# Journal Pre-proof


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

Ankur Sinha, Arka Das, Guneshwar Anand, Sachin Jayaswal

PII: S0377-2217(23)00121-2
DOI: https://doi.org/10.1016/j.ejor.2023.02.005
Reference: EOR 18339

To appear in: *European Journal of Operational Research*

Received date: 9 July 2021
Accepted date: 5 February 2023

Please cite this article as: Ankur Sinha, Arka Das, Guneshwar Anand, Sachin Jayaswal, A General Purpose Exact Solution Method for Mixed Integer Concave Minimization Problems, *European Journal of Operational Research* (2023), doi: https://doi.org/10.1016/j.ejor.2023.02.005

- We present an exact method for solving concave minimization problems.
- We use a bilevel program as an approximate representation of the problem.
- Solving the bilevel program iteratively guarantees convergence to global optimal.
- Concave knapsack and production-transportation problems are solved.
- The method is more than an order of magnitude faster than contemporary methods.

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

Ankur Sinha[a], Arka Das[a,*], Guneshwar Anand[b], Sachin Jayaswal[a]

[a]*Production and Quantitative Methods, Indian Institute of Management Ahmedabad, Gujarat, India, 380015*
[b]*Production and Operations Management, Indian Institute of Management Visakhapatnam, Andhra Pradesh, India, 530003*

**Abstract**

In this article, we discuss an exact algorithm for solving 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, for which we identify a tight value for general problems. 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:* Global Optimization, Concave Minimization, Mixed Integer Non-Linear Programming, Knapsack Problem, Production-Transportation Problem

## 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 (Holmberg & Tuy 1999, Kuno & Utsunomiya 2000, Saif 2016), facility location problems with concave costs (Soland 1974, Ni et al. 2021), location-inventory network design problems (Li et al. 2021, Farahani et al. 2015, Shen & Qi 2007, Jeet et al. 2009), concave minimum cost network flow problems (Guisewite & Pardalos 1990, Fontes & 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

*Corresponding author
*Email addresses:* asinha@iima.ac.in (Ankur Sinha), phd17arkadas@iima.ac.in (Arka Das), guneshwar.anand-phd19@iimv.ac.in (Guneshwar Anand), sachin@iima.ac.in (Sachin Jayaswal)

eliminated using Tuy cuts. However, the algorithm had a drawback since it was not guaranteed to be finite, which was tackled later by Zwart (1974), Majthay & Whinston (1974). Apart from cutting plane approaches, researchers also used relaxation-based ideas. For instance, Falk & Hoffman (1976), Carrillo (1977) computed successive underestimations of the concave function to solve 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 & Soland 1969, Horst 1976, Ryoo & Sahinidis 1996, Tawarmalani & 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 concave minimization problems, which requires few assumptions about the problem structure thereby making our approach general. We convert the concave minimization problem into a bilevel program that provides an approximate representation of the original problem. Solving bilevel programs over multiple iterations guarantee convergence to the global optimum of the original problem. While solving a bilevel program, one often attempts to reduce the program to a single level. Interestingly, in our approach converting a single level program to a bilevel program is the key to solving the original concave minimization problem. 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. The reduction of the bilevel program to a single level introduces a BigM, for which we identify a tight value. Though the identification of a tight value of BigM is often downplayed as a contribution, we would like to highlight it as a major contribution, given that it is applicable to general problems, and also because it has been recently shown by Kleinert et al. (2020) that identifying a tight BigM in similar contexts is an NP hard problem. The choice of an appropriate value for BigM, in our case, makes the method highly competitive and applicable to large dimensional mixed integer problems. The structure of the bilevel problem is exploited to arrive at a tight value of BigM for general problems. 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 and also provide a comparison of our algorithm against off-the-shelf solvers.

## 2. Algorithm Description

We consider optimization problems of the following kind

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

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

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

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 integer or continuous. The functions are assumed to be Lipschitz continuous. For a given set of $\tau$ sample points $S_c = \{z^1, z^2, \ldots, z^\tau\}$, where each $z^j$ is an $n$-dimensional point, the function $\phi(x)$ can be approximated as follows: (Rockafellar (1970)):

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

which is a linear program with $x$ as a parameter and $\mu$ as a decision vector for the linear program. For brevity, we will represent the approximation $\hat{\phi}(x|S_c)$ as $\hat{\phi}(x)$.

Figures 1 and 2 provide an illustration of the idea behind the working of the above linear program for piecewise approximation of a concave function in a single variable ($n = 1$). Figure 1 shows how a linear combination of the approximation points are used to represent the entire shaded region. Thereafter, Figure 2 shows how maximization in the shaded region leads to $\hat{\phi}(x)$ for any given $x$. Figure 3 again represents the piecewise approximation graphically and shows how it improves with the addition of a new point in the approximation set. Note that the feasible region with 5 points is a superset of the feasible region with 4 points. 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 value of the optimal solution of the formulation* (5)-(7) *provides a lower bound for the formulation* (1)-(3).

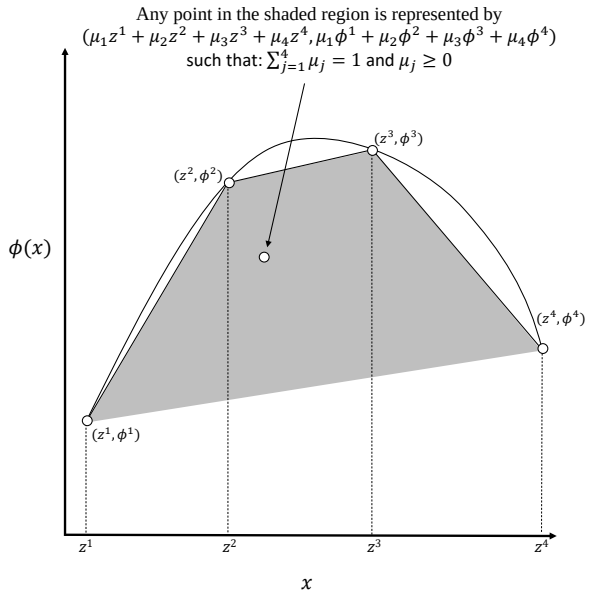


Figure 1: Linear combination of the corner points representing the entire shaded region.

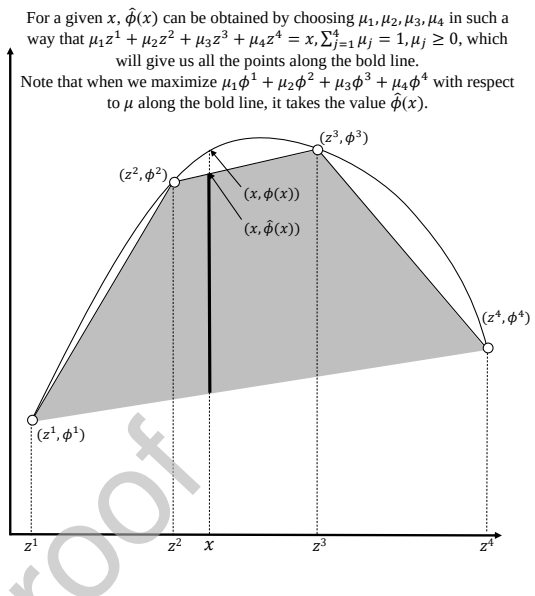


Figure 2: For a given $x$, maximization along the bold line leads to the value of $\hat{\phi}(x)$.

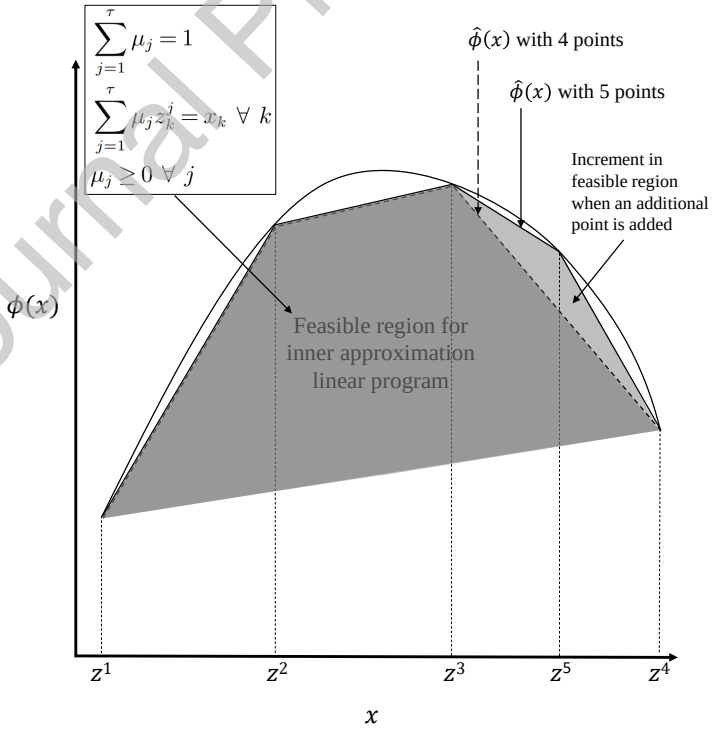


Figure 3: Inner-approximation of $\phi(x)$ with a set of points ($\tau = 4$ and $\tau = 5$).

