A neural network approach to solve geometric programs with joint probabilistic constraints

Siham Tassouli, Abdel Lisser

A neural network approach to solve geometric programs with joint probabilistic constraints

Siham Tassouli*[a], Abdel Lisser[a]

[a] *Laboratoire des Signaux et Systèmes (L2S), CentraleSupélec, Université Paris Saclay, 3, rue Joliot Curie, 91192 Gif sur Yvette cedex, France*

Abstract

In this paper we study a dynamical neural network approach for solving geometric programs with joint probabilistic constraints (GPPC for short) with normally distributed coefficients and independent matrix row vectors. We proof the convergence and the stability of our neural network in the sense of Lyapunov. Finally, we use the proposed method to solve a geometric transportation problem.

Keywords: Stochastic geometric programming, Joint probabilistic constraints, Biconvex optimisation, Dynamical neural network, Lyapunov theory, Partial KKT system

2010 MSC: 00-01, 99-00

1. Introduction

Geometric programming (GP for short) is a special class of nonlinear programming problems. It is used to minimize or maximize functions expressed by posynomials subject to constraints of the same type.

In the paper, we study the following stochastic geometric programming prob-

*Corresponding author.

Email addresses: siham.tassouli@centralesupelec.fr (Siham Tassouli*), abdel.lisser@centralesupelec.fr (Abdel Lisser)

lem:

$$\begin{aligned} \min_{t \in \mathcal{R}^M_{++}} \quad & E[\Sigma_{i\in I_0} c_i \prod_{j=1}^{M} t_j^{a_{ij}}], \\ \text{s.t.} \qquad & \mathcal{P}(\Sigma_{i\in I_k} c_i \prod_{j=1}^{M} t_j^{a_{ij}} \leq 1, k=1,....,K) \geq 1-\epsilon. \end{aligned} \tag{1}$$

Where $c_i, i \in I_k$ is a $K \times n$ are uncorrelated normally distributed random variables i.e $c_i \sim \mathcal{N}(\mathbb{E}_i, \sigma_i^2)$. The coefficients $a_{ij}, i \in I_k, j = 1, ..., M$ are deterministic, and $1-\epsilon$ is a given probability level with $\epsilon \in (0, 0.5]$.

Geometric programming was first introduced in 1967 by Duffin et al. [1]. Since then, GP was widely used in several fields. Chiang et al. [2] use geometric programming to solve power control problems in wireless cellular or ad hoc networks. Li & Chen [3] deal with the problem of choosing channel doping profile in semiconductor via geometric programming. Hoburg & Abbeel [4] formulate conceptual-stage aircraft design problems as geometric problems. Vanderhaegen et al [5] design operational amplifiers using reversed geometric programming. Dupacova [6] studies geometric programming for metal cutting optimization problems. Singh et al. [7] use a posynomial delay model to give a worst case design scheme for spatial gate sizing. Geometric programming has also been used in business for profit maximization problem , see Kojić & Lukač. [8]. Liu et al. [9] address the optimization of biochemical systems using an improved geometric programming approach. Li et al. [10] use geometric programming as a solver to optimize mechatronic system design.

To solve geometric optimisation problems several approaches were introduced. A differential evolution algorithm is used by Wang et al. [11] to solve a deterministic geometric program. Liu et al. [12] discuss joint stochastic geometric programs, where the stochastic parameters are normally distributed and pairwise independent. They propose two approaches, namely piecewise linear approximation and a sequential convex optimization algorithm. Chiang & Boyd [13] show that the Lagrange dual problems of the channel capacity problem are actually geometric programs and hence give an upper and a lower bounds for the initial problem. Perelman & Amin [14] approximate the network control problem of tree water supply systems using GP then transform the control problem into a convex optimization problem using a logarithmic transformation of the objective function and the constraints. To overcome the uncertainty in the geometric problem data, Hsiung et al. [15] use robust geometric programming. The method consists in approximating the constraints of the robust model by a piecewise-linear convex approximation. Khanjani-Shiraz et al. [16] study a stochastic geometric programming problem with joint chance constraints with elliptically distributed random parameters where the constraints are dependent. They use the variable transformation, proposed by Cheng & Lisser in [17] and Jia et al. in [12], to come up with lower and upper bounds.

Several papers in the literature use artificial neural networks to solve optimisation problems. Villarrubia et al. [18] propose to approximate the objective function by neural network for nonlinear geometric problem. Nazemi & Tahmasbi [19] introduce a dynamical neural network for solving individual chance constrained optimization problems.

Our paper is organised as follows. In Section 2 a biconvex deterministic equivalent of problem (1) is given together with a related partial KKT system. Section 3 proposes a dynamical neural network to solved the biconvex problem and discusses its stability and convergence. Finally, numerical experiments are given in Section 4.

2. Deterministic equivalent problem and partial KKT System

2.1. Deterministic equivalent problem

Problem (1) can be rewritten by introducing auxiliary variables $y_k, k = 1, ..., K$ as follows [12]

$$\begin{aligned}
\min_{t\in\mathcal{R}^M_{++}} \quad & \mathbb{E}[\Sigma_{i\in I_0} c_i \prod_{j=1}^M t_j^{a_{ij}}],\\
\text{s.t.} \quad & \mathcal{P}(\Sigma_{i\in I_k} c_i \prod_{j=1}^M t_j^{a_{ij}} \le 1) \ge y_k, k=1,...,K, \\
& \prod_{k=1}^K y_k \ge 1-\epsilon,\\
& 0 \le y_k \le 1, k=1,...,K.
\end{aligned} \tag{2}$$

