})$. Since $z^{\tau+i+1} = z^{\tau+i}$ and $\hat{\phi}(z^{\tau+i}|S_{i+1}) = \phi(z^{\tau+i})$, $f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1}) \le f(z^{\tau+i+1}) + \phi(z^{\tau+i+1})$ holds with an equality, it implies that $z^{\tau+i}$ is the optimal solution. $\square$

Theorem 3. *When the algorithm proceeds from iteration i to $i+1$ then the lower bound for (1)-(3) improves, if $z^{\tau+i}$ at iteration i is not the optimum for (1)-(3).*

Proof. It is given that $z^{\tau+i}$ is the solution for (5)-(7) at iteration i, which is not optimal for the original problem ((1)-(3)). Say that $z^{\tau+i+1}$ is the solution for (5)-(7) at iteration $i+1$, so from Theorem 2 we can say that $z^{\tau+i} \neq z^{\tau+i+1}$.

Note that for any given x, $\hat{\phi}(x|S_i) \leq \hat{\phi}(x|S_{i+1})$, as the linear program corresponding to $\hat{\phi}(x|S_{i+1})$ is a relaxation of $\hat{\phi}(x|S_i)$. This is shown in the next statement. If $\mu_{\tau+1} = 0$ is added in the linear program corresponding to $\hat{\phi}(x|S_{i+1})$, it becomes equivalent to the linear program corresponding to $\hat{\phi}(x|S_i)$, which shows that $\hat{\phi}(x|S_{i+1})$ is a relaxation of $\hat{\phi}(x|S_i)$.

Since $\hat{\phi}(x|S_i) \leq \hat{\phi}(x|S_{i+1})$ for all x, we can say that $f(x)+\hat{\phi}(x|S_i) \leq f(x)+\phi(x|S_{i+1})$ for all x. This implies that for (5)-(7) comparing the objective function at the optimum, we get $f(z^{\tau+i}) + \hat{\phi}(z^{\tau+i}) \leq f(z^{\tau+i+1}) + \phi(z^{\tau+i+1})$. Strict concavity of ϕ and $z^{\tau+i} \neq z^{\tau+i+1}$ ensure that the equality will not hold, implying $f(z^{\tau+i}) + \hat{\phi}(z^{\tau+i}) < f(z^{\tau+i+1}) + \phi(z^{\tau+i+1})$. Therefore, the lower bound strictly improves in the next iteration. □

Theorem 4. *If $\phi(x)$ is Lipschitz continuous with Lipschitz constant K, then $\hat{\phi}(x|S_i) : x \in conv\ S_i$ is also Lipschitz continuous with the maximum possible value of Lipschitz constant as K.*

Proof. From the Lipschitz condition $|\phi(x_1)-\phi(x_2)| \leq K||x_1-x_2||$ and the concavity of $\phi(x) : x \in \text{conv } S_i$, we can say that $||\omega|| \leq K\ \forall\ \omega \in \partial\phi(x)$, where $\partial\phi(x)$ represents the subgradient. The function $\hat{\phi}(x|S_i) : x \in \text{conv } S_i$ is a concave polyhedral function, i.e. consisting of piecewise hyperplanes. Therefore, consider bounded polyhedra $X_j, j = 1, \ldots, s$ on which the hyperplanes are defined, such that $\nabla\hat{\phi}(x)$ is constant in the interior of X_j. Note that $\hat{\phi}(x|S_i) = \phi(x)$ on the vertices, otherwise $\hat{\phi}(x|S_i) \leq \phi(x)$. From the property of concavity, it is clear that $\nabla\hat{\phi}(x) \in \partial\phi(x) : x \in X_j$. This implies that $||\nabla\hat{\phi}(x)|| \leq K\ \forall\ x \in X_j$, which can be generalized for all hyperplanes. □

Theorem 5. *If $z^{\tau+i}$ is the solution for (5)-(7) at iteration i, $z^{\tau+i+1}$ is the solution for (5)-(7) at iteration $i+1$, and $||z^{\tau+i+1} - z^{\tau+i}|| \leq \delta$, then the optimal function value, v^* for (1)-(3) has the following property: $0 \leq f(z^{\tau+i+1})+\hat{\phi}(z^{\tau+i+1}|S_{i+1})-v^* \leq (K_1+K_2)\delta$, where K_1 and K_2 are the Lipchitz constants for $f(x)$ and $\phi(x)$, respectively.*

Proof. Note that at iteration $i+1$, $\hat{\phi}(z^{\tau+i}|S_{i+1}) = \phi(z^{\tau+i})$ with $z^{\tau+i}$ being a feasible solution for (1)-(3). Therefore, $f(z^{\tau+i}) + \hat{\phi}(z^{\tau+i}|S_{i+1})$ is the upper bound for v^*. Also $f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1})$ is the lower bound for v^* from Theorem 1.

$$f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1}) \leq v^* \leq f(z^{\tau+i}) + \hat{\phi}(z^{\tau+i}|S_{i+1})$$

If K_2 is the Lipschitz constant for $\phi(x)$, then it is also the Lipschitz constant for $\hat{\phi}(x)$ from Theorem 4. From the Lipschitz property,

$$f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1}) \leq v^* \leq f(z^{\tau+i}) + \hat{\phi}(z^{\tau+i}|S_{i+1}) \leq f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1}) + (K_1 + K_2)\delta$$

which implies $0 \leq f(z^{\tau+i+1}) + \hat{\phi}(z^{\tau+i+1}|S_{i+1}) - v^* \leq (K_1 + K_2)\delta$. □

To illustrate the working of the algorithm, an example has been provided in the Appendix (see Section A). Next, we apply the algorithm on two common classes of concave minimization problems.

3 Concave Knapsack Problem

The integer/binary knapsack problem requires determining the items to be chosen from a given collection of items with certain weights and values so as to maximize the total value without exceeding a given total weight limit. Over the last sixty years, integer/binary knapsack problems have received considerable attention mostly due to their wide variety of applications in financial decision problems, knapsack cryptosystems, combinatorial auctions, etc. (Kellerer et al., 2004). The integer/binary Knapsack problem is known to be NP-complete, for which a variety of algorithms have been reported in the literature, including Lagrangian relaxation (Fayard and Plateau, 1982; Fisher, 2004), branch-and-bound (B&B) (Kolesar, 1967), dynamic programming (Martello et al., 1999), and hybrid methods combining B&B and dynamic programming (Marsten and Morin, 1978), etc. The literature has also seen a proliferation of papers on non-linear Knapsack problems (NKP), which arise from economies and dis-economies of scale in modelling various problems such as capacity planning (Bitran and Tirupati, 1989), production planning (Ziegler, 1982; Ventura and Klein, 1988; Maloney and Klein, 1993), stratified sampling problems (Bretthauer et al., 1999), financial models (Mathur et al., 1983), etc. NKP also arises as a subproblem in solving service system design problems and facility location problems with stochastic demand (Elhedhli, 2005). NKP problem may be a convex or a non-convex problem in nature. Each of these types can be further classified as continuous or integer knapsack problems, separable or non-separable knapsack problems. In this paper, we aim to solve the concave separable integer knapsack problem (CSINK), where concavity in the objective function arises due to the concave cost structure. There are a plethora of applications that involve concave costs, such as capacity planning and fixed charge problems with integer variables (Bretthauer and Shetty, 1995; Horst and Thoai, 1998; Horst and Tuy, 2013), and other problems with economies of scale (Pardalos and Rosen, 1987). Specifically, applications of CSINK include communication satellite selection (Witzgall, 1975), pluviometer selection in hydrological studies (Gallo et al., 1980a; Caprara et al., 1999), compiler design (Johnson et al., 1993; Pisinger, 2007), weighted maximum b-clique problems (Park et al., 1996; Dijkhuizen and Faigle, 1993; Pisinger, 2007; Caprara et al., 1999). Due to its wide applications, CSINK has attracted a lot of researchers to solve it efficiently.