*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:

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

□

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

$$\min_{x,\zeta} f(x) + \zeta \tag{8}$$

$$\text{subject to } \mu \in \underset{\mu}{\operatorname{argmax}} \left\{ \sum_{j=1}^{\tau} \mu_j \phi(z^j) : \sum_{j=1}^{\tau} \mu_j = 1, \sum_{j=1}^{\tau} \mu_j z_k^j = x_k, k = 1, \ldots, n, \mu_j \geq 0, j = 1, \ldots, \tau \right\} \tag{9}$$

$$\sum_{j=1}^{\tau} \mu_j \phi(z^j) \leq \zeta \tag{10}$$

$$g_i(x) \leq 0, \quad i = 1, \ldots, I \tag{11}$$

$$x_k^l \leq x_k \leq x_k^u, \quad k = 1, \ldots, n \tag{12}$$

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 $\alpha$, $\beta_k$ and $\gamma_j$ as the Lagrange multipliers for the constraints $\sum_{j=1}^{\tau} \mu_j = 1$, $\sum_{j=1}^{\tau} \mu_j z_k^j = x_k$ and $\mu_j \geq 0$, respectively, the above formulation reduces to the following.

$$\textbf{Mod-}S_c \qquad \min_{x,\alpha,\beta,\gamma,\zeta} f(x) + \zeta \tag{13}$$

$$\text{subject to} \tag{14}$$

$$g_i(x) \leq 0 \qquad i = 1, \ldots, I \tag{15}$$

$$\sum_{j=1}^{\tau} \mu_j \phi(z^j) \leq \zeta \tag{16}$$

$$\sum_{j=1}^{\tau} \mu_j = 1 \tag{17}$$

$$\sum_{j=1}^{\tau} \mu_j z_k^j = x_k \qquad k = 1, \ldots, n \tag{18}$$

$$\phi(z^j) - \alpha - \sum_{k=1}^{n} \beta_k z_j^k + \gamma_j = 0 \qquad j = 1, \ldots, \tau \tag{19}$$

$$\mu_j \gamma_j = 0 \qquad j = 1, \ldots \tau \tag{20}$$

$$\mu_j \geq 0 \qquad j = 1, \ldots \tau \tag{21}$$

$$\gamma_j \geq 0 \qquad j = 1, \ldots \tau \tag{22}$$

$$x_k^l \leq x_k \leq x_k^u \qquad k = 1, \ldots, n \tag{23}$$

Note that the above program contains product terms in the complementary slackness conditions ((20)),

which can be linearized using binary variables ($u$) and BigM ($M_1$ and $M_2$) as follows:

$$\gamma_j \leq M_1 u_j, \quad j = 1, \dots \tau \tag{24}$$

$$\mu_j \leq M_2(1 - u_j), \quad j = 1, \dots, \tau \tag{25}$$

$$u_j \in \{0, 1\}, \quad j = 1, \dots, \tau \tag{26}$$

Theorem 3 provides tight values for $M_1$ and $M_2$.

**Theorem 2.** *If $\phi(x)$ is Lipschitz continuous with Lipschitz constant K, then $\hat{\phi}(x|S_c) : x \in conv\ S_c$ 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 $\phi(x) : x \in \text{conv } S_c$, we can say that $||\omega|| \leq K\ \forall\ \omega \in \partial\phi(x)$, where $\partial\phi(x)$ represents the subgradient for $\phi(x)$. The function $\hat{\phi}(x|S_c) : x \in \text{conv } S_c$ is a concave polyhedral function, i.e. it consists of piecewise hyperplanes. Consider bounded polyhedra $X_j, j = 1, \dots, s$ on which the hyperplanes are defined, such that $\nabla\hat{\phi}(x)$ is constant in the interior of $X_j$. Note that $\hat{\phi}(x|S_c) = \phi(x)$ on the vertices, otherwise $\hat{\phi}(x|S_c) \leq \phi(x)$. 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 3.** *$M_1 = \psi^{max} = K||\Delta z||^{max} + \phi^{max} - \phi^{min}$ and $M_2 = 1$ are valid BigM values.*

*Proof.* From constraints (17) and (21), we observe that the maximum value that $\mu_j$ can take is 1. Hence, $M_2 = 1$ is acceptable. Next, let us look at the proof for $M_1$.

The value of $M_1$ can be set as a value larger than or equal to the maximum possible value that $\gamma_j$ may take. The dual of the lower lever problem in (9) can be written as

$$\min_{\alpha,\beta,\gamma} \sum_{k=1}^{n} \beta_k x_k + \alpha \tag{27}$$

$$\text{subject to} \sum_{k=1}^{n} \beta_k z_k^j + \alpha - \gamma_j = \phi(z^j), \qquad j = 1, \dots, \tau \tag{28}$$

$$\gamma_j \geq 0, \qquad j = 1, \dots, \tau \tag{29}$$

where $\alpha$ is the dual variable for the constraint $\sum_{j=1}^{\tau} \mu_j = 1$, $\beta_k$ is the dual for the constraint set $\sum_{j=1}^{\tau} \mu_j z_k^j = x_k$, and $\gamma_j$ is the dual for $\mu_j \geq 0$. Note that $\gamma_j$ being a dual for a non-negativity constraint in the primal is essentially a slack variable in the dual formulation. At the optimum, $\beta$ represents the normal to the inner-approximation plane on which the point $(x, \hat{\phi}(x))$ lies. At the optimum point of the above dual, at least one slack variable will be equal to zero. Without loss of generality, let us assume that $\gamma_1 = 0$.

$$\implies \sum_{k=1}^{n} \beta_k z_k^1 + \alpha = \phi(z^1) \tag{30}$$

$$\implies \alpha = \phi(z^1) - \sum_{k=1}^{n} \beta_k z_k^1 \tag{31}$$

By substituting $\alpha$ from (31), we arrive at the following equation for $j = 2$

$$\sum_{k=1}^{n}(\beta_k z_k^2 - \beta_k z_k^1) - \gamma_2 = \phi(z^2) - \phi(z^1) \tag{32}$$

$$\implies \gamma_2 = \sum_{k=1}^{n}(\beta_k z_k^2 - \beta_k z_k^1) + \phi(z^1) - \phi(z^2) \tag{33}$$

Given that $\beta$ is a normal to the inner-approximation plane, from Theorem 2, $\sum_{k=1}^{n}(\beta_k z_k^2 - \beta_k z_k^1) \leq K||z^2 - z^1||$, which implies that $\gamma_2^{max} = K||\Delta z||^{max} + \phi^{\max} - \phi^{\min}$. $K$ is the Lipschitz constant for $\phi$. $\phi^{\max}$ and $\phi^{\min}$ are the maximum and minimum values of $\phi$ in its domain, and $||\Delta z||^{max}$ is the maximum distance between any two values of $z$. Since $z = \{z_k : x_k^l \leq z_k \leq x_k^u, k = 1, \ldots, n\}$, $||\Delta z||^{max} = \sqrt{\sum_{k=1}^{n}(x_k^u - x_k^l)^2}$. □

Appropriate values for $M_1$ and $M_2$ are critical for the performance of the method, and therefore the above theorem plays an important role in making the proposed approach competitive. It has been recently shown (Kleinert et al. 2020) that identifying a BigM in similar contexts is an NP hard problem, but the structure of the bilevel program in our case allows the identification of tight values for these otherwise large numbers. After linearization of the complimentary slackness conditions, (13)-(19), (21)-(26) is a convex mixed integer program (MIP), which on solving to optimality generates a lower bound for the original program (1)-(3). Solving (13)-(19), (21)-(26) leads to $z^{\tau+1}$ as an optimal point, which is also a feasible point for the original problem (1)-(3). Therefore,substituting $x$ with $z^{\tau+1}$ 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 starts with an initial set of points $S_1 = \{z^1, \ldots, z^\tau\}$, such that

**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 $\tau$ points $S_c = \{z^1, z^2, \ldots, z^\tau\}$
**while** $((UB_{\mathcal{A}} - LB_{\mathcal{A}})/LB_{\mathcal{A}} > \epsilon$ ) begin **do**
    Solve Mod-$S_c$ ((13)-(23)) with a convex MIP solver after linearizing (20)
    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*

dom $\phi(x) \subseteq$ conv $S_1$.

### *2.1. The Initial Set*

In this section, we discuss the choice of the initial set $S_1 = \{z^1, \ldots, z^\tau\}$, 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 $x_k^l \leq x_k \leq x_k^u,\ k = 1, \ldots, n$. 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

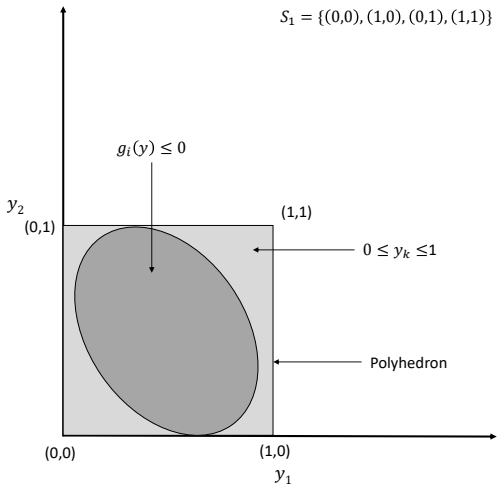


Figure 4: Smaller polyhedron with larger number of points.