We introduce h_k, $k = 1,..,K$ defined by $h_k = \Sigma_{i\in I_k} c_i \prod_{j=1}^M t_j^{a_{ij}}$. Note that the mean of h_k is $\bar{h}_k = \Sigma_{i\in I_k} E_i \prod_{j=1}^M t_j^{a_{ij}}$ and $var(h_k) = \Sigma_{i\in I_k} \sigma_i^2 \prod_{j=1}^M t_j^{a_{ij}}$. A deterministic equivalent problem of (2) is then given by:

$$\begin{aligned}
\min_{t\in\mathcal{R}^M_{++}} \quad & \Sigma_{i\in I_0} \mathbb{E}_i \prod_{j=1}^M t_j^{a_{ij}},\\
\text{s.t.} \quad & \bar{h}_k + \phi^{-1}(y_k)\sqrt{var(h_k)} \le 1, k=1,...,K,\\
& \prod_{k=1}^K y_k \ge 1-\epsilon,\\
& 0 \le y_k \le 1, k=1,...,K,
\end{aligned} \tag{3}$$

where $\phi^{-1}(y_k)$ is the quantile of the standard normal distribution. An equivalent biconvex problem for (3) can be obtained with the log-transformation $r_j = log(t_j), j = 1,...,M$ and $x_k = log(y_k)$, $k = 1,...,K$:

$$\begin{aligned}
\min_{r\in\mathcal{R}^M} \quad & \Sigma_{i\in I_0}\mathbb{E}_i exp(\Sigma_{j=1}^M a_{ij} r_j),\\
\text{s.t.} \quad & \Sigma_{i\in I_k}(\mathbb{E}_i exp(\Sigma_{j=1}^M a_{ij} r_j))\\
& + \sqrt{\Sigma_{i\in I_k}\sigma_i^2 exp(\Sigma_{j=1}^M (2a_{ij} r_j + log(\phi^{-1}(e^{x_k})^2)))} \le 1, k=1,...,K,\\
& \Sigma_{k=1}^K x_k \ge log(1-\epsilon), x_k \le 0, k=1,...,K.
\end{aligned} \tag{4}$$

2.2. Partial KKT System

Let $f(r) = \Sigma_{i\in I_0}\mathbb{E}_i exp(\Sigma_{j=1}^M a_{ij} r_j)$, $l_k(r,x) = \Sigma_{i\in I_k}(\mathbb{E}_i exp(\Sigma_{j=1}^M a_{ij} r_j)) + \sqrt{\Sigma_{i\in I_k}\sigma_i^2 exp(\Sigma_{j=1}^M (2a_{ij} r_j + log(\phi^{-1}(e^{x_k})^2)))} - 1$, $h(r,x) = log(1-\epsilon) - \Sigma_{k=1}^K x_k$

and $g_k(r,x) = x_k$, we can then write problem (4) as follows:

$$\begin{aligned} \min_{r\in\mathcal{R}^M} \quad & f(r), \\ \text{s.t.} \quad & l_k(r,x) \leq 0, k = 1,..,K, \\ & h(r,x) \leq 0, \\ & g_k(r,x) \leq 0, k = 1,..,K. \end{aligned} \tag{5}$$

The feasible set of (5) is denoted by $\mathcal{U} = \{(r,x) \in \mathcal{R}^m \times \mathcal{R}^k | l_k(r,x) \leq 0, h(r,x) \leq 0$ and $g_k(r,x) \leq 0, k = 1,..,K\}$.

Let $\mathcal{U}(r) = \{x \in \mathcal{R}^k | l_k(r,x) \leq 0, h(r,x) \leq 0$ and $g_k(r,x) \leq 0, k = 1,..,K\}$ and $\mathcal{U}(x) = \{r \in \mathcal{R}^m | l_k(r,x) \leq 0, h(r,x) \leq 0$ and $g_k(r,x) \leq 0, k = 1,..,K\}$, we define the following sub-problems:

when x is fixed find r such that

$$\begin{aligned} \min \quad & f(r), \\ \text{s.t.} \quad & r \in \mathcal{U}(x), \end{aligned} \tag{6}$$

and when r is fixed find x such that

$$x \in \mathcal{U}(r). \tag{7}$$

Definition 1. (r^*, x^*) is called a partial optimum of (5) if $f(r^*) \leq f(r)$, $\forall r \in \mathcal{U}(x^*)$. This means that r^* is an optimal solution for (6) when $x = x^*$ and x^* is feasible for (7) when $r = r^*$.

Remark 1. In biconvex optimisation problems, there always exists a partial optimal solution [20].

Let $(r^*, x^*) \in \mathcal{R}^m \times \mathcal{R}^K$, if there exists $\lambda^{(1)}, \lambda^{(2)}$, $\mu_i^{(1)}, \mu_i^{(2)}$, $\gamma_i^{(1)}$ and $\gamma_i^{(2)}$, $i = 1,..,K$ such that:

$$\nabla f(r^*) + \sum_{i=1}^{K} \mu_i^{(1)} \nabla_r l_i(r^*, x^*) + \lambda^{(1)} \nabla_r h(r^*, x^*) + \sum_{i=1}^{K} \gamma_i^{(1)} \nabla_r g_i(r^*, x^*) = 0, \tag{8}$$

$$\mu^{(1)} \geq 0, \mu_i^{(1)} l_i(r^*, x^*) = 0, \lambda^{(1)} \geq 0, \lambda^{(1)} h(r^*, x^*) = 0, \gamma_i^{(1)} \geq 0, \gamma_i^{(1)} g_i(r^*, x^*) = 0, i = 1,..,K, \tag{9}$$

$$\sum_{i=1}^{K} \mu_i^{(2)} \nabla_x l_i(r^*, x^*) + \lambda^{(2)} \nabla_x h(r^*, x^*) + \sum_{i=1}^{K} \gamma_i^{(2)} \nabla_x g_i(r^*, x^*) = 0, \tag{10}$$

$$\mu^{(2)} \geq 0, \mu_i^{(2)} l_i(r^*, x^*) = 0, \lambda^{(2)} \geq 0, \lambda^{(2)} h(r^*, x^*) = 0, \gamma_i^{(2)} \geq 0, \gamma_i^{(2)} g_i(r^*, x^*) = 0, i = 1,..,K, \tag{11}$$

then (r^*, x^*) is called a partial KKT point of (5), where $\lambda^{(1)}, \lambda^{(2)}$, $\mu_i^{(1)}, \mu_i^{(2)}$, $\gamma_i^{(1)}$ and $\gamma_i^{(2)}$ are the Lagrange multipliers associated with $\nabla_r h$, $\nabla_x h$, $\nabla_r l_i$, $\nabla_x l_i$, $\nabla_r g_i$ and $\nabla_x g_i$ $i = 1, .., K$, respectively.