Gallo et al. (1980a) reported one of the first approaches for the quadratic knapsack problem by utilizing the concept of the upper plane, which is generated by the outer linearization of the concave function. Researchers have also come up with different B&B-based algorithms to solve the concave minimization version of the problem with integer variables (Marsten and Morin, 1978; Victor Cabot and Selcuk Erenguc, 1986; Benson and Erenguc, 1990; Bretthauer et al., 1994; Caprara et al., 1999). Chaillou et al. (1989) proposed a Lagrangian relaxation-based bound of the quadratic knapsack problem. Moré and Vavasis (1990) proposed an algorithm that characterizes local minimizers when the objective function is strictly concave and used this characterization in determining the global minimizer of a concave knapsack problem with linear constraints. Michelon and Veilleux (1996) reported a Lagrangian based decomposition technique for solving the concave quadratic knapsack problem. Later, Sun et al. (2005) developed an iterative procedure of linearly underestimating the concave function and executing domain cut and partition by utilizing the special structure of the problem. Most recently, Wang (2019) reported an exact algorithm that combines the *contour cut* (Li et al., 2006) with a special cut to gradually reduce the duality gap through an iterative process to solve CSINK. Wang showed that his proposed algorithm outperformed the hybrid method proposed by Marsten and Morin (1978).

The model for CSINK is described below:

$$\min_{x} \quad \phi(x) = \sum_{j=1}^{n} \phi_j(x_j) \tag{21}$$

subject to

$$Ax \leq b \tag{22}$$

$$x \in X = \{x \in \mathbb{Z}^n \mid l_j \leq x_j \leq u_j\} \tag{23}$$

where $\phi_j(x_j),\ j = 1 \ldots n$ are concave non-decreasing functions, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $l = (l_1, \ldots l_n)^T, u = (u_1, \ldots u_n)^T$ are upper and lower bounds of x respectively. In the next section, we discuss about the dataset used for the computational experiments and present the results of the IA algorithm. We benchmark our method against Wang (2019).

3.1 Computational Experiments

In this section, we present the data generation technique, followed by a discussion on computational results. All computational experiments are carried out on a PC with Pentium(R) Dual-core CPU i5-6200U @2.3 GHz and 8 GB RAM. As described in Section 2, the IA algorithm is coded in C++, and the MIP in step 5 of Algorithm 1 is solved using the default Branch&Cut solver of CPLEX 12.7.1. The optimality gap ($\epsilon = 0.01$) is calculated as $\frac{UB-LB}{LB} \times 100$, where UB and LB denote the upper bound and lower bound for the original problem, respectively. The algorithm is set to terminate using $\epsilon = 0.01$ in step 4 of Algorithm 1 or using a CPU time limit of 2 hours, whichever reaches first. We compare the computational performance of our method against Wang (2019). The experiments by Wang (2019) are done on a PC with Pentium(R) Dual-core CPU E6700 @3.2GHz, which is approximately 2.79 times slower than our system (https://www.cpubenchmark.net/singleCompare.php). Hence, for a fair comparison, we scale the computational times of the IA algorithm by a factor of 2.79.

3.1.1 Data-Set

All our computational experiments are performed on random data-sets, generated using the following scheme as described by Wang (2019). In all the test data-sets: $A = \{a_{ij}\}_{n\times m} \in [-20, -10]$; $b_i = \sum_{j=1}^n a_{ij}l_j + r\left(\sum_{j=1}^n a_{ij}u_j - \sum_{j=1}^n a_{ij}l_j\right)$; where $r = 0.6$; and $l_j = 1, u_j = 5; n \in \{30, \ldots, 150\}, m \in \{10, 15\}$. Further, we employ two different forms of concavity in the objective function (21): (i) polynomial form; and (ii) non-polynomial form. The parameters settings for both categories are briefly described as follows.

(i) Polynomial concave function:

$$\phi(x) = \sum_{j=1}^{n}(c_j x_j^4 + d_j x_j^3 + e_j x_j^2 + h_j x_j)$$

We use the following three kinds of polynomial concave functions, as used by Wang (2019). For a fixed n and m, ten random test problems are generated from a uniform distribution using the following scheme.

- Quadratic: $c_j = 0, d_j = 0, e_j \in [-15, -1], h_j \in [-5, 5], j = 1, \ldots, n$.
- Cubic: $c_j = 0, d_j \in (-1, 0), e_j \in [-15, -1], h_j \in [-5, 5], j = 1, \ldots, n$.
- Quartic: $c_j \in (-1, 0), d_j \in (-5, 0), e_j \in [-15, -1], h_j \in [-5, 5], j = 1, \ldots, n$

(ii) Non-polynomial concave function:

$$\phi(x) = \sum_{j=1}^{n}(c_j ln(x_j) + d_j x_j)$$

Once again, we generate 10 random data instances for a fixed n and m using uniform distribution with the following parameters: $c_j \in (0, 1), d_j \in [-20, -10], j = 1, \ldots, n$.

3.1.2 Computational Results

Tables 1-4 provide a comparison of the computational performance of the IA algorithm against Wang (2019). Since Wang (2019) report only the average, minimum, and maximum CPU times over 10 randomly generated instances for each size of the problem, we also do the same for a meaningful comparison. It is important to highlight that, as discussed earlier in this section, for a fair comparison, the CPU times for the IA algorithm have been scaled by a factor of 2.79 before reporting in Tables 1-4. For each problem size, the better of the two average CPU times (one for the IA algorithm and the other for Wang (2019)) is highlighted in boldface. Tables 1-3 provide the results corresponding to the three different polynomial forms (quadratic, cubic and quartic). As evident from the tables, the IA algorithm consistently outperforms Wang (2019) over all the instances for the case of the quadratic objective function (21), and over most of the instances for the other forms of the objective function except for the few easy instances. Specifically, for the quadratic objective function, the IA algorithm takes 55.35 seconds on average, over all the instances, which is less than one-sixth of 375.65 required by Wang (2019). For the cubic and the quartic form of the objective function, the average times over all the instances taken by the IA algorithm are 233.13 and 132.35, respectively, while the same taken by Wang (2019) are 450.12 and 259.32. Table 4 provides the results corresponding to the non-polynomial (logarithmic) form of the objective function (21). Clearly, the IA algorithm consistently outperforms Wang (2019) over all the instances for the case of the logarithmic objective function, taking an average of only 1.92 seconds, which is around 88 times smaller than the average time of 169.18 seconds taken by Wang (2019).

To further see the difference in the performances of two methods, we present their performance profiles (Dolan and Moré, 2002) in Figures 5-8. For this, let $t_{p,s}$ represent the CPU time to solve instance $p \in P$ using method $s \in S$. Using this notation, the performance ratio ($r_{p,s}$), which is defined as the ratio of the CPU time taken by a given method to that taken by the best method for that instance, can be mathematically given as follows:

$$r_{p,s} = \frac{t_{p,s}}{\min\limits_{s\in S}\ t_{p,s}} \tag{24}$$

Table 1: Experimental results for quadratic concave knapsack problem

	CPU Time (seconds)					
	IA Algorithm*			Wang (2019)		
n×m	Avg	Min	Max	Avg	Min	Max
30×10	**6.94**	0.43	34.53	17.85	0.41	75.64
40×10	**8.57**	0.14	34.91	50.98	2.05	350.94
50×10	**5.75**	0.84	22.25	142.39	1.34	980.34
80×10	**79.96**	2.94	276.72	793.83	14.81	7212.39
150×10	**85.83**	1.43	538.69	1116.95	0.01	3232.39
20×15	**18.85**	0.84	131.93	31.77	2.88	237.75
30×15	**102.02**	0.79	395.04	140.91	1.14	587.64
40×15	**134.85**	3.65	408.48	710.48	2.45	3125.30
Avg	**55.35**	1.38	230.32	375.65	3.14	1975.30

*Original CPU times are scaled by 2.79 for a fair comparison.

Table 2: Experimental results for cubic concave knapsack problem

	CPU Time (seconds)					
	IA Algorithm*			Wang (2019)		
n×m	Avg	Min	Max	Avg	Min	Max
30×10	13.97	0.33	46.57	**10.91**	0.25	26.80
40×10	**14.28**	0.23	44.85	30.73	1.30	98.47
60×10	**40.54**	1.21	120.09	166.62	6.72	1002.75
80×10	**237.87**	2.12	1139.96	275.28	5.08	1672.88
90×10	**271.76**	0.39	1520.73	631.08	0.00	5457.95
20×15	**38.36**	1.91	152.15	153.48	1.41	1059.48
30×15	**132.61**	5.98	347.53	159.91	3.58	999.77
50×15	**1115.69**	84.94	4462.55	2172.94	36.08	12747.97
Avg	**233.13**	12.14	979.30	450.12	6.80	2883.26

*Original CPU times are scaled by 2.79 for a fair comparison.

