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

Siham Tassouli, Abdel Lisser

A neural network approach to solve linear 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

This paper studies a dynamical neural network approach for solving linear programs with joint Probabilistic Constraints (LPPC) with normally distributed random variables and independent matrix row vectors. We show that the proposed neural network is stable in the sense of Lyapunov and globally convergent. Finally, numerical results are given using randomly generated data.

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

2010 MSC: 00-01, 99-00

*Corresponding author.

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

1. Introduction

In the paper, we study the following stochastic linear programming problem:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \mathcal{P}(Tx \leq D) \geq 1 - \alpha \\ & x \in X \end{aligned} \tag{1}$$

Where $X \subset \mathcal{R}^n_+$ is a polyhedron, $c \in \mathcal{R}^n$, $D = (D_1, ..., D_k) \in \mathcal{R}^K$, $T = [T_1, ...T_K]^T$ is a $K \times n$ random vector in $\mathcal{R}^n$ and $0 < \alpha < 1$ is a specified confidence parameter.

Chance constrained programming with joint chance constraints was first introduced in 1965 by *Miler and Wagner* [1]. They propose a deterministic equivalent of the joint constrained problem and show its concavity. *Prékopa* [2] shows that in the case of the multivariate normal probability distribution with right hand side random variable, the constraints set is convex.

Luedtke and Ahmed (2008) [3] propose an approximate solution approach to solve a joint probabilistic linear program where the only random variable is the confidence level parameter. The proposed method relies on Monte Carlo sampling of the random variables. To solve the same problem, *Luedtke and Nemhauser* (2010) [4] study an integer programming approach. *Adam et al.* (2020) [5] introduce a discrete regularised approach for solving stochastic programs with joint chance constraints with discrete random distribution.

Chen et al. [6] propose an approach based on a classical worst case bound to approximate the joint constraints even in the case where the constraints are correlated. *Zymler and Rustem* [7] use Worst-Case Conditional Value-

at-Risk to approximate both individual and joint probabilistic constraints. D.Reich [8] develops two linear programming based heuristic methods, greedy and dual heuristics, for solving linear programs with joint probabilistic constraints, where the random variable is the right-hand side vector. *Ackooij and Henrion* [9] propose joint probabilistic models to deal with hydraulic valley optimization. *González et al.* [10] propose an optimization problem with a joint probabilistic constraint over an infinite system of random inequalities to assist gas network operators in managing uncertainty. *Hyunhee and Eheart* [11] present a screening technique to solve joint chance constrained programs for air quality managment problems. To solve power management problems, *Arnold et al.* [12] model the demand of the generation of the wind energy by a joint probabilistic constraint. *Cheng and Lisser* [13] propose two piecewise approximation approaches to give an upper and a lower bounds for the exact optimal solution of problem (1).

Ordinary Differential Equations (ODE) systems and machine learning techniques have been used for solving optimization problems. *Arrow et al.* [14] propose ODE methods to solve equality constrained optimization problems. *Jin et al.* [15] introduce a differential equation approach to solve nonlinear programming problems.

Adam and Branda [16] study an approach based on the stochastic gradient descent method to solve chance constrained problems with discrete random distribution. *Zhao and You* [17] use unsupervised learning with generative adversarial network in data generation and sampling for chance constrained problems.

Nazemi et al. [18] introduce a dynamical neural network for solving indi-

vidual chance constrained optimization problems. Earlier, He [19] proposed a neural network for solving the minimax problem. *Nazmi et al.* [20] solved geometric programming problem using the same approach and used this dynamical neural network to solve the maximum flow problem [21] and the shortest path problem [22].

This paper is organised as follows. In Section 2 we study the partial KKT system of an equivalent deterministic problem of (1). In Section 3 a neural network approach is proposed. Section 4 discusses the stability and the convergence of the proposed neural network. Finally, numerical results are given in Section 5.