Let $(r^*, x^*) \in \mathcal{R}^M \times \mathcal{R}^K$ be a partial solution of (5), with respect to partial slater constraints qualification [20] at (r^*, x^*). Then (r^*, x^*) is a partial KKT point of (5) if and only if the partial KKT system (8)-(11) holds with $\lambda^{(1)} = \lambda^{(2)}$, $\mu^{(1)} = \mu^{(2)}$and $\gamma^{(1)} = \gamma^{(2)}$ [20, 21].

3. Dynamical neural network approach

In this section, we propose a dynamic neural network model to solve problem (5). Let $l(r,x) = \begin{pmatrix} l_1(r,x) \\ \vdots \\ l_K(r,x) \end{pmatrix}$ and $g(r,x) = \begin{pmatrix} g_1(r,x) \\ \vdots \\ g_K(r,x) \end{pmatrix}$.
The dynamical equation of the neural network is given by

$$\begin{aligned} \frac{dr}{dt} &= -(\nabla f(r) + \nabla_r l(r,x)^T(\mu + l(r,x))_+ + \nabla_r h(r,x)^T(\lambda + h(r,x))_+ \\ &\quad +\nabla_r g(r,x)^T(\gamma + g(r,x))_+), \qquad (12) \\ \frac{dx}{dt} &= -(\nabla_x l(r,x)^T(\mu + l(r,x))_+ + \nabla_x h(r,x)^T(\lambda + h(r,x))_+ \\ &\quad +\nabla_x g(r,x)^T(\gamma + g(r,x))_+), \qquad (13) \\ \frac{d\mu}{dt} &= (\mu + l(r,x))_+ - \mu, \qquad (14) \\ \frac{d\lambda}{dt} &= (\lambda + h(r,x))_+ - \lambda, \qquad (15) \\ \frac{d\gamma}{dt} &= (\gamma + g(r,x))_+ - \gamma. \qquad (16) \end{aligned}$$

We denote $z = (r, x, \mu, \lambda, \gamma)$ and define

$$\Phi(z) = \begin{bmatrix} -(\nabla f(r) + \nabla_r l(r,x)^T(\mu + l(r,x))_+ + \nabla_r h(r,x)^T(\lambda + h(r,x))_+ + +\nabla_r g(r,x)^T(\gamma + g(r,x))_+) \\ -(\nabla_x l(r,x)^T(\mu + l(r,x))_+ + \nabla_x h(r,x)^T(\lambda + h(r,x))_+ + \nabla_x g(r,x)^T(\gamma + g(r,x))_+) \\ (\mu + l(r,x))_+ - \mu \\ (\lambda + h(r,x))_+ - \lambda \\ (\gamma + g(r,x))_+ - \gamma \end{bmatrix}$$

We can rewrite the neural network defined in (12)-(16) as

$$\begin{cases} \frac{dz}{dt} = \kappa\Phi_(z), \\ z(t_0) = z_0, \end{cases}$$

where κ is a scale parameter and indicates the convergence rate of the neural network (12)-(16). For the sake of simplicity, we set $\kappa = 1$. The interconnections between the different inputs of the neural network (12)-(16) are represented in Figure 1.

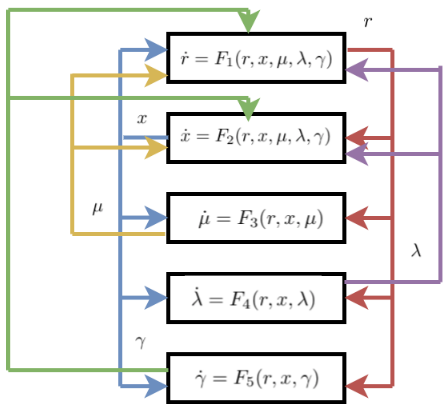
r
$\dot{r} = F_1(r, x, \mu, \lambda, \gamma)$
x
$\dot{x} = F_2(r, x, \mu, \lambda, \gamma)$
μ
$\dot{\mu} = F_3(r, x, \mu)$
λ
$\dot{\lambda} = F_4(r, x, \lambda)$
γ
$\dot{\gamma} = F_5(r, x, \gamma)$

Figure 1: Feedback interconnection of neural network (12)-(16)

Now, we study the convergence and the stability of the proposed dynamical neural network.

Theorem 2. Let $(r^*, x^*, \mu^*, \lambda^*, \gamma^*)$ an equilibrium point of the neural network defined by (12)-(16), then (r^*, x^*) is a partial KKT point of (5). On the other hand, if $(r^*, x^*) \in \mathcal{R}^M \times \mathcal{R}^K$ is a partial KKT point of (5), then there exists $\mu^* \geq 0$, $\lambda^* \geq 0$ and $\gamma^* \geq 0$ such that $(r^*, x^*, \mu^*, \lambda^*, \gamma^*)$ is an equilibrium point of the Neural Network (12)-(16).