Table 3: Experimental results for quartic concave knapsack problem

	CPU Time (seconds)					
	IA Algorithm*			Wang (2019)		
n×m	Avg	Min	Max	Avg	Min	Max
30×10	30.18	1.15	116.12	**13.09**	2.02	35.81
50×10	74.68	0.21	290.12	**44.92**	4.61	153.17
70×10	**54.12**	1.52	260.06	396.58	8.03	1994.47
100×10	**188.88**	4.50	1259.25	611.00	8.80	4963.13
20×15	**101.06**	9.41	353.96	117.90	3.03	402.61
30×15	**115.11**	1.14	564.25	188.39	3.22	1583.52
40×15	**361.70**	0.74	1405.47	443.32	6.09	1281.44
Avg	**132.25**	2.67	607.03	**259.32**	5.11	1487.73

*Original CPU times are scaled by 2.79 for a fair comparison.

Table 4: Experimental results for logarithmic concave knapsack problem($\phi(x) = \sum_{j=1}^{n}(c_j ln(x_j) + d_j x_j)$)

	CPU Time (seconds)					
	IA Algorithm*			Wang (2019)		
n×m	Avg	Min	Max	Avg	Min	Max
30×10	**0.41**	0.14	0.91	3.97	0.20	18.59
50×10	**0.56**	0.13	1.14	9.54	1.08	24.11
70×10	**0.61**	0.13	1.46	44.11	1.86	147.33
95×10	**0.79**	0.18	4.00	277.36	0.02	1484.20
30×15	**1.36**	0.15	5.97	12.70	0.94	39.55
50×15	**3.16**	0.41	12.60	289.21	9.11	1156.89
70×15	**6.59**	0.45	36.72	547.38	27.03	2100.30
Avg	**1.92**	0.23	8.97	169.18	5.75	710.14

*Original CPU times are scaled by 2.79 for a fair comparison.

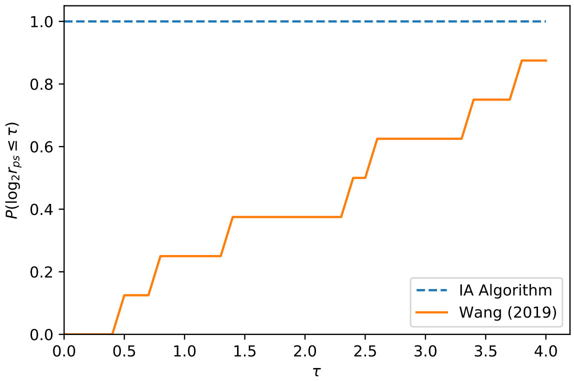

Figure 5: Performance profile of quadratic knapsack problem

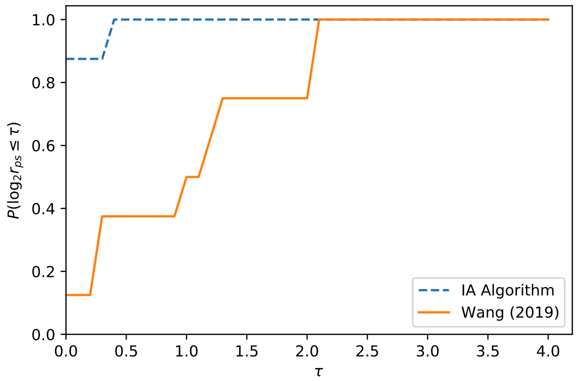

Figure 6: Performance profile of cubic knapsack problem

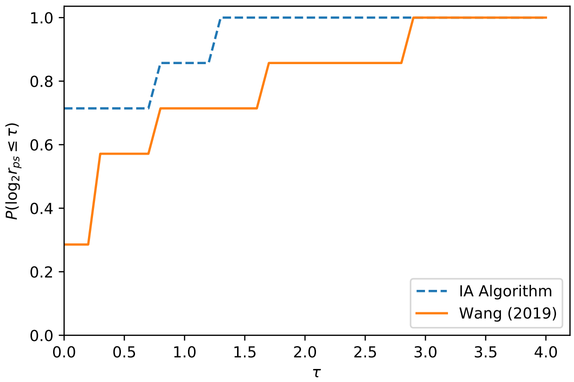

Figure 7: Performance profile of quartic knapsack problem

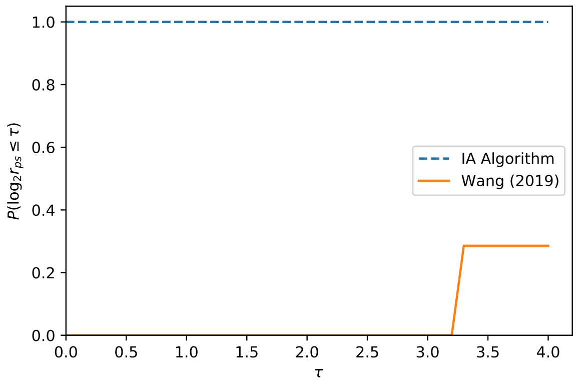

Figure 8: Performance profile of log knapsack problem

If we assume r_{ps} as a random variable, then the performance profile ($p_s(\tau)$) is the cumulative distribution function of r_{ps} at 2^τ, mathematically expressed as $p_s(\tau) = P(r_{p,s} \leq 2^\tau)$. In other words, it gives the probability that the CPU time taken by the method p does not exceed 2^τ times that taken by the better of the two methods. Further, for a given method p, the intercept of its performance profile on the y-axis shows the proportion of the instances for which it performs the best. The performance profiles for the polynomial functions are displayed in Figures 5-7, and the same for the non-polynomial function are displayed in Figure 8. In the absence of the CPU times for each individual instance by Wang (2019), we use the average computational times (after scaling by a factor of 2.79) for creating the performance profiles. From Figures 5-7, it can be concluded that the IA algorithm outperforms Wang (2019) for 100% of the instances for the quadratic objective function, while it is better for 87.5% and 71.4% of the instances for the cubic and quartic objective functions, respectively. For the non-polynomial (logarithmic) form of the objective function, Figure 8 shows the IA algorithm as outperforming Wang (2019) for 100% of the data instances. Furthermore, for the instances (for quadratic and non-polynomial objective functions) on which the performance of Wang (2019) is worse than the IA algorithm, it is unable to solve them to optimality even after $16 = (2^4)$ times the CPU time taken by the IA algorithm. Next, we discuss the formulation of the production-transportation problem and report the results of the IA algorithm benchmarking it against two approaches.

4 Production-Transportation Problem

Transportation problem is a classical optimization problem, which entails finding the minimum cost of transporting homogeneous products from a set of sources (e.g. factories) with their given supplies to meet the given demands at a set of destinations (e.g. warehouses). The production-transportation problem extends the classical transportation problem by introducing a production-related variable at each of the given sources, which decides the supply available at that source. The problem entails finding the production quantity at each source, besides the transportation quantities from supply sources to meet the demands at the destinations, at the minimum total production and transportation cost. To formally define a production-transportation problem, let $G = (V, U, E)$ be a bipartite graph, where V and U denote the sets of m sources and n destinations, respectively, and E denotes the set of $m \times n$ transportation arcs between the sources and the destinations. Let c_{ij} be the transportation cost per unit of the product on arc $(i, j) \in E$, and $\phi_i(y_i)$ be the cost of producing y_i units at source $i \in V$. Further, let d_j and k_i represent the demand at destination $j \in U$ and the production capacity at source $i \in V$, respectively. If we define x_{ij} as the amount of the product transported on the arc from i to j, and y_i as the production quantity at source i, then a production-transportation problem can be mathematically stated as:

$$\min_{x,y} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{i\in V} \phi_i(y_i) \tag{25}$$

subject to

$$\sum_{j\in U} x_{ij} \le y_i, \qquad \forall\, i \in V \tag{26}$$

$$y_i \le k_i, \qquad \forall\, i \in V \tag{27}$$

$$\sum_{i\in V} x_{ij} \ge d_j, \qquad \forall\, j \in U \tag{28}$$

$$x_{ij} \ge 0, \qquad \forall\, (i,j) \in E \tag{29}$$

$$y_i \ge 0, \qquad \forall\, i \in V \tag{30}$$

(25)-(30) specifically models the multiple sourcing version of the problem, by allowing any destination $j \in V$ to receive its shipment in parts from several supply sources $i \in U$. The single sourcing variant of the problem, which is also common in the literature, requires that any destination $j \in V$ receive its shipment from only one supply source $i \in U$. This is modelled by imposing a binary restriction on the x variables.