2. Deterministic reformulation

We consider the special case where $T_k,\ k = 1, ..., K$ are multivariate normally distributed independent row vectors with known mean vector $\mu_k = (\mu_{k1}, ..., \mu_{kn})$ and covariance matrix Σ_k. Problem (1) can be written as follows:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \prod_{k=1}^{K} \mathcal{P}(T_k x \leq D_k) \geq \prod_{k=1}^{K} (1-\alpha)^{y_k} \\ & \sum_{k=1}^{K} y_k = 1 \qquad (2) \\ & y_k \geq 0, k = 1, ..., K \end{aligned}$$

$$(3)$$

By the independence of the vectors $T_k, k = 1, .., K$ an equivalent nonlinear

program for problem (1) is given by [13, 23]:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \mu_k^T x + F^{-1}(p^{y_k})||\Sigma_k^{\frac{1}{2}} x|| \leq D_k, k = 1, ..., K \\ & \sum_{k=1}^{K} y_k = 1 \\ & y_k \geq 0, k = 1, ..., K \\ & x \in X \end{aligned} \tag{4}$$

In the the rest of the paper, we consider the following deterministic equiv-
65 alent problem:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \mu_k^T x + F^{-1}(p^{y_k})||\Sigma_k^{\frac{1}{2}} x|| - D_k \leq 0, k = 1, ..., K \\ & \sum_{k=1}^{K} y_k - 1 \leq 0 \\ & 1 - \sum_{k=1}^{K} y_k \leq 0 \\ & -y_k \leq 0, k = 1, ..., K \\ & -x \leq 0 \end{aligned} \tag{5}$$

Notice that problem (5) is nonlinear and a biconvex problem. We introduce the biconvex function g defined as follows:

$$g(x,y) = \begin{pmatrix} g_1(x,y) = \mu_1^T x + F^{-1}(p^{y_1})\sqrt{x^T \Sigma_1 x} - D_1 \\ \vdots \\ g_k(x,y) = \mu_k^T x + F^{-1}(p^{y_k})\sqrt{x^T \Sigma_k x} - D_k \\ g_{k+1}(x,y) = \sum_{k=1}^{K} y_k - 1 \\ g_{k+2}(x,y) = 1 - \sum_{k=1}^{K} y_k \\ g_{k+3}(x,y) = -y_1 \\ \vdots \\ g_{2k+2}(x,y) = -y_k \\ g_{2k+3}(x,y) = -x_1 \\ \vdots \\ g_{2k+n+2}(x,y) = -x_n \end{pmatrix}$$

Problem (4) becomes :

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & g(x,y) \leq 0 \end{aligned} \tag{6}$$

Let $(x^*, y^*) \in \mathcal{R}^n \times \mathcal{R}^k$, if there exists $u_i^{(1)}, u_i^{(2)}, i = 1, .., 2K+n+2$ such that:

$$\nabla f(x^*) + \sum_{i=1}^{2K+n+1} u_i^{(1)} \nabla_x g_i(x^*, y^*) = 0 \tag{7}$$

$$\sum_{i=1}^{2K+n+1} u_i^{(2)} \nabla_y g_i(x^*, y^*) = 0 \tag{8}$$

$$u_i^{(1)} g_i(x^*, y^*) = 0, u_i^{(2)} \geq 0, i = 1, .., 2K+n+1 \tag{9}$$

$$u_i^{(2)} g_i(x^*, y^*) = 0, u_i^{(1)} \geq 0, i = 1, .., 2K+n+1 \tag{10}$$

then (x^*, y^*) is a partial KKT point of (5) [24].

Definition 1. We consider the following problem:

$$\begin{aligned} \min \quad & f(x_1, x_2) \\ \text{s.t.} \quad & g(x_1, x_2) \leq 0 \qquad (P1) \end{aligned}$$