Proof. Let $(r^*, x^*, \mu^*, \lambda^*, \gamma^*)$ an equilibrium point of the neural network defined by (12)-(16). Then, $\frac{dr^*}{dt} = 0$, $\frac{dx^*}{dt} = 0$, $\frac{d\mu^*}{dt} = 0$, $\frac{d\lambda^*}{dt} = 0$ and $\frac{d\gamma^*}{dt} = 0$. We have
$-(\nabla f(r^*) + \nabla_r l(r^*, x^*)^T(\mu^* + l(r^*, x^*))_+ + \nabla_r h(r^*, x^*)^T(\lambda^* + h(r^*, x^*))_+ + \nabla_r g(r^*, x^*)^T(\gamma^* + g(r^*, x^*))_+) = 0$,
$-(\nabla_x l(r^*, x^*)^T(\mu^* + l(r^*, x^*))_+ + \nabla_x h(r^*, x^*)^T(\lambda^* + h(r^*, x^*))_+ + \nabla_x g(r^*, x^*)^T(\gamma^* + g(r^*, x^*))_+) = 0$,
$(\mu^* + l(r^*, x^*))_+ - \mu^* = 0$, $(\lambda^* + h(r^*, x^*))_+ - \lambda^* = 0$ and $(\gamma^* + g(r^*, x^*))_+ - \gamma^* = 0$

We note that $(\mu^* + l(r^*, x^*))_+ - \mu^* = 0$ *if and only if* $\mu^* \geq 0$, $l(r^*, x^*) \leq 0$ and ${\mu^*}^T l(r^*, x^*) = 0$. By the same approach we have, $\lambda^* \geq 0$, $h(r^*, x^*) \leq 0$, ${\lambda^*}^T h(r^*, x^*) = 0$, $\gamma^* \geq 0$, $g(r^*, x^*) \leq 0$ and ${\gamma^*}^T g(r^*, x^*) = 0$.
Therefore, we obtain the partial KKT system (6).The converse part of the theorem is straightforward. $\square$

In order to prove the stability and the convergence of the neural network, we show first that the jacobian matrix $\nabla\Phi(z)$ is negative semidefinite matrix.

Lemma 3. The jacobian matrix $\nabla\Phi(z)$ is negative semidefinite matrix.

Proof. We assume the existence of $0 < p, q < K$ such that
$(\mu - l)_+ = (\mu_1 + l_1(r, x), \mu_2 + l_2(r, x), ..., \mu_p + l_p(r, x), \underbrace{0, ..., 0}_{K-p})$,
$(\gamma - g)_+ = (\gamma_1 + g_1(r, x), \gamma_2 + g_2(r, x), ..., \gamma_q + g_q(r, x), \underbrace{0, ..., 0}_{K-q})$.
Without loss of generality we consider the case where $\lambda + h(r, x) \neq 0$. We represent the jacobian matrix $\nabla\Phi$ as follows

$$\nabla\Phi(z) = \begin{bmatrix} A_1 & A_2 & A_3 & A_4 & A_5 \\ B_1 & B_2 & B_3 & B_4 & B_5 \\ C_1 & C_2 & C_3 & C_4 & C_5 \\ D_1 & D_2 & D_3 & D_4 & D_5 \\ E_1 & E_2 & E_3 & E_4 & E_5 \end{bmatrix},$$

where
$A_1 = -(\nabla^2 f(r) + \sum_{i=1}^{p}((\mu_i + l_i)\nabla_r^2 l_i^p(r, x)) + \nabla_r l^p(r, x)^T \nabla_r l^p(r, x)) + (\lambda + h)\nabla_r^2 h(r, x) + \nabla_r h(r, x)^T \nabla_r h(r, x)) + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_r^2 g_i^q(r, x)) + \nabla_r g^q(r, x)^T \nabla_r g^q(r, x))$,
$A_2 = -(\sum_{i=1}^{p}((\mu_i + l_i)\nabla_x \nabla_r l_i^p(r, x)) + \nabla_x l^p(r, x)^T \nabla_r l^p(r, x)) + (\lambda + h)\nabla_x \nabla_r h(r, x) + \nabla_x h(r, x)^T \nabla_r h(r, x)) + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_x \nabla_r g_i^q(r, x)) + \nabla_x g^q(r, x)^T \nabla_r g^q(r, x))$,
$B_1 = -(\sum_{i=1}^{p}((\mu_i + l_i)\nabla_r \nabla_x l_i^p(r, x)) + \nabla_r l^p(r, x)^T \nabla_x l^p(r, x)) + (\lambda + h)\nabla_r \nabla_x h(r, x) + \nabla_r h(r, x)^T \nabla_x h(r, x)) + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_r \nabla_x g_i^q(r, x)) + \nabla_r g^q(r, x)^T \nabla_x g^q(r, x))$,
$B_2 = -(\sum_{i=1}^{p}((\mu_i + l_i)\nabla_x^2 l_i^p(r, x)) + \nabla_x l^p(r, x)^T \nabla_x l^p(r, x)) + (\lambda + h)\nabla_x^2 h(r, x) + \nabla_x h(r, x)^T \nabla_x h(r, x)) + \sum_{i=1}^{q}((\gamma_i + g_i)\nabla_x^2 g_i^q(r, x)) + \nabla_x g^q(r, x)^T \nabla_x g^q(r, x))$,

$A_3 = -C_1 = -\nabla_r l^p(r,x)^T$, $A_4 = -D_1 = -\nabla_r h(r,x)^T$, $A_5 = -E_1 = -\nabla_r g^q(r,x)^T$, $B_3 = -C_2 = -\nabla_x l^p(r,x)^T$, $B_4 = -D_2 = -\nabla_x h(r,x)^T$, $B_5 = -E_2 = -\nabla_x g^q(r,x)^T$, $C_3 = -S_p = \begin{bmatrix} O_{p\times p} & O_{p\times(K-p)} \\ O_{(K-p)\times p} & I_{(K-p)\times(K-p)} \end{bmatrix}$, $E_5 = -S_q = \begin{bmatrix} O_{q\times q} & O_{q\times(K-q)} \\ O_{(K-q)\times q} & I_{(K-q)\times(K-q)} \end{bmatrix}$,

$C_4 = 0$, $C_5 = 0$, $D_3 = 0$, $D_4 = 0$, $D_5 = 0$, $E_3 = 0$ and $E_4 = 0$.