In this paper, we are interested in testing the efficacy of the IA algorithm, as described in Section 2, in solving the non-linear production-transportation problem in which the production cost $\phi_i(y_i)$ is concave. To the best of our knowledge, Sharp et al. (1970) was the first to study a non-linear production-transportation problem. However, the production cost $\phi_i(y_i)$ was assumed to be convex, which is relatively easier than its concave counterpart. The production-transportation problem can be viewed as a capacitated minimum cost network flow problem (MCNF) having (m) variables representing the production cost function and (mn) variables representing transportation cost function. For $m \ll n$, the production-transportation problem with concave production cost has a low-rank concavity (Konno et al., 1997). Guisewite and Pardalos (1993); Klinz and Tuy (1993); Kuno and Utsunomiya (1997); Kuno (1997); Tuy et al. (1993a,b, 1996) have proposed methods specifically suited when the problem has low-rank concavity. These methods belong to a group of polynomial or pseudo-polynomial algorithms in n, which do not scale well for $m > 3$. More scalable approaches are B&B based algorithms, which consist of two varieties. For the single source uncapacitated version of minimum concave cost network flow problem, Gallo et al. (1980b); Guisewite and Pardalos (1991) implicitly enumerate the spanning tree of the network.Falk and Soland (1969); Soland (1971); Horst (1976); Benson (1985); Locatelli and Thoai (2000) use linear underestimators to approximate the concave function, which is improved by dividing the feasible space. Later, Kuno and Utsunomiya (2000) proposed a Lagrangian relaxation-based B&B to solve the multiple sourcing production-transportation problems with concave cost. Subsequently, Saif (2016) used Lagrangian relaxation-based B&B approaches to solve both the multiple and single sourcing versions of the problem.

The literature on production-transportation problems has also seen several other variants/extensions of the basic problem. Holmberg and Tuy (1999) studied a production-transportation problem with concave production cost and convex transportation cost, resulting in a difference of convex (DC) optimization problem, which is solved using a B&B method. Nagai and Kuno (2005) studied production-transportation problems with inseparable concave production costs, which is solved using a B&B method. Condotta et al. (2013) studied a production scheduling-transportation problem with only one supply source and one destination. The objective of the problem is to schedule the production of a number of jobs with given release dates and processing times, and to schedule their transportation to the customer using a number of vehicles with limited capacity so as to minimize the maximum lateness.

Next, we describe our computational experiments on both the multiple and single sourcing versions of the production-transportation problem using our proposed IA algorithm.

4.1 Computational Experiments

In this section, we present the data generation technique, followed by computational results for the multiple sourcing and single sourcing versions of the production-transportation problem. The choice of the solver, platform, and server configuration remains the same as reported in Section 3.1. The experiments are set to terminate using $\epsilon = 0.01$ in step 4 of Algorithm 1 or a maximum CPU time limit, whichever reaches earlier. A maximum CPU time of 30 minutes is used for multiple sourcing, and that of 8.5 hours is used for single sourcing problems.

4.1.1 Data-Set

The data used in the experiments are generated using the scheme described by Kuno and Utsunomiya (2000). The concave cost function, $\phi_i(y_i) = \gamma\sqrt{y_i}$, where $\gamma \sim$ Uniform$\{10, 20\}$; number of sources, $m = |V| \in \{5, \dots, 25\}$ for multiple sourcing and $m = |V| \in \{5, \dots, 15\}$ for single sourcing; number of destinations, $n = |U| \in \{25, \dots, 100\}$; transportation cost, $c_{ij} \sim$ Uniform$\{1, 2, \dots, 10\}\ \forall\ (i, j) \in E$; production capacity at source i, $k_i = 200\ \forall\ i \in V$; demand at destination j, $d_j = \left\lceil \frac{\alpha \sum_{i \in V} k_i}{|U|} \right\rceil\ \forall\ j \in U$, where $\alpha \in \{0.60, 0.75, 0.90\}$ is a measure of capacity tightness.

4.1.2 Computational Results

Tables 5-7 provide a comparison of the computational performance of the IA algorithm against those reported by Kuno and Utsunomiya (2000) and Saif (2016). The columns Kuno and Utsunomiya (2000) and Saif (2016) represent the computational results reported by the respective authors. The missing values in some of the rows indicate that the authors did not provide results for the corresponding data instances. Since Both Kuno and Utsunomiya (2000) and Saif (2016) reported only the average and the maximum CPU times over 10 randomly generated test instances (each corresponding to a randomly selected pair of values of γ and c_{ij}) for each size of the problem, we also do the same for a meaningful comparison. For each problem size, the best average CPU time among the three methods is highlighted in boldface. Following observations can be immediately made from the tables: (i) Of the very selected instances for which Saif (2016) has reported the computational results, his method never performs the best except for a few very easy instances that can be solved within a fraction of a second. (ii) Between the remaining two methods, our IA algorithm outperforms Kuno and Utsunomiya (2000) on majority of the instances for which the results have been reported by the latter. When $\alpha = 0.75$, for which Kuno and Utsunomiya (2000) have reported their results across all the problem sizes used in our experiments (refer to Table 6), their method takes 58.49 seconds on average, compared to 6.07 seconds taken by our IA algorithm. To further see the difference between the two methods, we present their performance profiles (created based on the average CPU times) in Figure 9. The figure shows the IA algorithm to be better on 68.75% of the instances, while the method by Kuno and Utsunomiya (2000) performs better on the remaining 31.25%. Further, on the instances on which the method by Kuno and Utsunomiya (2000) performs worse, it is unable to solve around 50% of them to optimality even after taking 16 ($= 2^4$) times the CPU time taken by the IA algorithm.

For the production-transportation problem with single sourcing, we provide a comparison of the computational performance of the IA algorithm only with Saif (2016) since the study by Kuno and Utsunomiya (2000) is restricted to only the multiple sourcing version of the problem. The computational results of the two methods for the single sourcing version are reported in Tables 8-10. For each problem size, the better of the two average CPU times is highlighted in boldface. Once again, like the multiple sourcing case, missing values in some of the rows indicate that Saif (2016) did not provide results for those data instances. Please note that when the capacity is tight (i.e., α is high), the single sourcing constraints (i.e., $x_{ij} \in \{0, 1\}\ \forall(i, j) \in E$) become increasingly difficult to satisfy as $m = |V|$ starts

Table 5: Experimental results of production-transportation problem with multiple sourcing ($\alpha = 0.6$)

	CPU Time (seconds)					
	IA Algorithm		Kuno and Utsunomiya (2000)		Saif (2016)	
m×n	Avg	Max	Avg	Max	Avg	Max
5×25	0.63	1.72	**0.21**	0.37	0.35	0.66
5×50	1.30	3.58	1.65	2.43	**0.86**	1.89
5×75	1.24	2.69	-	-	-	-
5×100	1.78	4.19	-	-	-	-
10×25	**1.28**	3.11	3.13	8.03	2.43	5.41
10×50	**10.04**	30.35	71.46	239.17	20.43	34.48
10×75	16.78	98.36	-	-	-	-
10×100	87.50	341.31	-	-	-	-
15×25	5.29	31.35	**0.44**	1.25	-	-
15×50	**6.51**	18.64	87.68	260.82	-	-
15×75	188.26	880.14	-	-	-	-
15×100	77.77	228.87	-	-	-	-
20×75	188.22	1496.60	-	-	-	-
20×100	97.79	800.77	-	-	-	-
25×75	4.56	21.31	-	-	-	-
25×100	7.64	39.42	-	-	-	-
Avg	43.54	250.15	-	-	-	-