Where $f, g : \mathcal{R}^{n_1} \times \mathcal{R}^{n_2} \longrightarrow \mathcal{R}$ are differentiable and biconvex. We denote: $X(x_1) = \{x_2 \in \mathcal{R}^{n_2} | g(x_1, x_2) \leq 0\}$ and $X(x_2) = \{x_1 \in \mathcal{R}^{n_1} | g(x_1, x_2) \leq 0\}$ (x_1^*, x_2^*) is a partial optimum of (P1) if :

$$f(x_1^*, x_2^*) \leq f(x_1, x_2^*), \forall x_1 \in X(x_2^*)$$

$$f(x_1^*, x_2^*) \leq f(x_1^*, x_2), \forall x_2 \in X(x_1^*)$$

Definition 2. [24] Let $(x^*, y^*) \in \mathcal{R}^n \times \mathcal{R}^k$, the constraint of (5) is called a partial Slater constraint qualification at (x^*, y^*), if there exists $(\tilde{x}, \tilde{y}) \in \mathcal{R}^n \times \mathcal{R}^k$ such that:

$$g_i(x^*, \tilde{y}) < 0, g_i(\tilde{x}, y^*) < 0, i = 1, .., 2K + n + 2$$

Theorem 1. [24] Let $(x^*, y^*) \in \mathcal{R}^n \times \mathcal{R}^k$. if (x^*, y^*) is a feasible solution for problem (5) with respect to partial slater constraints qualification, then (x^*, y^*) is a partial optimum of (5) if and only if (x^*, y^*) is a partial KKT point of (5) .

Theorem 2. [24] Let $(x^*, y^*) \in \mathcal{R}^n \times \mathcal{R}^k$ be a partial solution of (5), with respect to partial slater constraints qualification at (x^*, y^*), then (x^*, y^*) is a partial KKT point of (5) if and only if (6), (7) and (8) hold with $u_i^{(1)} = u_i^{(2)}, i = 1, ..., 2K + n + 1$.

In the the following, we write the partial KKT System of (5) as follows: $(x^*, y^*) \in \mathcal{R}^n \times \mathcal{R}^k$ is a partial optimum of (5) if and only if there exists $u_i^*, i = 1, .., 2K + n + 1$ such that:

$$\nabla f(x^*) + \sum_{i=1}^{2K+n+1} u_i^* \nabla_x g_i(x^*, y^*) = 0 \tag{11}$$

$$\sum_{i=1}^{2K+n+1} u_i^* \nabla_y g_i(x^*, y^*) = 0 \tag{12}$$

$$u_i^* g_i(x^*, y^*) = 0, u_i^* \geq 0, i = 1, .., 2K + n + 1 \tag{13}$$

3. Dynamical neural network approach

In this section, we propose a recurrent neural network model for solving problem (1). The dynamical equation of the neural network is given by:

$$\frac{dx}{dt} = -(\nabla f(x) + \nabla_x g(x, y)^T (u + g(x, y))_+) \tag{14}$$

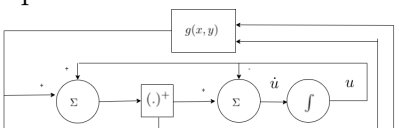
$$\frac{dy}{dt} = -(\nabla_y g(x, y)^T (u + g(x, y))_+) \tag{15}$$

$$\frac{du}{dt} = (u + g(x, y))_+ - u \tag{16}$$

We denote $z = (x, y, u)$ and define $\Phi(z) = \begin{bmatrix} -(\nabla f(x) + \nabla_x g(x,y)^T (u + g(x,y))_+) \\ -(\nabla_y g(x,y)^T (u + g(x,y))_+) \\ (u + g(x,y))_+ - u \end{bmatrix}$

We can rewrite the neural network defined in (13)-(15) as :

$$\begin{cases} \frac{dz}{dt} & = & \kappa \Phi_(z) \\ z(t_0) & = & z_0 \end{cases} \tag{17}$$

where κ is a scale parameter and indicates the convergence rate of the neural network (13)-(15). For the sake of simplicity we take $\kappa = 1$. The architectural construction of the neural network is detailed in Figure 1. Each line of the block implements an equation of the neural network (11)-(12). We take the first line for example : 

. From left side to right side, we have $g(x, y)$ and u as inputs of the operator sum. As output we have $g(x, y)+u$ which is an input of the operator $()_+$ which results in $(g(x, y)+u)_+$. Later, the inputs of the second sum operator are $(g(x, y)+u)_+$ and $-u$. In the output we have the expression cited in equation (14). Finally u is obtained using an integration operator.

3.1. Convergence and stability of the neural network

In this section we study the convergence and the stability of the proposed neural network (15).