Since g, h and l are twice differentiable, by Schwarz's theorem, we have $\nabla_r\nabla_x g_i^p(x,y) = \nabla_x\nabla_r g_i^p(x,y), \forall i \in [1,p]$, $\nabla_r\nabla_x l_i^q(x,y) = \nabla_x\nabla_r l_i^q(x,y), \forall i \in [1,q]$ and $\nabla_r\nabla_x h(x,y) = \nabla_x\nabla_r h(x,y)$. It follows that $A_2 = B_1^T$ $\nabla\Phi(z)$ becomes

$$\nabla\Phi(z) = \begin{bmatrix} A_1 & B_1^T & A_3 & A_4 & A_5 \\ B_1 & B_2 & B_3 & B_4 & B_5 \\ -A_3 & -B_3 & -S_p & 0 & 0 \\ -A_4 & -B_4 & 0 & 0 & 0 \\ -A_5 & -B_5 & 0 & 0 & -S_q \end{bmatrix}.$$

It is easy to show that $-S_p$ and $-S_q$ are negative semidefinite. It follows that $S = \begin{bmatrix} -S_p & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -S_q \end{bmatrix}$ is negative semidefinite.

Since the function f is convex and twice differentiable, then $\nabla^2 f(x)$ is positive semidefinite. Additionally, g, h and l are biconvex and twice differentiable, then $\nabla_x^2 g_i^p(x,y)$, $\nabla_y^2 g_i^p(x,y)$, $i = 1,...q$; $\nabla_x^2 h(x,y)$,$\nabla_y^2 h(x,y)$ and $\nabla_x^2 l_i^p(x,y)$, $\nabla_y^2 g_i^l(x,y)$, $i = 1,...q$ are positive semidefinite. It follows that A_1 and B_2 are negative semidefinite and then $A = \begin{bmatrix} A_1 & B_1^T \\ B_1 & B_2 \end{bmatrix}$ is negative semidefinite.

Let $B = \begin{bmatrix} A_3 & A_4 & A_5 \\ B_3 & B_4 & B_5 \end{bmatrix}$, $\nabla\Phi(z)$ can be written as $\nabla\Phi(z) = \begin{bmatrix} A & B \\ -B^T & S \end{bmatrix}$. Since S and A are negative semidefinite, the conclusion follows. The proof where $\lambda + h(r,x) = 0$ follows along the same lines. □

Definition 2. A mapping $F : \mathcal{R}^n \longrightarrow \mathcal{R}^n$ is said to be monotonic if:

$$(x-y)^T(F(x)-F(y)) \geq 0, \qquad \forall x, y \in \mathcal{R}^n.$$

Lemma 4. [22]. A differentiable mapping $F : \mathcal{R}^n \longrightarrow \mathcal{R}^n$ is monotonic, if and only if the jacobian matrix $\nabla F(x)$, $\forall x \in \mathcal{R}^n$, is positive semidefinite.

We show in the following theorem that the neural network (12)-(16) is stable in the sense of Lyapunov and globally convergent.

Theorem 5. The neural network (12)-(16) is stable in the Lyapunov sense and converges to $(r^*, x^*, \mu^*, \lambda^*, \gamma^*)$, where (r^*, x^*) is a partial KKT point of problem (5).

Proof. Let $y^* = (r^*, x^*, \mu^*, \lambda^*, \gamma^*)$ an equilibrium point for the neural network (12)-(16). We define the following Lyapunov function,

$$V(y) = ||\Phi(y)||^2 + \frac{1}{2}||y-y^*||^2. \tag{17}$$

We have

$$\frac{dV(y(t))}{dt} = (\frac{d\Phi}{dt})^T\Phi + \Phi^T\frac{d\Phi}{dt} + (y-y^*)^T\frac{dy}{dt}. \tag{18}$$

Since $\frac{d\Phi}{dt} = \frac{\nabla\Phi}{\nabla y}\frac{dy}{dt} = \nabla\Phi(y)\Phi(y)$, then

$$\frac{dV(y(t))}{dt} = \Phi^T(\nabla\Phi(y)^T + \nabla\Phi(y))\Phi + (y-y^*)^T\Phi(y). \tag{19}$$

By Lemma 3., we have $\Phi^T(\nabla\Phi(y))\Phi \leq 0$ and $\Phi^T(\nabla\Phi(y)^T)\Phi \leq 0$.
Based on Theorem 2. and Lemma 4., we prove that $(y-y^*)^T(\Phi(y)) = (y-y^*)^T(\Phi(y)-\Phi(y^*)) \leq 0$.
We conclude that $\frac{dV(y(t))}{dt} \leq 0$ and then the neural network (12)-(16) is stable at y^* in the sense of Lyapunov.
Since $V(y) \geq \frac{1}{2}\left\|y-y^*\right\|^2$, there exists a convergent subsequence $(y(t)_{t\geq 0})$ such that $lim_{t\longrightarrow\infty}y(t) = \hat{y}$, and $\frac{dV(\hat{y}(t))}{dt} = 0$. Let $M = \{y(t)|\frac{dV(y(t))}{dt} = 0\}$, using LaSalle's invariance principle we can see that any solution starting from a certain y_0 converges to the largest invariant set contained in M.

Notice that $\begin{cases} \frac{dr}{dt} = 0 \\ \frac{dx}{dt} = 0 \\ \frac{d\mu}{dt} = 0 \\ \frac{d\lambda}{dt} = 0 \\ \frac{d\gamma}{dt} = 0 \end{cases} \Leftrightarrow \frac{dV(y)}{dt} = 0$. It follows that $\hat{y}$ is an equilibrium point of the neural network (12)-(16).