- denotes that the result is not provided by the respective author

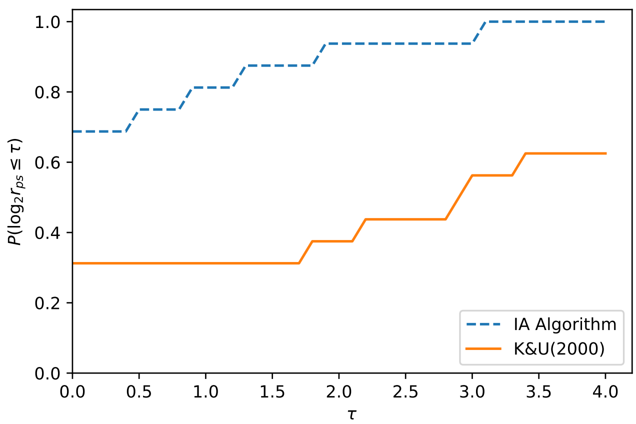

Figure 9: Performance profile of production-transportation problem with multiple sourcing for $\alpha = 0.75$

Table 6: Experimental results of production-transportation problem with multiple sourcing ($\alpha = 0.75$)

	CPU Time (seconds)					
	IA Algorithm		Kuno and Utsunomiya (2000)		Saif (2016)	
m×n	Avg	Max	Avg	Max	Avg	Max
5×25	0.18	0.40	**0.08**	0.18	0.09	0.17
5×50	0.31	0.68	1.04	1.50	**0.29**	0.55
5×75	**0.36**	0.65	6.20	10.38	-	-
5×100	**0.51**	1.12	19.25	30.48	-	-
10×25	2.53	13.66	**0.30**	0.78	0.61	3.06
10×50	**1.60**	4.94	6.85	11.20	7.84	35.47
10×75	**5.28**	23.43	55.41	115.10	-	-
10×100	**18.49**	74.50	334.64	1447.67	-	-
15×25	1.10	4.69	**0.30**	0.43	-	-
15×50	**1.12**	2.22	8.42	16.80	-	-
15×75	**3.10**	7.27	130.84	395.43	-	-
15×100	**3.26**	16.23	122.21	273.50	-	-
20×75	15.93	133.99	**11.85**	17.32	-	-
20×100	**18.48**	110.56	134.98	657.88	-	-
25×75	23.62	62.69	**12.76**	16.85	-	-
25×100	**1.29**	4.17	90.78	175.35	-	-
Avg	**6.07**	28.83	58.49	198.18	-	-

- denotes that the result is not provided by the respective author

Table 7: Experimental results of Production-Transportation problem with multiple sourcing ($\alpha = 0.9$)

	CPU Time (seconds)					
	IA Algorithm		Kuno and Utsunomiya (2000)		Saif (2016)	
m×n	Avg	Max	Avg	Max	Avg	Max
5×25	0.20	1.01	0.04	0.05	**0.03**	0.08
5×50	0.17	0.80	0.60	1.08	**0.08**	0.22
5×75	0.12	0.33	-	-	-	-
5×100	0.25	0.74	-	-	-	-
10×25	0.20	1.01	0.12	0.13	**0.07**	0.23
10×50	**0.24**	0.61	1.07	1.65	1.26	7.27
10×75	0.59	3.61	-	-	-	-
10×100	0.28	0.75	-	-	-	-
15×25	**0.19**	0.54	0.24	0.28	-	-
15×50	**0.21**	0.37	1.48	1.78	-	-
15×75	0.30	0.61	-	-	-	-
15×100	0.33	0.58	-	-	-	-
20×75	0.20	0.55	-	-	-	-
20×100	0.15	0.25	-	-	-	-
25×75	6.62	12.18	-	-	-	-
25×100	7.50	16.37	-	-	-	-
Avg	1.10	2.52	-	-	-	-

- denotes that the result is not provided by the respective author

approaching $n = |U|$. For, this reason, the instances of sizes $m = 10$, $n = 25$; $m = 15$, $n = 25$; and $m = 15$, $n = 50$ became infeasible for $\alpha = 0.9$, which are not reported in Table 10. Clearly, the IA algorithm outperforms the method by Saif (2016) by at least an order of magnitude on all the instances for which they have provided their results. We further test the efficacy of the IA method on even larger instances, the results for which are provided in Table 11. Some of these problem instances become computationally very difficult to solve, for which we set a maximum CPU time limit of 8.5 hours. Clearly, the IA algorithm is able to solve all these instances within less than a 1% optimality gap within the time limit.

Table 8: Experimental results of Production-Transportation problem with single sourcing ($\alpha = 0.6$)

	CPU Time (seconds)			
	IA Algorithm		Saif (2016)	
m×n	Avg	Max	Avg	Max
5×25	**0.75**	2.76	1.79	3.73
5×50	**1.21**	3.19	6.04	13.38
5×75	3.15	9.67	-	-
5×100	4.68	27.89	-	-
10×25	**2.69**	12.11	22.05	40.78
10×50	**46.90**	160.42	573.93	1710.85
10×75	220.10	851.83	-	-
15×25	28.62	134.83	-	-
15×50	1609.83	6364.17	-	-
Avg	213.10	840.76	-	-

- denotes that the result is not provided by the respective author

5 Conclusions

In this paper, we proposed an exact algorithm for solving concave minimization problems using a piecewise-linear inner-approximation of the concave function. The inner-approximation of the concave function results in a bilevel program, which is solved using a KKT-based approach. Our proposed algorithm guarantees improvement in the lower bound at each iteration and terminates at the global optimal solution. The algorithm has also been tested on two common application problems, namely, the concave knapsack problem and the production-transportation problem. Our extensive computational results show that our algorithm is able to significantly outperform the specialized methods that were reported in the literature for these two classes of problems. We believe that the algorithm will be useful for exactly solving a large number of other concave minimization applications for which practitioners often have to resort to customized methods or heuristics for solving the problem.

Table 9: Experimental results of Production-Transportation problem with single sourcing ($\alpha = 0.75$)

	CPU Time (seconds)			
	IA Algorithm		Saif (2016)	
m×n	Avg	Max	Avg	Max
5×25	**0.27**	0.63	1.44	2.59
5×50	**0.50**	1.06	4.17	6.65
5×75	0.87	1.31	-	-
5×100	14.10	68.45	-	-
10×25	**1.45**	8.50	27.92	53.88
10×50	**62.51**	327.83	455.51	1495.77
10×75	62.30	327.52	-	-
15×25	24.31	229.30	-	-
15×50	1211.16	8007.31	-	-
Avg	153.05	996.88	-	-

- denotes that the result is not provided by the respective author

Table 10: Experimental results of Production-Transportation problem with single sourcing ($\alpha = 0.9$)

	CPU Time (seconds)			
	IA Algorithm		Saif (2016)	
m×n	Avg	Max	Avg	Max
5×25	**0.10**	0.22	0.35	0.45
5×50	**0.57**	2.42	1.04	2.39
5×75	0.81	5.44	-	-
5×100	3.14	12.87	-	-
10×50	**0.12**	0.20	23.37	24.59
10×75	75.31	427.96	-	-
15×75	0.18	0.23	-	-
Avg	11.46	64.19	-	-

- denotes that the result is not provided by the respective author

Table 11: Experimental results of Production-Transportation problem with single sourcing

	Optimality Gap (%)			CPU Time (seconds)		
	m×n					
	10×100	15×75	15×100	10×100	15×75	15×100
			$\alpha = 0.6$			
Avg	0.01	0.05	0.06	16478.31	9146.75	18612.52
Min	0.00	0.00	0.00	1036.88	30.43	344.72
Max	0.05	0.40	0.27	30600.00	30600.00	30600.00
			$\alpha = 0.75$			
Avg	0.00	0.06	0.09	7042.87	25906.86	25568.05
Min	0.00	0.00	0.00	2.88	4.64	82.16
Max	0.02	0.25	0.25	30600.00	30600.00	30600.00
			$\alpha = 0.9$			
Avg	0.02	0.00	0.01	12320.55	0.18	12645.17
Min	0.00	0.00	0.00	2.60	0.16	1.91
Max	0.09	0.00	0.03	30600.00	0.23	30600.00

References

Benson, H. P. (1985). A finite algorithm for concave minimization over a polyhedron. *Naval Research Logistics Quarterly*, 32(1):165–177.

Benson, H. P. and Erenguc, S. S. (1990). An algorithm for concave integer minimization over a polyhedron. *Naval Research Logistics (NRL)*, 37(4):515–525.

Bitran, G. R. and Tirupati, D. (1989). Tradeoff curves, targeting and balancing in manufacturing queueing networks. *Operations Research*, 37(4):547–564.