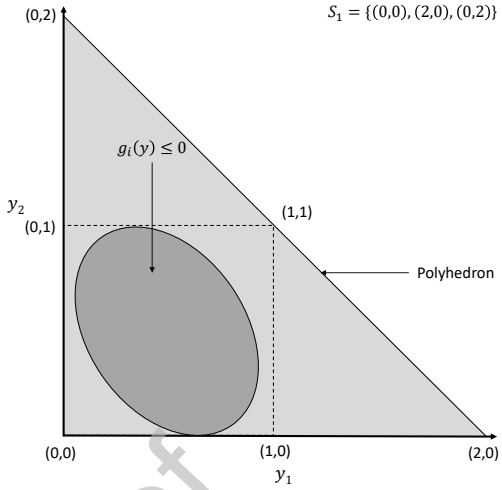


Figure 5: Larger polyhedron with smaller number of points.

$n$ 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 $0 \leq y_k \leq 1$. In such a case, Figure 4 shows the feasible region $g_i(y) \leq 0 \ \forall \ i$, enclosed in the polyhedron $0 \leq y_k \leq 1 \ \forall \ k$. Another polyhedron that encloses $g_i(y) \leq 0 \ \forall \ i$ completely is shown in Figure 5. While the polyhedron in Figure 4 is smaller in terms of the area (or volume), the polyhedron in Figure 5 is comparatively larger. However, the number of points required to form the polyhedron in Figure 4 for an $n$ dimensional problem would be $2^n$, whereas the polyhedron in Figure 5 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, $(0,0,\ldots,0), (n,0,\ldots,0), (0,n,\ldots,0), \ldots, (0,0,\ldots,n)$. 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 $g_i(x) \leq 0 \ \forall \ i$.

### *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 to the lower bound problem (Mod-$S_c$) in two consecutive iterations, then the solution is an optimal solution to the original concave minimization problem (1)-(3).

**Theorem 4.** *If two consecutive iterations $c$ and $c+1$ lead to the same solution to the lower bound problem (Mod-$S_c$), then the solution is optimal for* (1)-(3).

*Proof.* Say that $z^{\tau+c}$ is the solution at iteration $c$. Note that $S_{c+1} = S_c \cup z^{\tau+c}$, which implies that at iteration $c+1$, $\hat{\phi}(z^{\tau+c}|S_{c+1}) = \phi(z^{\tau+c})$, i.e. when a new point $z^{\tau+c}$ is added in the approximation set, the value of $\hat{\phi}$ and $\phi$ is the same at that point. From Theorem 1, at iteration $c+1$, $f(z^{\tau+c+1}) + \hat{\phi}(z^{\tau+c+1}|S_{c+1}) \leq f(z^{\tau+c+1}) + \phi(z^{\tau+c+1})$. Since $z^{\tau+c+1} = z^{\tau+c}$ and $\hat{\phi}(z^{\tau+c}|S_{c+1}) = \phi(z^{\tau+c})$, $f(z^{\tau+c+1}) + \hat{\phi}(z^{\tau+c+1}|S_{c+1}) \leq f(z^{\tau+c+1}) + \phi(z^{\tau+c+1})$ holds with an equality implying that $z^{\tau+c}$ is the optimal solution. □

**Theorem 5.** *When the algorithm proceeds from iteration c to c+1, then the lower bound for* (1)-(3) *improves if* $z^{\tau+c}$ *at iteration c is not the optimum for* (1)-(3).

*Proof.* It is given that $z^{\tau+c}$ is the solution for (5)-(7) at iteration $c$, which is not optimal for the original problem ((1)-(3)). Say that $z^{\tau+c+1}$ is the solution for (5)-(7) at iteration $c+1$, so from Theorem 4 we can say that $z^{\tau+c} \neq z^{\tau+c+1}$.

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

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

**Theorem 6.** *If* $z^{\tau+c}$ *is the solution for* (5)-(7) *at iteration c,* $z^{\tau+c+1}$ *is the solution for* (5)-(7) *at iteration* $c+1$*, and* $||z^{\tau+c+1} - z^{\tau+c}|| \leq \delta$*, then the optimal function value* $v^*$ *for* (1)-(3) *has the following property:* $0 \leq f(z^{\tau+c+1}) + \hat{\phi}(z^{\tau+c+1}|S_{c+1}) - v^* \leq (K_1 + K_2)\delta$*, where* $K_1$ *and* $K_2$ *are the Lipchitz constants for* $f(x)$ *and* $\phi(x)$*, respectively.*

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

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

If $K_2$ is the Lipschitz constant for $\phi(x)$, then it is also the Lipschitz constant for $\hat{\phi}(x)$ from Theorem 2. From the Lipschitz property,

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

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

To illustrate the working of the algorithm, an example has been provided in the Appendix (see Section Appendix 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 & 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

& Morin 1978), cut-and-branch algorithm (Fomeni et al. 2020), 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 & Tirupati 1989), production planning (Ziegler 1982, Ventura & Klein 1988, Maloney & 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 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 & Shetty 1995, Horst & Thoai 1998, Horst & Tuy 2013), and other problems with economies of scale (Pardalos & 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 & 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 & Morin 1978, Victor Cabot & Selcuk Erenguc 1986, Benson & 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 & 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 continuous concave knapsack problem with linear constraints. Michelon & 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 for solving CSINK. They used their proposed algorithm to solve only a single-dimensional CSINK, i.e., one with only one constraint. 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 a multidimensional CSINK, i.e., one with multiple knapsack constraints. Wang showed that his proposed algorithm outperformed the hybrid method proposed by Marsten & Morin (1978), which to the best of our knowledge, is the only other exact method to solve multidimensional CSINK.

The model for CSINK is described below:

$$\min_{x} \quad \phi(x) = \sum_{j=1}^{n} \phi_j(x_j) \tag{34}$$

subject to

$$Ax \leq b \tag{35}$$

$$x \in X = \{x \in \mathbb{Z}^n \mid l_j \leq x_j \leq u_j\} \tag{36}$$

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 lower and upper bounds of $x$ respectively. For $m \geq 1$, the above problem is commonly referred to as multidimensional CSINK. In the next section, we discuss the data-sets used for the computational experiments and present the results of the IA algorithm to solve multidimensional CSINK. 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 (34): (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 pair $n$ 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 pair $n$ 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*

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 | **5.62** | 0.32 | 20.48 | 17.85 | 0.41 | 75.64 |
| 40×10 | **3.80** | 0.16 | 10.24 | 50.98 | 2.05 | 350.94 |
| 50×10 | **3.99** | 0.81 | 12.74 | 142.39 | 1.34 | 980.34 |
| 80×10 | **19.32** | 1.61 | 40.32 | 793.83 | 14.81 | 7212.39 |
| 150×10 | **12.65** | 0.48 | 46.37 | 1116.95 | 0.01 | 3232.39 |
| 20×15 | **5.33** | 0.48 | 17.66 | 31.77 | 2.88 | 237.75 |
| 30×15 | **44.67** | 0.81 | 132.10 | 140.91 | 1.14 | 587.64 |
| 40×15 | **87.96** | 1.77 | 241.38 | 710.48 | 2.45 | 3125.30 |
| Avg | **22.92** | 0.81 | 65.16 | 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 | **10.74** | 0.25 | 35.82 | 10.91 | 0.25 | 26.80 |
| 40×10 | **10.98** | 0.18 | 34.50 | 30.73 | 1.30 | 98.47 |
| 60×10 | **31.18** | 0.93 | 92.38 | 166.62 | 6.72 | 1002.75 |
| 80×10 | **182.98** | 1.63 | 876.89 | 275.28 | 5.08 | 1672.88 |
| 90×10 | **209.05** | 0.30 | 1169.80 | 631.08 | 0.00 | 5457.95 |
| 20×15 | **29.51** | 1.47 | 117.04 | 153.48 | 1.41 | 1059.48 |
| 30×15 | **102.01** | 4.60 | 267.33 | 159.91 | 3.58 | 999.77 |
| 50×15 | **858.22** | 65.34 | 3432.73 | 2172.94 | 36.08 | 12747.97 |
| Avg | **179.33** | 9.34 | 753.31 | 450.12 | 6.80 | 2883.26 |

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

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 (34), 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 22.92 seconds on average, over all the instances, which is less than one-sixteenth 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 179.33 and

101.73, respectively, while the average times 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 (34). 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 1.48 seconds, which is around 114 times smaller than the average time of 169.18 seconds taken by Wang (2019).