Now we consider a new Lyapunov function defined by $\hat{V}(y) = \|\Phi(y)\|^2 + \frac{1}{2}\|y - \hat{y}\|^2$. Since $\hat{V}$ is continuously differentiable, $\hat{V}(\hat{y}) = 0$ and $lim_{t\longrightarrow\infty} y(t) = \hat{y}$ then $lim_{t\longrightarrow\infty}\hat{V}(y(t)) = \hat{V}(\hat{y}) = 0$. Furthermore, since $\frac{1}{2}\|y - \hat{y}\|^2 \leq \hat{V}(y(t))$ then $lim_{t\longrightarrow\infty}\|y - \hat{y}\| = 0$ and $lim_{t\longrightarrow\infty} y(t) = \hat{y}$. We conclude that the neural network (12)-(16) is convergent in the sense of Lyupanov to an equilibrium point $\hat{y} = (\hat{r}, \hat{x}, \hat{\mu}, \hat{\lambda}, \hat{\gamma})$ where $(\hat{r}, \hat{x})$ is a partial KKT point of problem (5). $\square$

4. Numerical experiments

To evaluate the performances of our approach, we study in this section a transportation problem where the objective is to find the optimal shape of a transportation box subject to some geometric constraints. We generalise the three-dimension problem in order to evaluate the performances of the approach on large size instances.

We use *Python* to implement our dynamic neural network.The random instances are generated by *numpy.random*, the ODE systems are solved using *solve_ivp* of *scipy.integrate*, the deterministic equivalent programs are solved by the package *gekko* and the gradients and partial derivatives are computed using *autograd.grad* and *autograd.jacobian*.

We run our algorithms on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz.

The transportation problem consisits in shifting the grain from a warehouse to a factory in an open rectangular box as shown by Figure 2 of length x_1 meters, width x_2 meters and height x_3 meters. The aim is to maximize the volume $x_1x_2x_3$ subject to two constraints on the floor area and the wall area of the rectangular box in order to meet the shape of a given truck. The limits on the total wall area and the floor area are considered as random variables. We assume that these random variables are independent. We consider a stochastic

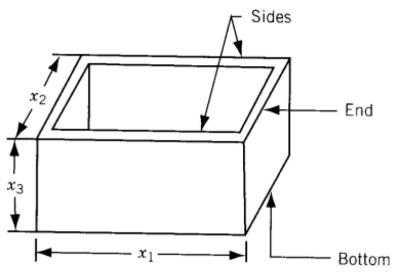

Figure 2: Shape of the box [23]

205 geometric transportation problem adapted from [24] given by:

$$\begin{aligned}
\min \quad & x_1^{-1}x_2^{-1}x_3^{-1}, \\
\text{s.t.} \quad & \mathcal{P}(\frac{1}{A_{wall}}(2x_3x_2 + 2x_1x_3) \leq 1, \frac{1}{A_{floor}}x_1x_2 \leq 1) \geq 1 - \epsilon. \\
& \alpha x_1^{-1}x_2 \leq 1, \\
& \beta x_2^{-1}x_2 \leq 1, \\
& \gamma x_2^{-1}x_3 \leq 1, \\
& \delta x_3^{-1}x_2 \leq 1,
\end{aligned} \tag{20}$$

To solve the joint probabilistic problem (20), we give the following equivalent deterministic problem:

$$\begin{aligned}
\min \quad & x_1^{-1}x_2^{-1}x_3^{-1}, \\
\text{s.t.} \quad & \mathcal{P}(\frac{1}{A_{wall}}(2x_3x_2 + 2x_1x_3) \leq 1) \geq y_1, \\
& \mathcal{P}(\frac{1}{A_{floor}}x_1x_2 \leq 1) \geq y_2, \\
& \alpha x_1^{-1}x_2 \leq 1, \\
& \beta x_2^{-1}x_2 \leq 1, \\
& \gamma x_2^{-1}x_3 \leq 1, \\
& \delta x_3^{-1}x_2 \leq 1, \\
& y_1y_2 \geq 1-\epsilon, \\
& 0 \leq y_1, y_2 \leq 1.
\end{aligned} \tag{21}$$

For the computations we set $\epsilon = 0.1$, $\frac{1}{A_{wall}} \sim \mathcal{N}(0.05, 0.01)$, $\frac{1}{A_{floor}} \sim \mathcal{N}(0.5, 0.01)$ and $\alpha = \beta = \gamma = \delta = 0.5$. In addition to problem (21), we solve a deterministic problem with the mean values of $\frac{1}{A_{wall}}$ and $\frac{1}{A_{floor}}$ as values of $\frac{1}{A_{wall}}$ and $\frac{1}{A_{floor}}$, respectively. We solve the problem obtained by replacing the joint constraints by individual ones. The results are recapitulated in Table 1. Column one gives the objective value obtained by the deterministic approach, column two gives the number of violated scenarios (VS) over 100 scenarios generated and column three the correspondent CPU time. The left columns represent the same information for the remaining approaches.

We observe that the deterministic problem gives the lowest value for the objective function. We notice that the number of violated scenarios (VS) in Figure 3 is equal to 100. The joint constraints approach ensures the best robustness compared to the two other problems as only two scenarios are violated, see Figure 5.

In order to evaluate the performances of our approach on large size instances, we introduce a more general formulation that is defined as follows,

Deterministic Approach			Individual constraints			Joint constraints		
Obj value	VS	CPU Time	Obj value	VS	CPU Time	Obj value	VS	CPU Time
0.125	100	0.01	0.131	8	0.06	0.132	2	8.75