Theorem 3. Let (x^*, y^*, u^*) an equilibrium point of the neural network defined by (13)-(15), then (x^*, y^*) is a partial KKT point of (5). On the other

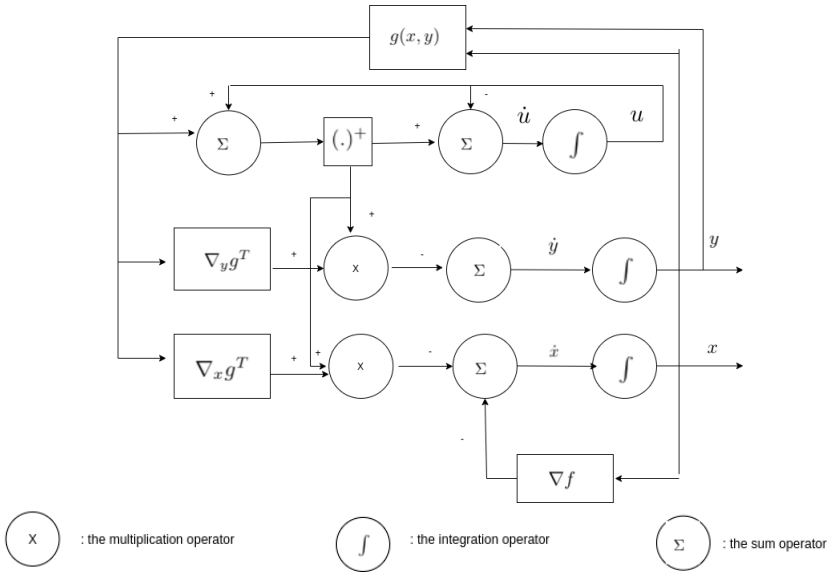


Figure 1: A diagram by block of the neural network (5), (6) and (7)

hand, if $(x^*, y^*) \in \mathcal{R}^n \times \mathcal{R}^k$ is a partial KKT point of (5), then there exists $u^* \geq 0$ such that (x^*, y^*, u^*) is an equilibrium point of the Neural Network (13)-(15).

Proof. Let (x^*, y^*, u^*) be an equilibrium point of (13)-(15), then: $\frac{dx^*}{dt} = 0$, $\frac{dy^*}{dt} = 0$ and $\frac{du^*}{dt} = 0$ and we have that:

$$\begin{cases} \nabla f(x^*) + \nabla_x g(x^*, y^*)^T (u^* + g(x^*, y^*))_+ = 0 \\ \nabla_y g(x^*, y^*)^T (u^* + g(x^*, y^*))_+ = 0 \\ (u^* + g(x^*, y^*))_+ - u^* = 0 \end{cases}$$

We also have $(u^* + g(x^* + y^*))_+ = u^*$ *if and only if* $u^* \geq 0$, $g(x^*, y^*) \leq 0$ and ${u^*}^T g(x^*, y^*) = 0$

Furthermore, by substitution we have : $\nabla f(x^*) + \nabla_x g(x^*, y^*)^T u^* = 0$ and $\nabla_y g(x^*, y^*)^T u^* = 0$. Therefore, (x^*, y^*) is a partial KKT point of (5).

Now let (x^*, y^*) be a partial KKT point of (5), by the system (10)-(12) there exists u^* such that:

$$\begin{cases} \nabla_x f(x^*) + \nabla_x g(x^*, y^*)^T u^* = 0 \\ \nabla_y g(x^*, y^*)^T u^* = 0 \\ u^* \geq 0, u^{*T} g(x^*, y^*) = 0 \end{cases}$$

It is obvious that (x^*, y^*, u^*) is an equilibrium point for the neural network (13)-(15). □

Theorem 4. For any initial point $(x(t_0), y(t_0), u(t_0))$, there exists an unique continuous solution $(x(t), y(t), u(t))$ for (13)-(15).