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 | 23.22 | 0.88 | 89.33 | **13.09** | 2.02 | 35.81 |
| 50×10 | 57.44 | 0.16 | 223.17 | **44.92** | 4.61 | 153.17 |
| 70×10 | **41.63** | 1.17 | 200.05 | 396.58 | 8.03 | 1994.47 |
| 100×10 | **145.29** | 3.46 | 968.65 | 611.00 | 8.80 | 4963.13 |
| 20×15 | **77.74** | 7.24 | 272.28 | 117.90 | 3.03 | 402.61 |
| 30×15 | **88.55** | 0.88 | 434.04 | 188.39 | 3.22 | 1583.52 |
| 40×15 | **278.23** | 0.57 | 1081.13 | 443.32 | 6.09 | 1281.44 |
| Avg | **101.73** | 2.05 | 466.95 | 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 | **0.32** | 0.11 | 0.70 | 3.97 | 0.20 | 18.59 |
| 50×10 | **0.43** | 0.10 | 0.88 | 9.54 | 1.08 | 24.11 |
| 70×10 | **0.47** | 0.10 | 1.13 | 44.11 | 1.86 | 147.33 |
| 95×10 | **0.61** | 0.14 | 3.08 | 277.36 | 0.02 | 1484.20 |
| 30×15 | **1.05** | 0.12 | 4.59 | 12.70 | 0.94 | 39.55 |
| 50×15 | **2.43** | 0.32 | 9.69 | 289.21 | 9.11 | 1156.89 |
| 70×15 | **5.07** | 0.35 | 28.24 | 547.38 | 27.03 | 2100.30 |
| Avg | **1.48** | 0.18 | 6.90 | 169.18 | 5.75 | 710.14 |

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

To further see the difference in the performances of two methods, we present their performance profiles (Dolan & Moré 2002) in Figures 6-9. 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:

$$r_{p,s} = \frac{t_{p,s}}{\min\limits_{s \in S} t_{p,s}} \tag{37}$$

If we assume $r_{p,s}$ as a random variable, then the performance profile ($p_s(\tau)$) is the cumulative distribution function of $r_{p,s}$ at $2^\tau$, 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 $2^{\tau}$ times that taken by the best 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 6-8, and the same for the non-polynomial function are displayed in Figure 9. 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 6-8, it can be concluded that the IA algorithm outperforms Wang (2019) for 100% of the instances for the quadratic objective function and cubic objective function, while it is better for 71.42% of the instances for the quartic objective functions. For the non-polynomial (logarithmic) form of the objective function, Figure 9 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 that of 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.

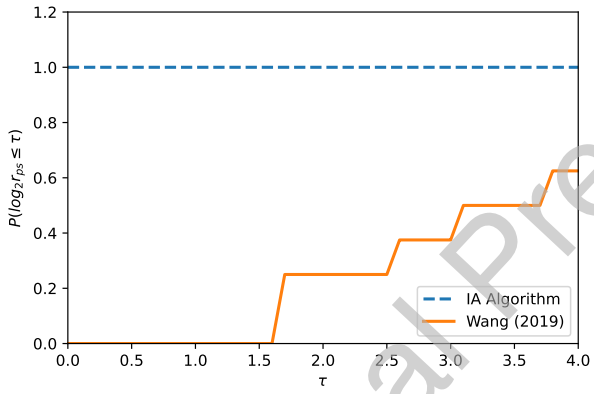


Figure 6: Performance profile of quadratic knapsack problem

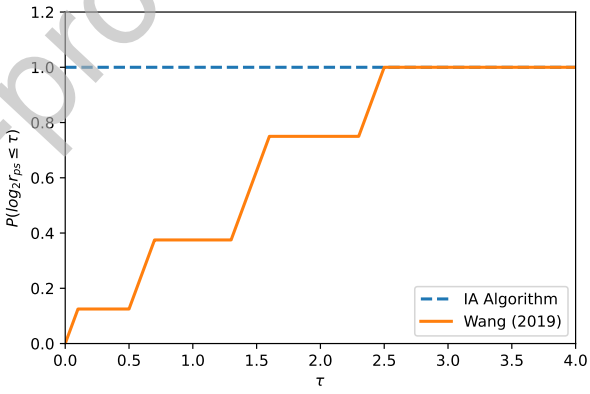


Figure 7: Performance profile of cubic knapsack problem

## 4. Production-Transportation Problem

The 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 the 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 non-negative 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

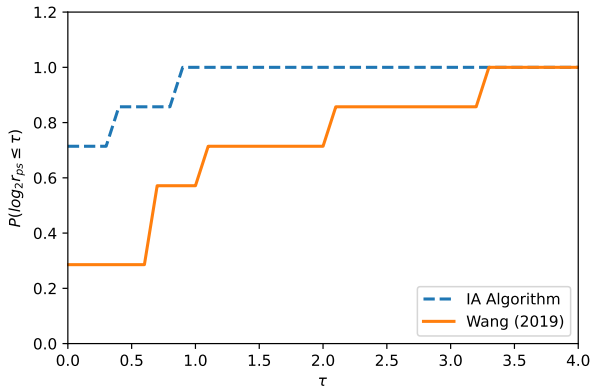


Figure 8: Performance profile of quartic knapsack problem

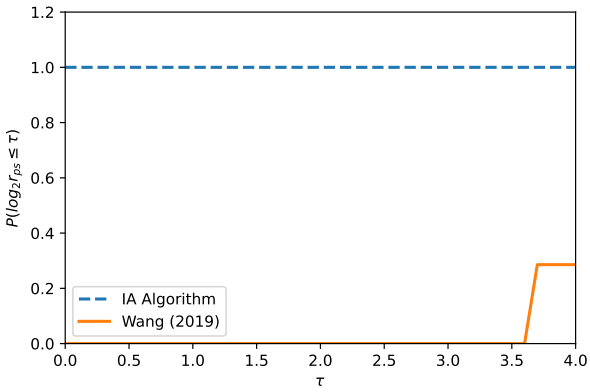


Figure 9: Performance profile of log knapsack problem

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:

$$\min_{x,y} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{i\in V} \phi_i(y_i) \tag{38}$$

subject to

$$\sum_{j\in U} x_{ij} \leq y_i, \qquad \forall\, i \in V \tag{39}$$

$$y_i \leq k_i, \qquad \forall\, i \in V \tag{40}$$

$$\sum_{i\in V} x_{ij} \geq d_j, \qquad \forall\, j \in U \tag{41}$$

$$x_{ij} \geq 0, \qquad \forall\, (i,j) \in E \tag{42}$$

$$y_i \geq 0, \qquad \forall\, i \in V \tag{43}$$

(38)-(43) 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 $j \in V$ receives its shipment from only one supply source $i \in U$. This is modelled by imposing a binary restriction on the $x$ variables in the model. The model for single sourcing is provided in Appendix D.

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) were the first to report 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 the 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 & Pardalos (1993), Klinz & Tuy (1993), Kuno & 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 & Pardalos (1991) implicitly enumerate the spanning tree of the network.Falk & Soland (1969), Soland (1971), Horst (1976), Benson (1985), Locatelli & Thoai (2000) use linear under-estimators to approximate the concave function, which is improved by partitioning the feasible space. Later, Kuno & 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. Recently, Wu et al. (2021) proposed a deterministic annealing neural network-based method and two neural networks to solve the multiple sourcing version of the production-transportation problem. The authors tested the method for problems with small dimensions. They neither tested the method on high dimensional data-sets nor compared the computational performances against existing methods.

The literature on production-transportation problems has also seen several other variants/extensions of the basic problem. Holmberg & 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 & 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 7 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 & Utsunomiya (2000). The concave cost function, $\phi_i(y_i) = \gamma\sqrt{y_i}$, where $\gamma \sim$ Uniform$\{10, 20\}$; number of sources, $m = |V| \in \{5, \ldots, 25\}$ for multiple sourcing and $m = |V| \in \{5, \ldots, 15\}$ for single sourcing; number of destinations, $n = |U| \in \{25, \ldots, 100\}$; transportation cost, $c_{ij} \sim$ Uniform$\{1, 2, \ldots, 10\} \forall\, (i, j) \in E$; 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 & Utsunomiya (2000) and Saif (2016). The columns Kuno & 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 & 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 $\gamma$ 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. The 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 & Utsunomiya (2000) on the majority of the instances for which the results have been reported by the latter. When $\alpha = 0.75$, for which Kuno & 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 5.06 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 10. The figure shows the IA algorithm to be better on 68.75% of the instances, while the method by Kuno & Utsunomiya (2000) performs better on the remaining 31.25%. Further, on the instances on which the method by Kuno & 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.