Bretthauer, K. M., Ross, A., and Shetty, B. (1999). Nonlinear integer programming for optimal allocation in stratified sampling. *European Journal of Operational Research*, 116(3):667–680.

Bretthauer, K. M. and Shetty, B. (1995). The nonlinear resource allocation problem. *Operations Research*, 43(4):670–683.

Bretthauer, K. M., Victor Cabot, A., and Venkataramanan, M. (1994). An algorithm and new penalties for concave integer minimization over a polyhedron. *Naval Research Logistics (NRL)*, 41(3):435–454.

Caprara, A., Pisinger, D., and Toth, P. (1999). Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137.

Carrillo, M. J. (1977). A relaxation algorithm for the minimization of a quasiconcave function on a convex polyhedron. *Mathematical Programming*, 13(1):69–80.

Chaillou, P., Hansen, P., and Mahieu, Y. (1989). Best network flow bounds for the quadratic knapsack problem. In *Combinatorial Optimization*, pages 225–235. Springer.

Condotta, A., Knust, S., Meier, D., and Shakhlevich, N. V. (2013). Tabu search and lower bounds for a combined production–transportation problem. *Computers & Operations Research*, 40(3):886–900.

Dijkhuizen, G. and Faigle, U. (1993). A cutting-plane approach to the edge-weighted maximal clique problem. *European Journal of Operational Research*, 69(1):121–130.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Elhedhli, S. (2005). Exact solution of a class of nonlinear knapsack problems. *Operations Research Letters*, 33(6):615–624.

Falk, J. E. and Hoffman, K. R. (1976). A successive underestimation method for concave minimization problems. *Mathematics of Operations Research*, 1(3):251–259.

Falk, J. E. and Soland, R. M. (1969). An algorithm for separable nonconvex programming problems. *Management Science*, 15(9):550–569.

Fayard, D. and Plateau, G. (1982). An algorithm for the solution of the 0–1 knapsack problem. *Computing*, 28(3):269–287.

Fisher, M. L. (2004). The lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12_supplement):1861–1871.

Floudas, C., Aggarwal, A., and Ciric, A. (1989). Global optimum search for nonconvex NLP and MINLP problems. *Computers & Chemical Engineering*, 13(10):1117–1132.

Floudas, C. A., Pardalos, P. M., Adjiman, C., Esposito, W. R., Gümüs, Z. H., Harding, S. T., Klepeis, J. L., Meyer, C. A., and Schweiger, C. A. (1999). *Handbook of test problems in local and global optimization*, volume 33. Springer Science & Business Media.

Fontes, D. B. and Gonçalves, J. F. (2007). Heuristic solutions for general concave minimum cost network flow problems. *Networks: An International Journal*, 50(1):67–76.

Gallo, G., Hammer, P. L., and Simeone, B. (1980a). Quadratic knapsack problems. In *Combinatorial optimization*, pages 132–149. Springer.

Gallo, G., Sandi, C., and Sodini, C. (1980b). An algorithm for the min concave cost flow problem. *European Journal of Operational Research*, 4(4):248–255.

Guisewite, G. M. and Pardalos, P. M. (1990). Minimum concave-cost network flow problems: Applications, complexity, and algorithms. *Annals of Operations Research*, 25(1):75–99.

Guisewite, G. M. and Pardalos, P. M. (1991). Global search algorithms for minimum concave-cost network flow problems. *Journal of Global Optimization*, 1(4):309–330.

Guisewite, G. M. and Pardalos, P. M. (1993). A polynomial time solvable concave network flow problem. *Networks*, 23(2):143–147.

Han, X., Ma, N., Makino, K., and Chen, H. (2017). Online knapsack problem under concave functions. In *International Workshop on Frontiers in Algorithmics*, pages 103–114. Springer.

Holmberg, K. and Tuy, H. (1999). A production-transportation problem with stochastic demand and concave production costs. *Mathematical Programming*, 85(1):157–179.

Horst, R. (1976). An algorithm for nonconvex programming problems. *Mathematical Programming*, 10(1):312–321.

Horst, R. and Thoai, N. V. (1998). An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks. *Operations-Research-Spektrum*, 20(1):47–53.

Horst, R. and Tuy, H. (2013). *Global optimization: Deterministic approaches.* Springer Science & Business Media.

Johnson, E. L., Mehrotra, A., and Nemhauser, G. L. (1993). Min-cut clustering. *Mathematical Programming*, 62(1-3):133–151.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). Some selected applications. In *Knapsack Problems*, pages 449–482. Springer.

Klinz, B. and Tuy, H. (1993). Minimum concave-cost network flow problems with a single nonlinear arc cost. In *Network Optimization Problems: Algorithms, Applications and Complexity*, pages 125–145. World Scientific.

Kolesar, P. J. (1967). A branch and bound algorithm for the knapsack problem. *Management Science*, 13(9):723–735.

Konno, H., Thach, P. T., and Tuy, H. (1997). Low-rank nonconvex structures. In *Optimization on Low Rank Nonconvex Structures*, pages 95–117. Springer.

Kuno, T. (1997). A pseudo-polynomial algorithm for solving rank three concave production-transportation problems. *Acta Mathematica Vietnamica*, 22:159–182.

Kuno, T. and Utsunomiya, T. (1997). A pseudo-polynomial primal-dual algorithm for globally solving a production-transportation problem. *Journal of Global Optimization*, 11(2):163–180.

Kuno, T. and Utsunomiya, T. (2000). A lagrangian based branch-and-bound algorithm for production-transportation problems. *Journal of Global Optimization*, 18(1):59–73.

Li, D., Sun, X., and Wang, F. (2006). Convergent lagrangian and contour cut method for nonlinear integer programming with a quadratic objective function. *SIAM Journal on Optimization*, 17(2):372–400.

Li, X., Tomasgard, A., and Barton, P. I. (2011). Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of Optimization Theory and Applications*, 151(3):425.

Locatelli, M. and Thoai, N. V. (2000). Finite exact branch-and-bound algorithms for concave minimization over polytopes. *Journal of Global Optimization*, 18(2):107–128.

Majthay, A. and Whinston, A. (1974). Quasi-concave minimization subject to linear constraints. *Discrete Mathematics*, 9(1):35–59.

Maloney, B. M. and Klein, C. M. (1993). Constrained multi-item inventory systems: An implicit approach. *Computers & Operations research*, 20(6):639–649.

Marsten, R. E. and Morin, T. L. (1978). A hybrid approach to discrete mathematical programming. *Mathematical Programming*, 14(1):21–40.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424.

Mathur, K., Salkin, H. M., and Morito, S. (1983). A branch and search algorithm for a class of nonlinear knapsack problems. *Operations Research Letters*, 2(4):155–160.

Michelon, P. and Veilleux, L. (1996). Lagrangean methods for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 92(2):326–341.

Moré, J. J. and Vavasis, S. A. (1990). On the solution of concave knapsack problems. *Mathematical Programming*, 49(1-3):397–411.

Murty, K. G. (1968). Solving the fixed charge problem by ranking the extreme points. *Operations Research*, 16(2):268–279.

Nagai, H. and Kuno, T. (2005). A simplicial branch-and-bound algorithm for production-transportation problems with inseparable concave production cost. *Journal of the Operations Research Society of Japan*, 48(2):97–110.

Pardalos, P. M. and Rosen, J. B. (1987). *Constrained global optimization: algorithms and applications*, volume 268. Springer.

Park, K., Lee, K., and Park, S. (1996). An extended formulation approach to the edge-weighted maximal clique problem. *European Journal of Operational Research*, 95(3):671–682.

Pisinger, D. (2007). The quadratic knapsack problem—a survey. *Discrete Applied Mathematics*, 155(5):623–648.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Ryoo, H. S. and Sahinidis, N. V. (1996). A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138.

Saif, A. (2016). *Supply Chain Network Design with Concave Costs: Theory and Applications*. PhD thesis, University of Waterloo.

Sharp, J. F., Snyder, J. C., and Greene, J. H. (1970). A decomposition algorithm for solving the multifacility production-transportation problem with nonlinear production costs. *Econometrica: Journal of the Econometric Society*, pages 490–506.

Soland, R. M. (1971). An algorithm for separable nonconvex programming problems II: Nonconvex constraints. *Management Science*, 17(11):759–773.

Soland, R. M. (1974). Optimal facility location with concave costs. *Operations Research*, 22(2):373–382.