Proof. Since $\nabla f(x)$, $\nabla_x g(x, y)$ and $\nabla_y g(x, y)$ are continuously differentiable on open sets, then $-(\nabla f(x) + \nabla_x g(x, y)^T (u + g(x, y))_+)$, $-(\nabla_y g(x, y)^T (u + g(x, y))_+)$ and $((u + g(x, y))_+ - u)$ are locally Lipschitz continuous. According to the local existence of ordinary differential equations also known as *Picard–Lindelöf Theorem* [25], the neural network (13)-(15) has a unique continuous solution $(x(t), y(t), u(t))$ [18]. □

Before proving the stability and the convergence of the proposed neural network, we show that the matrix $\nabla\Phi$ is negative semidefinite.

Theorem 5. The jacobian matrix $\nabla\Phi(z)$ defined in (4.1) is negative semidefinite matrix.

Proof. We consider the dynamical neural network (13)-(15).

We assume that there exist $0 < p < n + 2k + 2$ such that

$(u + g)_+ = (u_1 + g_1(x, y), u_2 + g_2(x, y), ..., u_p + g_p(x, y), \underbrace{0, ..., 0}_{n+2k+2-p})$

The jacobian matrix of Φ is given by:

$$\nabla\Phi(z) = \begin{bmatrix} A & B & -\nabla_x g^p(x,y)^T \\ C & D & -\nabla_y g^p(x,y)^T \\ \nabla_x g^p(x,y) & \nabla_y g^p(x,y) & -S_p \end{bmatrix}$$

Where:

$A = -(\nabla^2 f(x) + \sum_{i=1}^{p}((u_i + g_i)\nabla_x^2 g_i^p(x,y)) + \nabla_x g^p(x,y)^T \nabla_x g^p(x,y))$

$B = -(\sum_{i=1}^{p}((u_i + g_i)\nabla_y \nabla_x g_i^p(x,y)) + \nabla_x g^p(x,y)^T \nabla_y g^p(x,y))$

$C = -(\sum_{i=1}^{p}((u_i + g_i)\nabla_x \nabla_y g_i^p(x,y)) + \nabla_y g^p(x,y)^T \nabla_x g^p(x,y))$

$D = -(\sum_{i=1}^{p}((u_i + g_i)\nabla_y^2 g_i^p(x,y)) + \nabla_y g^p(x,y)^T \nabla_y g^p(x,y))$

$$S_p = \begin{bmatrix} O_{p\times p} & O_{p\times(n+2k+2-p)} \\ O_{(n+2k+2-p)\times p} & I_{(n+2k+2-p)\times(n+2k+2-p)} \end{bmatrix}$$

$$\text{and } \nabla_x g^p(x,y) = \begin{bmatrix} \frac{\nabla g_1}{\nabla_{x1}}(x,y) & \dots & \frac{\nabla g_1}{\nabla_{xn}}(x,y) \\ \vdots & \ddots & \vdots \\ \frac{\nabla g_p}{\nabla_{x1}}(x,y) & \dots & \frac{\nabla g_p}{\nabla_{xn}}(x,y) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

Since g is twice differentiable, by Schwarz's theorem, we have $\nabla_y \nabla_x g_i^p(x,y) = \nabla_x \nabla_y g_i^p(x,y)$, $\forall i \in [1,p]$. Therefore $B = C^T$

$\nabla\Phi(z)$ can be written as follows:

$$\nabla\Phi(z) = \begin{bmatrix} A & C^T & -\nabla_x g^p(x,y)^T \\ C & D & -\nabla_y g^p(x,y)^T \\ \nabla_x g^p(x,y) & \nabla_y g^p(x,y) & S_p \end{bmatrix}$$

We first proof that $M = \begin{bmatrix} A & C^T \\ C & D \end{bmatrix}$ is negative semidefinite matrix.