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

| | CPU Time (seconds) | | | | | |
|---|---|---|---|---|---|---|
| | IA Algorithm | | Kuno & Utsunomiya (2000) | | Saif (2016) | |
| m×n | Avg | Max | Avg | Max | Avg | Max |
| 5×25 | 0.52 | 1.44 | **0.21** | 0.37 | 0.35 | 0.66 |
| 5×50 | 1.09 | 2.99 | 1.65 | 2.43 | **0.86** | 1.89 |
| 5×75 | 1.03 | 2.24 | - | - | - | - |
| 5×100 | 1.48 | 3.49 | - | - | - | - |
| 10×25 | **1.06** | 2.59 | 3.13 | 8.03 | 2.43 | 5.41 |
| 10×50 | **8.37** | 25.29 | 71.46 | 239.17 | 20.43 | 34.48 |
| 10×75 | 13.99 | 81.97 | - | - | - | - |
| 10×100 | 72.92 | 284.43 | - | - | - | - |
| 15×25 | 4.41 | 26.13 | **0.44** | 1.25 | - | - |
| 15×50 | **5.42** | 15.53 | 87.68 | 260.82 | - | - |
| 15×75 | 156.88 | 733.45 | - | - | - | - |
| 15×100 | 64.81 | 190.73 | - | - | - | - |
| 20×75 | 156.85 | 1247.16 | - | - | - | - |
| 20×100 | 81.49 | 667.31 | - | - | - | - |
| 25×75 | 3.80 | 17.76 | - | - | - | - |
| 25×100 | 6.36 | 32.85 | - | - | - | - |
| Avg | 36.28 | 208.46 | - | - | - | - |

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

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 & 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

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

| | CPU Time (seconds) | | | | | |
|---|---|---|---|---|---|---|
| | IA Algorithm | | Kuno & Utsunomiya (2000) | | Saif (2016) | |
| m×n | Avg | Max | Avg | Max | Avg | Max |
| 5×25 | 0.15 | 0.34 | **0.08** | 0.18 | 0.09 | 0.17 |
| 5×50 | **0.26** | 0.57 | 1.04 | 1.50 | 0.29 | 0.55 |
| 5×75 | **0.30** | 0.54 | 6.20 | 10.38 | - | - |
| 5×100 | **0.43** | 0.93 | 19.25 | 30.48 | - | - |
| 10×25 | 2.11 | 11.39 | **0.30** | 0.78 | 0.61 | 3.06 |
| 10×50 | **1.33** | 4.12 | 6.85 | 11.20 | 7.84 | 35.47 |
| 10×75 | **4.40** | 19.52 | 55.41 | 115.10 | - | - |
| 10×100 | **15.41** | 62.08 | 334.64 | 1447.67 | - | - |
| 15×25 | 0.91 | 3.91 | **0.30** | 0.43 | - | - |
| 15×50 | **0.93** | 1.85 | 8.42 | 16.80 | - | - |
| 15×75 | **2.58** | 6.06 | 130.84 | 395.43 | - | - |
| 15×100 | **2.72** | 13.52 | 122.21 | 273.50 | - | - |
| 20×75 | 13.28 | 111.66 | **11.85** | 17.32 | - | - |
| 20×100 | **15.40** | 92.13 | 134.98 | 657.88 | - | - |
| 25×75 | 19.68 | 52.24 | **12.76** | 16.85 | - | - |
| 25×100 | **1.08** | 3.47 | 90.78 | 175.35 | - | - |
| Avg | **5.06** | 24.02 | 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 & Utsunomiya (2000) | | Saif (2016) | |
| m×n | Avg | Max | Avg | Max | Avg | Max |
| 5×25 | 0.17 | 0.84 | 0.04 | 0.05 | **0.03** | 0.08 |
| 5×50 | 0.15 | 0.67 | 0.60 | 1.08 | **0.08** | 0.22 |
| 5×75 | 0.10 | 0.27 | - | - | - | - |
| 5×100 | 0.21 | 0.62 | - | - | - | - |
| 10×25 | 0.17 | 0.84 | 0.12 | 0.13 | **0.07** | 0.23 |
| 10×50 | **0.20** | 0.51 | 1.07 | 1.65 | 1.26 | 7.27 |
| 10×75 | 0.49 | 3.01 | - | - | - | - |
| 10×100 | 0.23 | 0.63 | - | - | - | - |
| 15×25 | **0.16** | 0.45 | 0.24 | 0.28 | - | - |
| 15×50 | **0.18** | 0.31 | 1.48 | 1.78 | - | - |
| 15×75 | 0.25 | 0.51 | - | - | - | - |
| 15×100 | 0.27 | 0.48 | - | - | - | - |
| 20×75 | 0.17 | 0.46 | - | - | - | - |
| 20×100 | 0.12 | 0.21 | - | - | - | - |
| 25×75 | 5.52 | 10.15 | - | - | - | - |
| 25×100 | 6.25 | 13.64 | - | - | - | - |
| Avg | 0.91 | 2.10 | - | - | - | - |

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

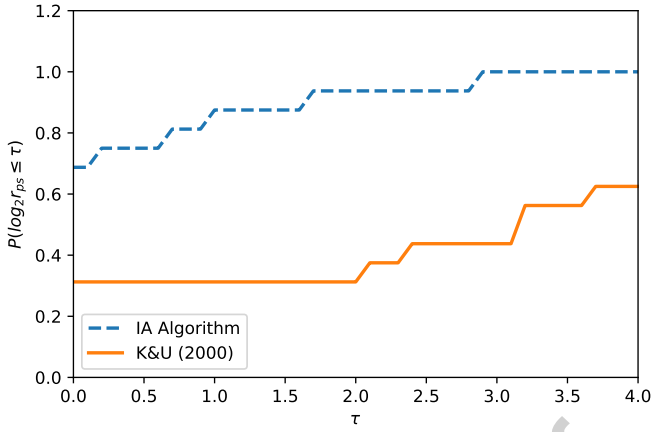


Figure 10: Performance profile of production-transportation problem with multiple sourcing for $\alpha = 0.75$

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., $\alpha$ 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 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 $\alpha = 0.9$, and the corresponding results are not reported in Table 10. Clearly, the IA algorithm outperforms the method by Saif (2016) by at least one order of magnitude on all the instances for which Saif (2016) have provided their results. We further test the efficacy of the IA method on even larger instances, the corresponding results 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 7 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 | **0.62** | 2.30 | 1.79 | 3.73 |
| 5×50 | **1.00** | 2.66 | 6.04 | 13.38 |
| 5×75 | 2.63 | 8.06 | - | - |
| 5×100 | 3.90 | 23.24 | - | - |
| 10×25 | **2.24** | 10.09 | 22.05 | 40.78 |
| 10×50 | **39.09** | 133.69 | 573.93 | 1710.85 |
| 10×75 | 183.42 | 709.86 | - | - |
| 15×25 | 23.85 | 112.36 | - | - |
| 15×50 | 1341.52 | 5303.48 | - | - |
| Avg | 177.59 | 700.64 | - | - |

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

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 | **0.23** | 0.52 | 1.44 | 2.59 |
| 5×50 | **0.42** | 0.89 | 4.17 | 6.65 |
| 5×75 | 0.73 | 1.10 | - | - |
| 5×100 | 11.75 | 57.04 | - | - |
| 10×25 | **1.21** | 7.08 | 27.92 | 53.88 |
| 10×50 | **52.09** | 273.19 | 455.51 | 1495.77 |
| 10×75 | 51.91 | 272.93 | - | - |
| 15×25 | 20.26 | 191.08 | - | - |
| 15×50 | 1009.3 | 6672.76 | - | - |
| Avg | 127.54 | 830.73 | - | - |

- 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 | **0.08** | 0.18 | 0.35 | 0.45 |
| 5×50 | **0.48** | 2.02 | 1.04 | 2.39 |
| 5×75 | 0.68 | 4.53 | - | - |
| 5×100 | 2.61 | 10.72 | - | - |
| 10×50 | **0.10** | 0.17 | 23.37 | 24.59 |
| 10×75 | 62.76 | 356.63 | - | - |
| 15×75 | 0.15 | 0.19 | - | - |
| Avg | 9.55 | 53.49 | - | - |

- 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 | 13570.38 | 7532.62 | 15327.96 |
| Min | 0.00 | 0.00 | 0 | 853.90 | 25.06 | 283.89 |
| Max | 0.05 | 0.40 | 0.27 | 25200.00 | 25200.00 | 25200.00 |
| | | | $\alpha = 0.75$ | | | |
| Avg | 0.00 | 0.06 | 0.09 | 5800.01 | 21335.06 | 21056.04 |
| Min | 0.00 | 0.00 | 0 | 2.37 | 3.82 | 67.66 |
| Max | 0.02 | 0.25 | 0.25 | 25200.00 | 25200.00 | 25200.00 |
| | | | $\alpha = 0.9$ | | | |
| Avg | 0.02 | 0.00 | 0.01 | 10146.33 | 0.14 | 10413.67 |
| Min | 0 | 0.00 | 0.00 | 2.14 | 0.13 | 1.57 |
| Max | 0.09 | 0.00 | 0.03 | 25200.00 | 0.19 | 25200.00 |

## 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. We make theoretical contributions by identifying a tight value of BigM for general problems that can help in efficiently solving the bilevel program that provides a lower bound to the original problem. 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.

## References


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Farahani, R. Z., Rashidi Bajgan, H., Fahimnia, B., & Kaviani, M. (2015). Location-inventory problem in supply chains: a modelling review. *International Journal of Production Research*, *53*, 3769–3788.

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

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

Floudas, C., Aggarwal, A., & Ciric, A. (1989). Global optimum search for nonconvex NLP and MINLP problems. *Computers & Chemical Engineering*, *13*, 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., & Schweiger, C. A. (1999). *Handbook of test problems in local and global optimization* volume 33. Springer Science & Business Media.