Table 1: Results of the box shaping problem

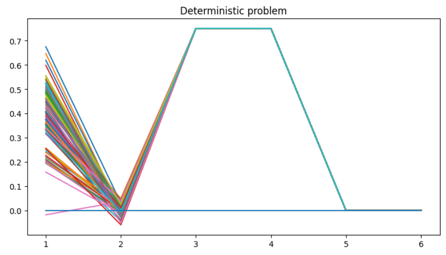

Figure 3: Out of 100 scenarios the constraints were violated 100 times

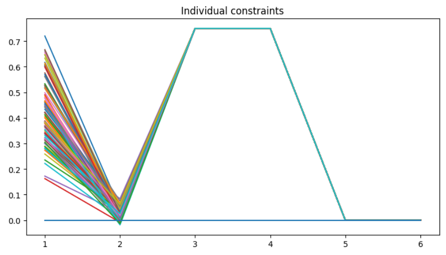

Figure 4: Out of 100 scenarios the constraints were violated 8 times

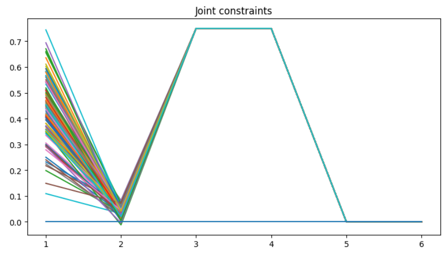

Figure 5: Out of 100 scenarios the constraints were violated 2 times

$$\begin{aligned} \min_{x\in\mathcal{R}_+^M} \quad & \prod_{i=1}^{m} x_i^{-1}, \\ \text{s.t.} \quad & \mathcal{P}(\Sigma_{j=1}^{m-1}(\frac{m-1}{A_{wall\,j}}x_1\prod_{i=1,i\neq j}^{m}x_i)\leq 1, \frac{1}{A_{floor}}\prod_{j=2}^{m}x_j\leq 1, \\ & \frac{1}{\gamma_{i,j}}x_ix_j^{-1}\leq 1, 1\leq i\neq j\leq M)\geq 1-\epsilon. \end{aligned} \tag{22}$$

As for the numerical experiments, we set $\epsilon = 0.05$, $\frac{1}{A_{floor}} \sim \mathcal{N}(1.0/20.0, 0.1)$,

$\frac{1}{A_{wall_j}}$ follows a normal distribution of mean randomly drawn from $[1.0/60.0, 1.0/40.0]$ and of standard deviation randomly drawn from $[0.3, 0.5]$ and $\gamma_{i,j} \sim \mathcal{N}(0.5, 0.1)$. The results for different values of m for problem (22) are given in Table 2, where Column one gives the data of the problem i.e (2, 4) means the problem is composed of 2 variables and 4 constraints. Columns two and three show the optimal value obtained by the expected value approach and the number of VS over 100 generated scenarios, respectively. Column four gives the associated CPU time. Columns five, six, seven and eight give the objective value of the individual constraints problem, the number of VS, the gap with the deterministic problem and the CPU Time, respectively. We note that the gap is computed compared to the deterministic approach and computed as follows gap = [abs ((objective value of the deterministic program - objective value of the individual program) / objective value of the deterministic program)].The last four columns give the optimal value obtained by the joint constraints problem, the number of VS, the gap with the deterministic approach and the CPU time. We observe that the problem with joint constraints remains the approach that covers well the risk region, the number of VS for joint problem is lower than those for the deterministic and the individual approaches i.e for $m = 6$ eleven scenarios were violated over 100 scenarios compared to 59 for the individual constraints and 100 for the deterministic program see Figures 6-9. We observe also that the joint approach is a time consuming method i.e for $m = 6$ the convergence takes 2113.29 seconds. In fact, the ODE solver is time consuming when the size of the problem increases.

5. conclusion

In this paper, we propose a dynamical neural network based on the partial KKT system to solve joint probabilistic geometric programs. The problem is first transformed into a deterministic program using the independence of the random variables. A biconvex equivalent is then given using a logarithmic transformation. Finally, a neural network is constructed based on the partial KKT

Data	Deterministic Approach			Individual constraints				Joint constraints			
	Obj value	VS	CPU Time	Obj value	VS	CPU Time	GAP	Obj value	VS	CPU Time	GAP
(2,4)	0.020	63	0.01	0.026	17	0.12	0.23	0.028	10	4.47	0.30
(3,8)	0.062	88	0.01	0.080	24	0.50	0.29	0.088	11	20.57	0.41
(4,14)	0.111	97	0.02	0.130	38	1.10	0.28	0.144	5	271.75	0.41
(5,22)	0.209	100	0.02	0.254	54	2.10	0.21	0.273	4	399.50	0.30
(6,32)	0.355	100	0.03	0.400	59	3.04	0.12	0.421	11	2113.29	0.18
(7,44)	0.520	100	0.03	0.607	59	6.51	0.16	0.648	12	11988.12	0.24

Table 2: Results of the generalised transportation problem

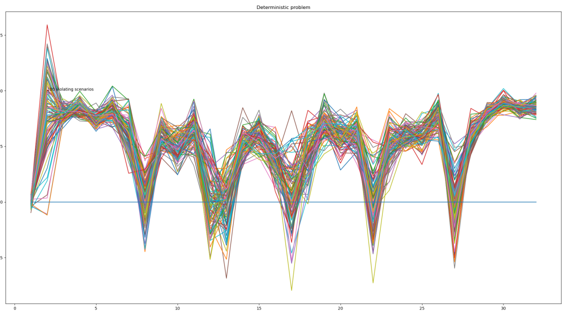

Figure 6: Out of 100 scenarios the constraints were violated 100 times

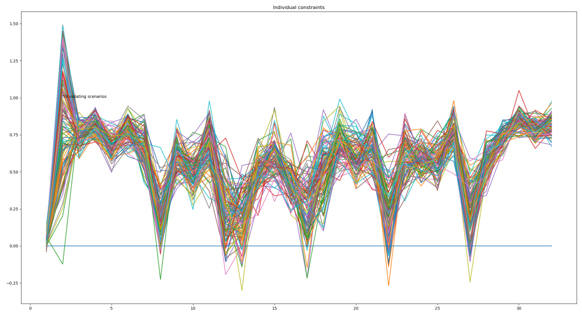

Figure 7: Out of 100 scenarios the constraints were violated 59 times

sytem of the obtained biconvex problem. We show the stability and the convergence of the proposed neural network.

To evaluate the performances of our dynamical neural network, we study a shape optimisation problem. The results show the robustness of the proposed approach. We use a standard ODE solver iteratively which leads to a significant CPU time. We believe that an ODE solver based on machine learning would decrease drastically the CPU time of our approach.