It's easy to see that the matrices $\nabla_x g^p(x,y)^T \nabla_x g^p(x,y)$ and $\nabla_y g^p(x,y)^T \nabla_y g^p(x,y)$ are positive semidefinite. Since the function f is convex and twice differentiable and g is biconvex and twice differentiable [26], then matrices $\nabla^2 f(x)$, $\nabla_x^2 g_i^p(x,y)$ and $\nabla_y^2 g_i^p(x,y)$, $i = 1, ...p$ are positive semidefinite matrices.
We conclude that A and D are negative semidefinite matrices, and M is a negative semidefinite matrix [27].
Then, we have

$$\nabla\Phi(z) = \begin{bmatrix} M & -Q^T \\ Q & -S_p \end{bmatrix} \text{ when } Q = \begin{bmatrix} \nabla_x g^p(x,y) \\ \nabla_y g^p(x,y) \end{bmatrix}$$

$-S_p$ is negative semidefinite and M is negative semidefinte, then $\nabla\Phi(z)$ is negative semidefinite. □

To study the stability and the convergence of the dynamical network, we introduce a monotonic map defined as follows

Definition 3. 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 6. 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 [28].

Theorem 7. The neural network (13)-(15) is globally stable in the Lyapunov sense and is globally convergent to (x^*, y^*, u^*), where (x^*, y^*) is a partial KKT point of problem (5).

Proof. We consider the following Lyapunov function:

$$E(z) = \|\Phi(z)\|^2 + \frac{1}{2}\|z - z^*\|^2 \tag{18}$$

we have $\frac{d\Phi}{dt} = \frac{\nabla\Phi}{\nabla z}\frac{dz}{dt} = \nabla\Phi(z)\Phi(z)$, then

$$\frac{dE(z(t))}{dt} = (\frac{d\Phi}{dt})^T\Phi + \Phi^T\frac{d\Phi}{dt} + (z - z^*)^T\frac{dz}{dt}$$

$$= \Phi^T(\nabla\Phi(z)^T + \nabla\Phi(z))\Phi + (z - z^*)^T\Phi(z)$$

By **Theorem 5**, we have $\Phi^T(\nabla\Phi(z)^T + \nabla\Phi(z))\Phi \leq 0$. By **Theorem 5**

and **Lemma 6**, we have $(z - z^*)^T \Phi(z) = (z - z^*)^T (\Phi(z) - \Phi(z^*)) \leq 0$ then:

$$\frac{dE(z(t))}{dt} \leq 0 \tag{19}$$

$E(z)$ is positive and $\frac{dE(z(t))}{dt} \leq 0$ then the neural network (13)-(15) is globally stable in the sense of Lyapunov [29].

As $E(z) \geq \frac{1}{2} \|z - z^*\|^2$, then there exists a convergent subsequence $\{z(t_k) | t_0 < t_1 < .. < t_k < t_{k+1}\}$ where $t_k \longrightarrow \infty$ when $k \longrightarrow \infty$, such that $lim_{k \longrightarrow \infty} z(t_k) = \hat{z}$, where $\hat{z}$ satisfies $\frac{dE(z(t))}{dt} = 0$.

Notice that $\hat{z}$ is a w-limit point of $\{z(t)\}_{t \geq t_0}$. By *LaSalle's invariant set theorem* [30], there exists a certain L such that $z(t) \longrightarrow L$ when $t \longrightarrow \infty$. From (10)-(12) and (18), it follows that :

$$\begin{cases} \frac{dx}{dt} = 0 \\ \frac{dy}{dt} = 0 \quad \Leftrightarrow \frac{dE(z)}{dt} = 0 \\ \frac{du}{dt} = 0 \end{cases}$$

Therefore, $\hat{z}$ is an equilibrium point for the neural network (13)-(15).

We define now a new Lyapunov function:

$$\hat{E}(z) = \|\Phi(z)\|^2 + \frac{1}{2} \|z - \hat{z}\|^2 \tag{20}$$

$\hat{E}(z)$ is continuously differentiable, $\hat{E}(\hat{z}) = 0$ and $lim_{k \longrightarrow \infty} z(t_k) = \hat{z}$ then $lim_{k \longrightarrow \infty} \hat{E}(z(t_k)) = \hat{E}(\hat{z})$. We have also $\frac{d\hat{E}(z)}{dt} \leq 0$, then $\frac{1}{2} \|z - \hat{z}\|^2 \leq \hat{E}(z(t))$. Hence, $lim_{t \longrightarrow \infty} \|z - \hat{z}\| = 0$ and $lim_{t \longrightarrow \infty} z(t) = \hat{z}$. Therefore, the neural network (13)-(15) is globally convergent in the sense of Lyupanov to an equilibrium point $\hat{z} = (\hat{x}, \hat{y}, \hat{u})$ where $(\hat{x}, \hat{y})$ is a partial KKT point of

problem (5) [18].