Fomeni, F. D., Kaparis, K., & Letchford, A. N. (2020). A cut-and-branch algorithm for the quadratic knapsack problem. *Discrete Optimization*, (p. 100579).

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

Gallo, G., Hammer, P. L., & Simeone, B. (1980a). Quadratic knapsack problems. In *Combinatorial Optimization* (pp. 132–149). Springer.

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

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

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

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

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

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

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

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

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

Jeet, V., Kutanoglu, E., & Partani, A. (2009). Logistics network design with inventory stocking for low-demand parts: Modeling and optimization. *IIE Transactions*, *41*, 389–407.

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

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

Kleinert, T., Labbé, M., Plein, F. a., & Schmidt, M. (2020). ThereâĂŹs no free lunch: on the hardness of choosing a correct big-m in bilevel optimization. *Operations Research*, *68*, 1716–1721.

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

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

Konno, H., Thach, P. T., & Tuy, H. (1997). Low-rank nonconvex structures. In *Optimization on Low Rank Nonconvex Structures* (pp. 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., & Utsunomiya, T. (1997). A pseudo-polynomial primal-dual algorithm for globally solving a production-transportation problem. *Journal of Global Optimization*, *11*, 163–180.

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

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

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

Li, Y., Lin, Y., & Shu, J. (2021). Location and two-echelon inventory network design with economies and diseconomies of scale in facility operating costs. *Computers & Operations Research*, *133*, 105347.

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

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

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

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

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

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

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

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

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

Nagai, H., & 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*, 97–110.

Ni, W., Shu, J., Song, M., Xu, D., & Zhang, K. (2021). A branch-and-price algorithm for facility location with general facility cost functions. *INFORMS Journal on Computing*, *33*, 86–104.

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

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

Pisinger, D. (2007). The quadratic knapsack problemâĂŤa survey. *Discrete Applied Mathematics*, *155*, 623–648.

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

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

Saif, A. (2016). *Supply Chain Network Design with Concave Costs: Theory and Applications*. Ph.D. thesis University of Waterloo. URL: `http://hdl.handle.net/10012/10121`.

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

Shen, Z.-J. M., & Qi, L. (2007). Incorporating inventory and routing costs in strategic location models. *European Journal of Operational Research*, *179*, 372–389.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Wu, Z., Gao, Q., Jiang, B., & Karimi, H. R. (2021). Solving the production transportation problem via a deterministic annealing neural network method. *Applied Mathematics and Computation*, *411*, 126518.

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

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

## Appendix A. Illustrative Example for Concavity in Objective Function

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

$$\min_{x} \quad \phi(x) = -5x_1^{\frac{3}{2}} + 8x_1 - 30x_2 \tag{A.1}$$

$$\text{subject to} \quad -9x_1 + 5x_2 \leq 9 \tag{A.2}$$

$$x_1 - 6x_2 \leq 6 \tag{A.3}$$

$$3x_1 + x_2 \leq 9 \tag{A.4}$$

$$x \in X = \{x_j \in \mathbb{Z}^n | 1 \leq x_j \leq 7, j = 1, 2\} \tag{A.5}$$

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 (A.1)-(A.5) as bilevel program. Let $g(x_1) = -x_1^{\frac{3}{2}}$, then $g(1) = -1$, $g(7) = -18.52$.

$$\min_{x} \quad \phi(x) = 5t_1 + 8x_1 - 30x_2 \tag{A.6}$$

$$\text{subject to} \quad -9x_1 + 5x_2 \leq 9 \tag{A.7}$$

$$x_1 - 6x_2 \leq 6 \tag{A.8}$$

$$3x_1 + x_2 \leq 9 \tag{A.9}$$

$$x \in X = \{x_j \in \mathbb{Z}^n | 1 \leq x_j \leq 7, j = 1, 2\} \tag{A.10}$$

$$\mu \in \underset{\mu}{\text{argmax}} \{-\mu_1 - 18.52\mu_2 : \mu_1 + \mu_2 = 1, \mu_1 + 7\mu_2 = x_1, -\mu_1 \leq 0 - \mu_2 \leq 0\} \tag{A.11}$$

$$t_1 \geq -\mu_1 - 18.52\mu_2 \tag{A.12}$$

Let $\gamma_1, \gamma_2, \gamma_3, \text{and}, \gamma_4$ be the Lagrange multipliers for the constraints in (A.11), then the KKT conditions for the lower level program in (A.11) can be written as follows:

$$1 + \gamma_1 + \gamma_2 - \gamma_3 = 0 \tag{A.13}$$
$$18.52 + \gamma_1 + 7\gamma_2 - \gamma_4 = 0 \tag{A.14}$$
$$-\mu_1\gamma_3 = 0 \tag{A.15}$$
$$-\mu_2\gamma_4 = 0 \tag{A.16}$$
$$\mu_1, \mu_2, \gamma_3, \gamma_4 \geq 0 \tag{A.17}$$
$$\gamma_1, \gamma_2 - \text{unrestricted} \tag{A.18}$$

We linearize equations (A.15) and (A.16) using the BigM values proposed in Theorem 3.

$$\mu_1 \leq MZ_1 \tag{A.19}$$
$$\gamma_3 \leq M(1 - Z_1) \tag{A.20}$$
$$\mu_2 \leq MZ_2 \tag{A.21}$$
$$\gamma_4 \leq M(1 - Z_2) \tag{A.22}$$
$$Z_1, Z_2 \in \{0, 1\} \tag{A.23}$$

The relaxed model for the original problem ((A.1)-(A.5)) is given below as a mixed integer linear program (MILP).

$$\begin{aligned}
[EX1-1]\, \min\ & 5t_1 + 8x_1 - 30x_2 \\
\text{subject to } & t_1 \geq -\mu_1 - 18.52\mu_2 \\
& \mu_1 + 7\mu_2 = x_1 \\
& \mu_1 + \mu_2 = 1 \\
& \text{(A.7)} - \text{(A.10)}, \text{(A.13)} - \text{(A.14)}, \text{(A.17)} - \text{(A.23)}
\end{aligned}$$

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:

$$\min_x \quad \phi(x) = 5t_1 + 8x_1 - 30x_2 \tag{A.24}$$
$$\text{subject to} \quad \text{(A.7)} - \text{(A.10)} \tag{A.25}$$
$$\mu \in \underset{\mu}{\text{argmax}} \left\{ -\mu_1 - 18.52\mu_2 - 2.83\mu_3 \ : \ \sum_{i=1}^{3} \mu_i = 1, \mu_1 + 7\mu_2 + 2\mu_3 = x_1, -\mu_i \leq 0 \ \forall\ i = 1, 2, 3 \right\} \tag{A.26}$$
$$t_1 \geq -\mu_1 - 18.52\mu_2 - 2.83\mu_3 \tag{A.27}$$

Let $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \text{and}, \gamma_5$ be the Lagrange multipliers for the constraints in (A.26), then the following represents the KKT conditions.

$$1 + \gamma_1 + \gamma_2 - \gamma_3 = 0 \tag{A.28}$$
$$18.52 + \gamma_1 + 7\gamma_2 - \gamma_4 = 0 \tag{A.29}$$
$$2.83 + \gamma_1 + 2\gamma_2 - \gamma_5 = 0 \tag{A.30}$$
$$-\mu_1\gamma_3 = 0 \tag{A.31}$$
$$-\mu_2\gamma_4 = 0 \tag{A.32}$$
$$-\mu_3\gamma_5 = 0 \tag{A.33}$$
$$\mu_1, \mu_2, \gamma_3, \gamma_4, \gamma_5 \geq 0 \tag{A.34}$$
$$\gamma_1, \gamma_2 - \text{unrestricted} \tag{A.35}$$

We once again linearize equation (A.31)- (A.33) using the BigM values proposed in Theorem 3.

$$\mu_1 \leq MZ_1 \tag{A.36}$$
$$\gamma_3 \leq M(1 - Z_1) \tag{A.37}$$
$$\mu_2 \leq MZ_2 \tag{A.38}$$
$$\gamma_4 \leq M(1 - Z_2) \tag{A.39}$$
$$\mu_3 \leq MZ_3 \tag{A.40}$$
$$\gamma_5 \leq M(1 - Z_3) \tag{A.41}$$
$$Z_1, Z_2, Z_3 \in \{0, 1\} \tag{A.42}$$

A tighter relaxed problem for (A.1)-(A.5) as compared to the one in iteration 1 is given as follows:

$$\begin{aligned}
[EX1-2] \min\ & 5t_1 + 8x_1 - 30x_2 \\
\text{subject to } & t_1 \geq -\mu_1 - 18.52\mu_2 - 2.83\mu_3 \\
& \mu_1 + \mu_2 + \mu_3 = 1 \\
& \mu_1 + 7\mu_2 + 2\mu_3 = x_1 \\
& \text{(A.7)} - \text{(A.10)}, \text{(A.28)} - \text{(A.30)}, \text{(A.34)} - \text{(A.42)}
\end{aligned}$$

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.