Strekalovsky, A. S. (2015). On local search in dc optimization problems. *Applied Mathematics and Computation*, 255:73–83.

Sun, X., Wang, F., and Li, D. (2005). Exact algorithm for concave knapsack problems: Linear underestimation and partition method. *Journal of Global Optimization*, 33(1):15–30.

Taha, H. A. (1973). Concave minimization over a convex polyhedron. *Naval Research Logistics Quarterly*, 20(3):533–548.

Tawarmalani, M. and Sahinidis, N. V. (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591.

Tuy, H. (1964). Concave programming under linear constraints. *Soviet Math.*, 5:1437–1440.

Tuy, H., Dan, N. D., and Ghannadan, S. (1993a). Strongly polynomial time algorithms for certain concave minimization problems on networks. *Operations Research Letters*, 14(2):99–109.

Tuy, H., Ghannadan, S., Migdalas, A., and VÄarbrand, P. (1993b). Strongly polynomial algorithm for a production-transportation problem with concave production cost. *Optimization*, 27(3):205–227.

Tuy, H., Ghannadan, S., Migdalas, A., and Värbrand, P. (1996). A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Mathematical Programming*, 72(3):229–258.

Ventura, J. A. and Klein, C. M. (1988). A note on multi-item inventory systems with limited capacity. *Operations Research Letters*, 7(2):71–75.

Victor Cabot, A. and Selcuk Erenguc, S. (1986). A branch and bound algorithm for solving a class of nonlinear integer programming problems. *Naval Research Logistics Quarterly*, 33(4):559–567.

Wang, F. (2019). A new exact algorithm for concave knapsack problems with integer variables. *International Journal of Computer Mathematics*, 96(1):126–134.

Witzgall, C. (1975). Mathematical methods of site selection for electronic message systems (ems). *STIN*, 76:18321.

Ziegler, H. (1982). Solving certain singly constrained convex optimization problems in production planning. *Operations Research Letters*, 1(6):246–252.

Zwart, P. B. (1974). Global maximization of a convex function with linear inequality constraints. *Operations Research*, 22(3):602–609.

Appendix

A Illustrative Example for Concavity in Objective Function

To illustrate the algorithm, we consider a small-size numerical example:

$$\min_{x} \quad \phi(x) = -5x_1^{\frac{3}{2}} + 8x_1 - 30x_2 \tag{31}$$
$$\text{subject to} \quad -9x_1 + 5x_2 \leq 9 \tag{32}$$
$$x_1 - 6x_2 \leq 6 \tag{33}$$
$$3x_1 + x_2 \leq 9 \tag{34}$$
$$x \in X = \{x_j \in \mathbb{Z}^n | 1 \leq x_j \leq 7, j = 1, 2\} \tag{35}$$

Iteration 1: We replace the concave function $-x_1^{\frac{3}{2}}$ by a new variable t_1.

$$\begin{aligned}
\min_{x} \quad & \phi(x) = 5t_1 + 8x_1 - 30x_2 \\
\text{subject to} \quad & -9x_1 + 5x_2 \leq 9 \\
& x_1 - 6x_2 \leq 6 \\
& 3x_1 + x_2 \leq 9 \\
& x \in X = \{x_j \in \mathbb{Z}^n | 1 \leq x_j \leq 7, j = 1, 2\} \\
& t_1 \geq -x_1^{\frac{3}{2}}
\end{aligned}$$

Next, we replace the concave constraints with inner-approximation generated using two points, $x_1 \in \{1, 7\}$, which gives us the relaxation of the problem (31)-(35) as bilevel program. Let $g(x_1) = -x_1^{\frac{3}{2}}$, then $g(1) = -1$, $g(7) = -18.52$.

$$\min_{x} \quad \phi(x) = 5t_1 + 8x_1 - 30x_2 \tag{36}$$
$$\text{subject to} \quad -9x_1 + 5x_2 \leq 9 \tag{37}$$
$$x_1 - 6x_2 \leq 6 \tag{38}$$
$$3x_1 + x_2 \leq 9 \tag{39}$$
$$x \in X = \{x_j \in \mathbb{Z}^n | 1 \leq x_j \leq 7, j = 1, 2\} \tag{40}$$
$$\mu \in \operatorname*{argmax}_{\mu} \{-\mu_1 - 18.52\mu_2 : \mu_1 + \mu_2 = 1, \mu_1 + 7\mu_2 = x_1, -\mu_1 \leq 0 - \mu_2 \leq 0\} \tag{41}$$
$$t_1 \geq -\mu_1 - 18.52\mu_2 \tag{42}$$

Let $\gamma_1, \gamma_2, \gamma_3,$ and, γ_4 be the Lagrange multipliers for the constraints in (41), then the KKT conditions for the lower level program in (41) can be written as follows:

$$1 + \gamma_1 + \gamma_2 - \gamma_3 = 0 \tag{43}$$
$$18.52 + \gamma_1 + 7\gamma_2 - \gamma_4 = 0 \tag{44}$$
$$-\mu_1\gamma_3 = 0 \tag{45}$$
$$-\mu_2\gamma_4 = 0 \tag{46}$$
$$\mu_1, \mu_2, \gamma_3, \gamma_4 \geq 0 \tag{47}$$
$$\gamma_1, \gamma_2 - \text{unrestricted} \tag{48}$$

We linearize equations (45) and (46) using the BigM values.

$$\mu_1 \leq MZ_1 \tag{49}$$
$$\gamma_3 \leq M(1 - Z_1) \tag{50}$$
$$\mu_2 \leq MZ_2 \tag{51}$$
$$\gamma_4 \leq M(1 - Z_2) \tag{52}$$
$$Z_1, Z_2 \in \{0, 1\} \tag{53}$$

The relaxed model for the original problem ((31)-(35)) is given below as a mixed integer linear program (MILP).

$$\begin{aligned} [EX1-1] \min\ & 5t_1 + 8x_1 - 30x_2 \\ \text{subject to } & t_1 \geq -\mu_1 - 18.52\mu_2 \\ & \mu_1 + 7\mu_2 = x_1 \\ & \mu_1 + \mu_2 = 1 \\ & (37) - (40), (43) - (44), (47) - (53) \end{aligned}$$

The above formulation can be solved using an MILP solver to arrive at the following solution, $x_1 = 2, x_2 = 3,$ objective value $= -93.6$. Hence, the lower bound is -93.6 and the upper bound is -88.14.
Iteration 2: The solution obtained from iteration 1 gives an additional point, $x_1 = 2$, to approximate $g(x_1) = -x_1^{\frac{3}{2}}$, where $g(2) = -2.83$. The updated problem with an additional point is given as follows:

$$\min_x \quad \phi(x) = 5t_1 + 8x_1 - 30x_2 \tag{54}$$

$$\text{subject to} \quad (37) - (40) \tag{55}$$

$$\mu \in \underset{\mu}{\operatorname{argmax}} \Big\{ -\mu_1 - 18.52\mu_2 - 2.83\mu_3 \ : \tag{56}$$

$$\sum_{i=1}^{3} \mu_i = 1, \mu_1 + 7\mu_2 + 2\mu_3 = x_1, -\mu_i \leq 0 \; \forall \; i = 1, 2, 3 \Big\} \tag{57}$$

$$t_1 \geq -\mu_1 - 18.52\mu_2 - 2.83\mu_3 \tag{58}$$

Let $\gamma_1, \gamma_2, \gamma_3, \gamma_4,$ and, γ_5 be the Lagrange Multipliers for the constraints in (57), the the following represents the KKT conditions for (57).

$$1 + \gamma_1 + \gamma_2 - \gamma_3 = 0 \tag{59}$$

$$18.52 + \gamma_1 + 7\gamma_2 - \gamma_4 = 0 \tag{60}$$

$$2.83 + \gamma_1 + 2\gamma_2 - \gamma_5 = 0 \tag{61}$$

$$-\mu_1\gamma_3 = 0 \tag{62}$$

$$-\mu_2\gamma_4 = 0 \tag{63}$$

$$-\mu_3\gamma_5 = 0 \tag{64}$$

$$\mu_1, \mu_2, \gamma_3, \gamma_4, \gamma_5 \geq 0 \tag{65}$$

$$\gamma_1, \gamma_2 - \text{unrestricted} \tag{66}$$

We once again linearize equation (62)- (64).