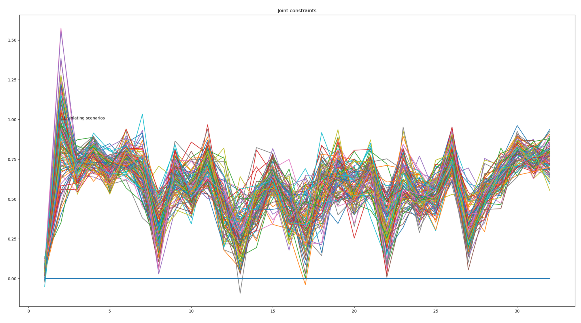

Figure 8: Out of 100 scenarios the constraints were violated 11 times

References

[1] D. D. Perlumutter, Geometric programming–theory and application, AIChE Journal 13 (4) (1967) 829–830. `doi:https://doi.org/10.1002/aic.690130408`.
URL `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690130408`

[2] M. Chiang, C. W. Tan, D. P. Palomar, D. O'neill, D. Julian, Power control by geometric programming, IEEE Transactions on Wireless Communications 6 (7) (2007) 2640–2651. `doi:10.1109/TWC.2007.05960`.

[3] Y. Li, Y.-C. Chen, Optimal channel doping profile of two-dimensional metal-oxide-semiconductor field-effect transistors via geometric programming, Journal of Advanced Simulation in Science and Engineering 2 (1) (2015) 178–200. `doi:10.15748/jasse.2.178`.

[4] W. Hoburg, P. Abbeel, Geometric programming for aircraft design optimization, Vol. 52, 2012. `doi:10.2514/6.2012-1680`.

[5] J. P. Vanderhaegen, R. W. Brodersen, Automated design of operational transconductance amplifiers using reversed geometric programming, in:

Proceedings of the 41st Annual Design Automation Conference, DAC '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 133–138. `doi:10.1145/996566.996608`.
URL `https://doi.org/10.1145/996566.996608`

[6] J. Dupacová, Stochastic geometric programming with an application, Kybernetika 46 (2010) 374–386.

[7] J. Singh, Z.-Q. Luo, S. S. Sapatnekar, A geometric programming-based worst case gate sizing method incorporating spatial correlation, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27 (2) (2008) 295–308. `doi:10.1109/TCAD.2007.913391`.

[8] V. Kojić, Z. Lukač, Solving profit maximization problem in case of the cobb-douglas production function via weighted ag inequality and geometric programming, in: 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2018, pp. 1900–1903. `doi:10.1109/IEEM.2018.8607446`.

[9] C.-S. Liu, G. Xu, , L. Wang, An improved geometric programming approach for optimization of biochemical systems, Journal of Applied Mathematics 2014 (2014). `doi:10.1155/2014/719496`.

[10] Y. Li, A. Duan, A. Gratner, L. Feng, A geometric programming approach to the optimization of mechatronic systems in early design stages, in: 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2016, pp. 1351–1656. `doi:10.1109/AIM.2016.7576958`.

[11] X. Wang, Q. Su, Y. Miao, A differential evolution algorithm for solving geometric programming problems, in: 2013 Ninth International Conference on Natural Computation (ICNC), 2013, pp. 359–363. `doi:10.1109/ICNC.2013.6818001`.

[12] J. Liu, A. Lisser, Z. Chen, Stochastic geometric optimization with joint

probabilistic constraints, Operations Research Letters 44 (5) (2016) 687–691. doi:https://doi.org/10.1016/j.orl.2016.08.002.

[13] C. Mung, S. Boyd, Geometric programming duals of channel capacity and rate distortion, IEEE Transactions on Information Theory 50 (2) (2004) 245–258. doi:10.1109/TIT.2003.822581.

[14] L. Sela Perelman, S. Amin, Control of tree water networks: A geometric programming approach, Water Resources Research 51 (10) (2015) 8409–8430. arXiv:https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2014WR016756, doi:https://doi.org/10.1002/2014WR016756. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014WR016756

[15] K.-L. Hsiung, S.-J. Kim, S. Boyd, Tractable approximate robust geometric programming, Optimization and Engineering 9 (2008) 95–118. doi:10.1007/s11081-007-9025-z.

[16] R. Khanjani-Shiraz, S. Khodayifar, P. M. Pardalos, Copula theory approach to stochastic geometric programming, Journal of Global Optimization 81 (2021) 435–468. doi:10.1007/s10898-021-01062-7.

[17] J. Cheng, A. Lisser, A second-order cone programming approach for linear programs with joint probabilistic constraints, Operations Research Letters 40 (5) (2012) 325–328. doi:https://doi.org/10.1016/j.orl.2012.06.008.
URL https://www.sciencedirect.com/science/article/pii/S0167637712000831

[18] G. Villarrubia, J. F. De Paz, P. Chamoso, F. D. la Prieta, Artificial neural networks used in optimization problems, Neurocomputing 272 (2018) 10–16. doi:https://doi.org/10.1016/j.neucom.2017.04.075.
URL https://www.sciencedirect.com/science/article/pii/S0925231217311116

[19] A. Nazemi, N. Tahmasbi, A high performance neural network model for solving chance constrained optimization problems 121 (2013) 540–550. doi:10.1016/j.neucom.2013.05.034.

[20] M. Jiang, Z. Meng, R. Shen, Partial exactness for the penalty function of biconvex programming, Entropy 23 (2) (2021). doi:10.3390/e23020132.

[21] S. Tassouli, A. Lisser, A neural network approach to solve linear programs with joint probabilistic constraints, working paper or preprint (Jul. 2021). URL https://hal.archives-ouvertes.fr/hal-03281954

[22] R. Tyrrell, J. Roger, B. Wets, Variational Analysis, Springer, Berlin, Heidelberg, 1998. doi:10.1007/978-3-642-02431-3.

[23] Geometric Programming, John Wiley and Sons, Ltd, 2009, Ch. 8, pp. 492–543. doi:https://doi.org/10.1002/9780470549124.ch8.

[24] A. Lisser, J. Liu, S. Peng, Rectangular chance constrained geometric optimization, Optimization and Engineering 21 (2020) 1573–2924. doi:https://doi.org/10.1007/s11081-019-09460-3.