## Appendix 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). However, for problems with concavity in constraints we have not been

able to propose a tight value for BigM.

$$\min_{x,y} \quad -0.7y + 5(x_1 - 0.5)^2 + 0.8 \tag{B.1}$$
$$\text{subject to} \quad x_2 \geq -e^{(x_1 - 0.2)} \tag{B.2}$$
$$x_2 - 1.1y \leq -1 \tag{B.3}$$
$$x_1 - 1.2y \leq 0.2 \tag{B.4}$$
$$0.2 \leq x_1 \leq 1 \tag{B.5}$$
$$-2.22554 \leq x_2 \leq -1 \tag{B.6}$$
$$y \in \{0, 1\} \tag{B.7}$$

The above problem has a convex objective function, but it is nonconvex because of equation (B.2). 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 (B.1)-(B.7) by replacing the concave constraint with its inner-approximation generated using two points.

$$\min_{x} \quad \phi(x) = -0.7y + 5(x_1 - 0.5)^2 + 0.8 \tag{B.8}$$
$$\text{subject to} \quad \text{(B.3)} - \text{(B.7)} \tag{B.9}$$
$$\mu \in \underset{\mu}{\operatorname{argmax}} \{-\mu_1 - 2.22\mu_2 : \mu_1 + \mu_2 = 1, 0.2\mu_1 + \mu_2 = x_1, -\mu_1 \leq 0 - \mu_2 \leq 0\} \tag{B.10}$$
$$x_2 \geq -\mu_1 - 2.22\mu_2 \tag{B.11}$$

Let $\lambda_1, \lambda_2, \lambda_3,$ and, $\lambda_4$ be the Lagrange multipliers of the constraints in (B.10) then KKT conditions for (B.10) can be written as:

$$1 + \lambda_1 + 0.2\lambda_2 - \lambda_3 = 0 \tag{B.12}$$
$$2.22 + \lambda_1 + \lambda_2 - \lambda_4 = 0 \tag{B.13}$$
$$-\mu_1\lambda_3 = 0 \tag{B.14}$$
$$-\mu_2\lambda_4 = 0 \tag{B.15}$$
$$\mu_1, \mu_2, \lambda_3, \lambda_4 \geq 0 \tag{B.16}$$
$$\lambda_1, \lambda_2 - \text{unrestricted} \tag{B.17}$$

We linearize equations (B.14) and (B.15) using a BigM value.

$$\mu_1 \leq MZ_1 \tag{B.18}$$
$$\lambda_3 \leq M(1 - Z_1) \tag{B.19}$$
$$\mu_2 \leq MZ_2 \tag{B.20}$$
$$\lambda_4 \leq M(1 - Z_2) \tag{B.21}$$
$$Z_1, Z_2 \in \{0, 1\} \tag{B.22}$$

At iteration 1 we solve the following quadratic program:

$$
\begin{aligned}
[EX2-1]\min\ & -0.7y + 5(x_1 - 0.5)^2 + 0.8 \\
\text{subject to } & x_2 \geq -\mu_1 - 2.22\mu_2 \\
& 0.2\mu_1 + \mu_2 = x_1 \\
& \mu_1 + \mu_2 = 1 \\
& \text{(B.3)} - \text{(B.7)}, \text{(B.12)} - \text{(B.13)}, \text{(B.16)} - \text{(B.22)}
\end{aligned}
$$

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:

$$\min_{x} \quad \phi(x) = -0.7y + 5(x_1 - 0.5)^2 + 0.8 \tag{B.23}$$

$$\text{subject to} \quad \text{(B.3)} - \text{(B.7)} \tag{B.24}$$

$$\mu \in \underset{\mu}{\operatorname{argmax}} \left\{ -\mu_1 - 2.22\mu_2 - 2.06\mu_3 : \sum_{i=1}^{3} \mu_i = 1, 0.2\mu_1 + \mu_2 + 0.921\mu_3 = x_1, -\mu_i \leq 0 \ \forall\ i = 1,2,3 \right\} \tag{B.25}$$

$$x_2 \geq -\mu_1 - 2.22\mu_2 - 2.06\mu_3 \tag{B.26}$$

Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4,$ and, $\gamma_5$ be Lagrange multipliers for the constraints of equation (B.25) then the corresponding KKT conditions are as follows:

$$1 + \lambda_1 + 0.2\lambda_2 - \lambda_3 = 0 \tag{B.27}$$

$$2.22 + \lambda_1 + \lambda_2 - \lambda_4 = 0 \tag{B.28}$$

$$2.06 + \lambda_1 + 0.921\lambda_2 - \lambda_5 = 0 \tag{B.29}$$

$$-\mu_1\lambda_3 = 0 \tag{B.30}$$

$$-\mu_2\lambda_4 = 0 \tag{B.31}$$

$$-\mu_3\lambda_5 = 0 \tag{B.32}$$

$$\mu_1, \mu_2, \lambda_3, \lambda_4, \lambda_5 \geq 0 \tag{B.33}$$

$$\lambda_1, \lambda_2 - \text{unrestricted} \tag{B.34}$$

Upon linearization of (B.30)-(B.32) using a BigM value we get:

$$\mu_1 \leq MZ_1 \tag{B.35}$$
$$\lambda_3 \leq M(1 - Z_1) \tag{B.36}$$
$$\mu_2 \leq MZ_2 \tag{B.37}$$
$$\lambda_4 \leq M(1 - Z_2) \tag{B.38}$$
$$\mu_3 \leq MZ_3 \tag{B.39}$$
$$\lambda_5 \leq M(1 - Z_3) \tag{B.40}$$
$$Z_1, Z_2, Z_3 \in \{0, 1\} \tag{B.41}$$

At iteration 2 we solve the following quadratic program:

$$\begin{aligned} [EX2-2]\min \ & -0.7y + 5(x_1 - 0.5)^2 + 0.8 \\ \text{subject to} \ & x_2 \geq -\mu_1 - 2.22\mu_2 - 2.06\mu_3 \\ & \mu_1 + \mu_2 + \mu_3 = 1 \\ & 0.2\mu_1 + \mu_2 + 0.921\mu_3 = x_1 \\ & (\text{B.3}) - (\text{B.7}), (\text{B.27}) - (\text{B.29}), (\text{B.33}) - (\text{B.41}) \end{aligned}$$

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).

## Appendix C. IA Algorithm Versus Gurobi, BONMIN, and KNITRO

Non-convex quadratic programs can be solved to optimality by some commercial solvers, like Gurobi. The solvers exploit the quadratic terms in the formulation and convert the non-convex quadratic program into a bilinear program. The bilinear terms can then be handled using envelopes, like McCormick envelopes, in a spatial branching framework. However, this idea cannot be extended to optimization problems with general concave functions. Table C.12 provides a comparison of the computational performance of the proposed IA algorithm against Gurobi on the concave quadratic test case. Note that the computational experiments reported in the table have been carried out on the same PC as reported in Section 3.1. The actual computational times have been reported for both the approaches. Clearly, Gurobi is computationally more efficient than our proposed IA method as it exploits the quadratic structure of the functions in the problem. However, Gurobi solver cannot handle the other three classes (quartic, cubic, and logrithmic) of concave separable integer knapsack problems or the non-quadratic production-transportation problem discussed in this paper.

Similarly, we also compare our algorithm against commonly used non-linear solvers, like BONMIN and KNITRO, that lead to global optimality for convex optimization problems and deploy heuristics for non-convex optimization problems. BONMIN uses a hybrid outer-approximation based branch-and-cut algorithm, and KNITRO uses a mixed integer sequential quadratic programming algorithm with heuristics and

multi-start for handling non-convex problems. The results are reported through Tables C.13-C.15, where we observe that for most of the test problems, BONMIN and KNITRO are unable to solve all the instances to global optimality. Since BONMIN and KNITRO solved different set of instances to global optimality, we have compared BONMIN and KNITRO separately against IA algorithm. We have reported the percentage of instances that are solved to global optimality by BONMIN and KNITRO, and the CPU time required to solve these successful instances. The average CPU time for the successful instances is compared against the average CPU time required by the IA algorithm for the same instances. While IA algorithm outperforms BONMIN and KNITRO in terms of solving all the instances to global optimality, it also beats BONMIN in terms of CPU time. However, KNITRO, in a number of test problems, is better in terms of CPU time that it took to solve a certain percentage of the instances to global optimality. Higher dimensional test cases where KNITRO and BONMIN were not able to solve any of the instances successfully have been excluded from the tables.