$$\mu_1 \leq MZ_1 \tag{67}$$

$$\gamma_3 \leq M(1 - Z_1) \tag{68}$$

$$\mu_2 \leq MZ_2 \tag{69}$$

$$\gamma_4 \leq M(1 - Z_2) \tag{70}$$

$$\mu_3 \leq MZ_3 \tag{71}$$

$$\gamma_5 \leq M(1 - Z_3) \tag{72}$$

$$Z_1, Z_2, Z_3 \in \{0, 1\} \tag{73}$$

A tighter relaxed problem for (31)-(35) as compared to the one in iteration 1 is given as follows:

$$\begin{aligned} [EX1-2] \min\ & 5t_1 + 8x_1 - 30x_2 \\ \text{subject to } & t_1 \geq -\mu_1 - 18.52\mu_2 - 2.83\mu_3 \\ & \mu_1 + \mu_2 + \mu_3 = 1 \\ & \mu_1 + 7\mu_2 + 2\mu_3 = x_1 \\ & (37) - (40), (59) - (61), (65) - (73) \end{aligned}$$

Solution of the above formulation is $x_1 = 2, x_2 = 3,$ objective value $= -88.15$. The lower bound is -88.15 and the upper bound is -88.14. Additional iterations would lead to further tightening of the bounds.

B Illustrative Example for Concavity in Constraints

The proposed algorithm can also solve the class of problems in which concavity is present in the constraints. We illustrate this using an example problem that has been taken from Floudas et al. (1999) (refer to Section 12.2.2 in the handbook).

$$\min_{x,y} \quad -0.7y + 5(x_1 - 0.5)^2 + 0.8 \tag{74}$$

$$\text{subject to} \quad x_2 \geq -e^{(x_1 - 0.2)} \tag{75}$$

$$x_2 - 1.1y \leq -1 \tag{76}$$

$$x_1 - 1.2y \leq 0.2 \tag{77}$$

$$0.2 \leq x_1 \leq 1 \tag{78}$$

$$-2.22554 \leq x_2 \leq -1 \tag{79}$$

$$y \in \{0, 1\} \tag{80}$$

The above problem has a convex objective function, but it is nonconvex because of equation (75). Let us start the iterations with two points, $x_1 \in \{0.2, 1\}$. Let $h(x_1) = -e^{(x_1-0.2)}$, then $h(0.2) = -1, h(1) = -2.22$. Next, we reformulate the problem (74)-(80) by replacing the concave constraint with its inner-approximation generated using two points.

$$\min_{x} \quad \phi(x) = -0.7y + 5(x_1 - 0.5)^2 + 0.8 \tag{81}$$

$$\text{subject to} \quad (76) - (80) \tag{82}$$

$$\mu \in \underset{\mu}{\text{argmax}} \{-\mu_1 - 2.22\mu_2 : \mu_1 + \mu_2 = 1, 0.2\mu_1 + \mu_2 = x_1, -\mu_1 \leq 0 - \mu_2 \leq 0\} \tag{83}$$

$$x_2 \geq -\mu_1 - 2.22\mu_2 \tag{84}$$

Let $\lambda_1, \lambda_2, \lambda_3, \text{and}, \lambda_4$ be the Lagrange multipliers of the constraints in (83) then KKT conditions for (83) can be written as:

$$1 + \lambda_1 + 0.2\lambda_2 - \lambda_3 = 0 \tag{85}$$

$$2.22 + \lambda_1 + \lambda_2 - \lambda_4 = 0 \tag{86}$$

$$-\mu_1\lambda_3 = 0 \tag{87}$$

$$-\mu_2\lambda_4 = 0 \tag{88}$$

$$\mu_1, \mu_2, \lambda_3, \lambda_4 \geq 0 \tag{89}$$

$$\lambda_1, \lambda_2 - \text{unrestricted} \tag{90}$$

We linearize equations (87) and (88) using a BigM value.

$$\mu_1 \leq MZ_1 \tag{91}$$

$$\lambda_3 \leq M(1 - Z_1) \tag{92}$$

$$\mu_2 \leq MZ_2 \tag{93}$$

$$\lambda_4 \leq M(1 - Z_2) \tag{94}$$

$$Z_1, Z_2 \in \{0, 1\} \tag{95}$$

At iteration 1 we solve the following quadratic program:

$$\begin{aligned} [EX2-1] \min \; & -0.7y + 5(x_1 - 0.5)^2 + 0.8 \\ \text{subject to } & x_2 \geq -\mu_1 - 2.22\mu_2 \\ & 0.2\mu_1 + \mu_2 = x_1 \\ & \mu_1 + \mu_2 = 1 \\ & (76) - (80), (85) - (86), (89) - (95) \end{aligned}$$

The solution of the $EX2-1$ is given as, $x_1 = 0.921, x_2 = -2.1, y = 1$, objective value $= 0.9875$. The above solution gives an additional point $x_1 = 0.921$ to approximate the $h(x_1) = e^{(x_1-0.2)}$, where

$h(0.921) = 2.06$. Hence the updated problem is as follows:

$$\min_{x} \quad \phi(x) = -0.7y + 5(x_1 - 0.5)^2 + 0.8 \tag{96}$$

$$\text{subject to} \quad (76) - (80) \tag{97}$$

$$\mu \in \underset{\mu}{\text{argmax}} \Big\{ -\mu_1 - 2.22\mu_2 - 2.06\mu_3 : \tag{98}$$

$$\sum_{i=1}^{3} \mu_i = 1, 0.2\mu_1 + \mu_2 + 0.921\mu_3 = x_1, -\mu_i \leq 0 \;\forall\, i = 1,2,3 \Big\} \tag{99}$$

$$x_2 \geq -\mu_1 - 2.22\mu_2 - 2.06\mu_3 \tag{100}$$

Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \text{and}, \gamma_5$ be Lagrange multipliers for the constraints of equation (99) then the corresponding KKT conditions are as follows:

$$1 + \lambda_1 + 0.2\lambda_2 - \lambda_3 = 0 \tag{101}$$

$$2.22 + \lambda_1 + \lambda_2 - \lambda_4 = 0 \tag{102}$$

$$2.06 + \lambda_1 + 0.921\lambda_2 - \lambda_5 = 0 \tag{103}$$

$$-\mu_1\lambda_3 = 0 \tag{104}$$

$$-\mu_2\lambda_4 = 0 \tag{105}$$

$$-\mu_3\lambda_5 = 0 \tag{106}$$

$$\mu_1, \mu_2, \lambda_3, \lambda_4, \lambda_5 \geq 0 \tag{107}$$

$$\lambda_1, \lambda_2 - \text{unrestricted} \tag{108}$$

Upon linearization of (104)-(106) using a BigM value we get:

$$\mu_1 \leq MZ_1 \tag{109}$$

$$\lambda_3 \leq M(1 - Z_1) \tag{110}$$

$$\mu_2 \leq MZ_2 \tag{111}$$

$$\lambda_4 \leq M(1 - Z_2) \tag{112}$$

$$\mu_3 \leq MZ_3 \tag{113}$$

$$\lambda_5 \leq M(1 - Z_3) \tag{114}$$

$$Z_1, Z_2, Z_3 \in \{0, 1\} \tag{115}$$

At iteration 2 we solve the following quadratic program:

$$\begin{aligned}
[EX2-2]\min \quad & -0.7y + 5(x_1 - 0.5)^2 + 0.8 \\
\text{subject to} \quad & x_2 \geq -\mu_1 - 2.22\mu_2 - 2.06\mu_3 \\
& \mu_1 + \mu_2 + \mu_3 = 1 \\
& 0.2\mu_1 + \mu_2 + 0.921\mu_3 = x_1 \\
& (76) - (80), (101) - (103), (107) - (115)
\end{aligned}$$

The solution of $EX2-2$ is $x_1 = 0.9419, x_2 = -2.1, y = 1,$ objective value $= 1.0769$.

The new point $x_1 = 0.9419$ is used in iteration 3, where the solution is $x_1 = 0.9419, x_2 = -2.1, y = 1$ and the lower bound is 1.0765. The algorithm can be terminated when the violation for the concave constraint is small. In this case, we stop further iterations of the algorithm. The known global optimal solution for the problem is $x_1 = 0.9419, x_2 = -2.1, y = 1$ with an optimal objective value of 1.0765 (Floudas et al., 1999).