Table C.12: Experimental result comparisons for quadratic knapsack problem

| | CPU Time (seconds) | | | | | |
|---|---|---|---|---|---|---|
| | IA Algorithm with CPLEX | | | Gurobi | | |
| Data set(n×m) | Avg | Min | Max | Avg | Min | Max |
| 30×10 | 2.01 | 0.12 | 7.34 | **0.25** | 0.08 | 0.54 |
| 40×10 | 1.36 | 0.06 | 3.67 | **0.32** | 0.04 | 0.66 |
| 50×10 | 1.43 | 0.29 | 4.57 | **0.35** | 0.10 | 0.93 |
| 80×10 | 6.93 | 0.58 | 14.45 | **0.62** | 0.11 | 2.10 |
| 150×10 | 4.53 | 0.17 | 16.62 | **0.50** | 0.22 | 0.97 |
| 20×15 | 1.91 | 0.17 | 6.33 | **0.28** | 0.12 | 0.57 |
| 30×15 | 16.01 | 0.29 | 47.35 | **0.70** | 0.27 | 1.69 |
| 40×15 | 31.53 | 0.64 | 86.52 | **1.75** | 0.16 | 7.68 |
| Avg | 8.21 | 0.29 | 23.36 | **0.60** | 0.14 | 1.89 |

Table C.13: IA algorithm converges for 100% of the instances on Concave Knapsack Problem. Comparison of CPU time for BONMIN against IA for instances that BONMIN solves globally, and comparison of KNITRO against IA for instances that KNITRO solves globally.

| | BONMIN | | IA Algorithm | KNITRO | | IA Algorithm |
|---|---|---|---|---|---|---|
| Function type | CPU Time (seconds) | Instances solved globally | CPU Time (seconds) | CPU Time (seconds) | Instances solved globally | CPU Time (seconds) |
| Quadratic | 37.24 | 60% | 10.56 | 5.11 | 70% | 10.17 |
| Quartic | 22.65 | 31% | 9.35 | 4.66 | 36% | 15.66 |
| Cubic | 38.14 | 50% | 24.67 | 4.99 | 60% | 37.52 |
| Logarithmic | 362.11 | 100% | 0.53 | 47.26 | 100% | 0.53 |

Table C.14: IA algorithm converges for 100% of the instances on production-transportation problem with multiple sourcing. Comparison of CPU time for BONMIN against IA for instances that BONMIN solves globally, and comparison of KNITRO against IA for instances that KNITRO solves globally.

| | BONMIN | | IA Algorithm | KNITRO | | IA Algorithm |
|---|---|---|---|---|---|---|
| Data(n×m) | CPU Time (seconds) | Instances solved globally | CPU Time (seconds) | CPU Time (seconds) | Instances solved globally | CPU Time (seconds) |
| | | | $\alpha = 0.6$ | | | |
| 5×25 | 0.10 | 20% | 0.26 | 0.10 | 40% | 0.54 |
| 5×50 | 0.10 | 60% | 1.05 | 0.07 | 90% | 1.19 |
| 5×75 | 0.11 | 80% | 1.05 | 0.08 | 80% | 0.92 |
| 5×100 | 0.16 | 100% | 1.48 | 0.05 | 90% | 1.35 |
| 10×25 | 0.20 | 10% | 1.30 | 0.20 | 10% | 0.25 |
| 10×50 | - | 0% | - | 0.61 | 10% | 7.38 |
| 10×75 | - | 0% | - | - | 0% | - |
| 10×100 | - | 0% | - | - | 0% | - |
| 15×25 | - | 0% | - | - | 0% | - |
| 15×50 | - | 0% | - | - | 0% | - |
| 15×75 | - | 0% | - | - | 0% | - |
| 15×100 | - | 0% | - | - | 0% | - |
| | | | $\alpha = 0.75$ | | | |
| 5×25 | 0.09 | 50% | 0.17 | 0.02 | 50% | 0.17 |
| 5×50 | 0.11 | 70% | 0.15 | 0.05 | 70% | 0.19 |
| 5×75 | 0.13 | 90% | 0.30 | 0.05 | 90% | 0.30 |
| 5×100 | 0.19 | 70% | 0.35 | 0.08 | 60% | 0.40 |
| 10×25 | 0.26 | 10% | 0.04 | 0.34 | 30% | 0.24 |
| 10×50 | - | 0% | - | 1.24 | 10% | 0.80 |
| 10×75 | - | 0% | - | 0.43 | 30% | 8.25 |
| 10×100 | - | 0% | - | 0.64 | 10% | 1.88 |
| 15×25 | 1.21 | 10% | 0.55 | 0.30 | 20% | 0.22 |
| 15×50 | 2.11 | 20% | 0.98 | - | 0% | - |
| 15×75 | - | 0% | - | - | 0% | - |
| 15×100 | - | 0% | - | - | 0% | - |
| | | | $\alpha = 0.9$ | | | |
| 5×25 | 0.06 | 90% | 0.18 | 0.05 | 80% | 0.19 |
| 5×50 | 0.10 | 100% | 0.15 | 0.05 | 100% | 0.15 |
| 5×75 | 0.14 | 90% | 0.11 | 0.05 | 100% | 0.10 |
| 5×100 | 0.16 | 70% | 0.21 | 0.05 | 70% | 0.18 |
| 10×25 | 0.18 | 60% | 0.23 | 0.13 | 60% | 0.23 |
| 10×50 | 0.61 | 20% | 0.43 | - | 0% | - |
| 10×75 | 0.17 | 10% | 3.02 | 0.09 | 10% | 3.02 |
| 10×100 | - | 0% | - | 0.13 | 20% | 0.46 |
| 15×25 | 0.46 | 40% | 0.09 | 0.27 | 20% | 0.19 |
| 15×50 | 0.72 | 10% | 0.12 | 2.25 | 10% | 0.12 |
| 15×75 | - | 0% | - | - | 0% | - |
| 15×100 | - | 0% | - | - | 0% | - |

Table C.15: IA algorithm converges for 100% of the instances on production-transportation problem with single sourcing. Comparison of CPU time for BONMIN against IA for instances that BONMIN solves globally, and comparison of KNITRO against IA for instances that KNITRO solves globally.

| | BONMIN | | IA Algorithm | KNITRO | | IA Algorithm |
|---|---|---|---|---|---|---|
| Data(n×m) | CPU Time (seconds) | Instances solved globally | CPU Time (seconds) | CPU Time (seconds) | Instances solved globally | CPU Time (seconds) |
| | | | $\alpha = 0.6$ | | | |
| 5×25 | 1.02 | 70% | 0.30 | 0.20 | 70% | 0.30 |
| 5×50 | 1.87 | 70% | 0.90 | 0.18 | 80% | 1.12 |
| 5×75 | 0.57 | 90% | 2.88 | 0.21 | 90% | 2.88 |
| 5×100 | 12.30 | 100% | 3.90 | 0.30 | 100% | 3.90 |
| 10×25 | 20.08 | 40% | 3.64 | 0.55 | 30% | 1.48 |
| 10×50 | 102.17 | 20% | 69.69 | 0.52 | 10% | 5.08 |
| 10×75 | 424.49 | 10% | 126.37 | - | 0% | - |
| 15×25 | - | 0% | - | 4.20 | 90% | 14.08 |
| 15×50 | - | 0% | - | 2.49 | 10% | 6.63 |
| | | | $\alpha = 0.75$ | | | |
| 5×25 | 70.86 | 100% | 0.23 | 0.38 | 100% | 0.23 |
| 5×50 | 13.70 | 90% | 0.40 | 0.35 | 100% | 0.42 |
| 5×75 | 2.62 | 80% | 0.69 | 0.66 | 100% | 0.73 |
| 5×100 | 6.54 | 90% | 10.62 | 1.69 | 100% | 11.75 |
| 10×25 | 308.61 | 80% | 1.31 | 0.91 | 70% | 1.48 |
| 10×50 | 3846.44 | 70% | 43.44 | 4.71 | 50% | 5.71 |
| 10×75 | - | 0% | - | 4.35 | 20% | 2.20 |
| 15×25 | 1239.75 | 20% | 1.04 | 1.19 | 80% | 1.43 |
| 15×50 | 8050.246 | 10% | 0.65 | 18.41 | 70% | 345.89 |
| | | | $\alpha = 0.9$ | | | |
| 5×25 | 29.90 | 100% | 0.08 | 0.31 | 100% | 0.08 |
| 5×50 | 117.54 | 100% | 0.48 | 1.01 | 100% | 0.48 |
| 5×75 | 14326.17 | 10% | 4.55 | 2.08 | 100% | 0.68 |
| 5×100 | 3583.63 | 80% | 3.20 | 2.85 | 90% | 2.88 |
| 10×50 | - | 0% | - | 4.03 | 100% | 0.10 |
| 10×75 | - | 0% | - | 18.30 | 80% | 80.90 |

## Appendix D. Production-Transportation Problem with Single Sourcing

In multiple sourcing, every destination may accept supply from multiple sources, but the destination can accept supply only from one source in case of single sourcing problem. Hence, in single sourcing binary restriction is imposed in $x_{ij}$ variable. The modified model for production-transportation problem can be stated as follows:

$$\min_{x,y} \quad \sum_{(i,j)\in E} c_{ij}x_{ij}d_j \; + \sum_{i\in V} \phi_i(y_i) \tag{D.1}$$

subject to

$$\sum_{j\in U} x_{ij}d_j \leq y_i, \qquad \forall\, i \in V \tag{D.2}$$

$$y_i \leq k_i, \qquad \forall\, i \in V \tag{D.3}$$

$$\sum_{i\in V} x_{ij} \geq 1, \qquad \forall\, j \in U \tag{D.4}$$

$$x_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in E \tag{D.5}$$

$$y_i \geq 0, \qquad \forall\, i \in\ V \tag{D.6}$$