□

4. Numerical Study

We first solve the problem cited in [18], using a joint probabilistic approach. The original problem is defined as follows

$$\begin{aligned} \max \quad & 50x_1 + 100x_2 \\ \text{s.t.} \quad & \begin{cases} P(a_{11}x_1 + a_{12}x_2 \leq 2500) \geq 0.99 \\ P(a_{21}x_1 + a_{22}x_2 \leq 2000) \geq 0.99 \\ P(a_{31}x_1 + a_{32}x_2 \leq 450) \geq 0.99 \\ x_1, x_2 \geq 0 \end{cases} \end{aligned} \tag{21}$$

We solve the following joint probabilstic problem using the neural network (13)-(15)

$$\begin{aligned} \max \quad & 50x_1 + 100x_2 \\ \text{s.t.} \quad & \begin{cases} P(a_{11}x_1 + a_{12}x_2 \leq 2500, a_{21}x_1 + a_{22}x_2 \leq 2000, \\ \qquad a_{31}x_1 + a_{32}x_2 \leq 450) \geq 0.99 \\ x_1, x_2 \geq 0 \end{cases} \end{aligned} \tag{22}$$

A.Nazemi et al. show that an optimal solution for problem (20) is equal to 6199.99 with $x^* = (42.07, 40.96)$. Using our approach we find 6199.38 with $x^* = (42.11, 40.93)$.

The following table recapitulates the results obtained using the two approaches using different values for the confidence level.

The confidence level	Individual constraints	Joint constraints
0.05	8006.59	7955.12
0.10	9480.90	9212.95
0.15	10828.02	10168.41
0.20	12209.51	11007.80

Table 1: Individual vs joint constraints

From table 1, we can see that the joint chance constrained model is more conservative than the individual chance constrained model which is consistent with the probabilistic constraint theoretical results are larger than the ones obtained using the joint constraints. Although, using the joint constraints enables a better covering for the risk zone.

Now to evaluate properly the quality and the robustness of our neural network (13)-(15), we compare the results obtained by the neural network on various randomly generated data for several instance sizes with the results obtained using individual and joint chance constraints and the expected value approach which consists in replacing the random variables $T_k, k = 1, .., K$ by their respective mean values.

We generate randomly different instances of problem (5) with $\alpha = 0.05$. Table 2 shows our numerical experiments where column one gives the size of the problem, i.e., the number of variables and the number of constraints. Columns two and three show the optimal value obtained by the expected value approach and the corresponding CPU time, respectively. Columns four, five and six present the optimal value obtained by individual chance constraints, the relative CPU time and the gap with the expected value

approach. The gap is defined by the following formula: (objective value of the first method - objective value of the second method) / objective value of the first method * 100. The last three columns give the optimal value obtained by the neural network, the corresponding CPU time and the gap with the expected value approach.

We implement our algorithms in python. We use the function *uniform* of the package *numpy.random* to generate the random instances of problem (5). For our numerical expriments we choose the values of the mean vectors randomly in the interval $[10, 15]$, the values of the standard deviation vectors in $[1, 4]$, the values of the vector β in $[600, 700]$, and the values of the vector c in $[55, 65]$. We use *Gekko*, a Python package for machine learning and optimization, to solve the deterministic problem. To solve the ODE system of the neural network we use the *solve_ivp* function of the *scipy.integrate* library, using the backward differentiation formula as an integration method. The stopping criteria for our neural network is $e = 10^{-6}$. The gradient and the partial derivatives in the the matrix $\Phi(z)$ of the neural network are computed by the functions *grad* and *jacobian* of the package *autograd*. We run our algorithms on Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz.

Data	Expected value approach		Individual constraints			Joint constraints		
	Obj value	CPU Time	Obj value	CPU Time	GAP	Obj value	CPU Time	GAP
(3,2)	3500.00	0.01	4358.30	0.06	24 %	4542.46	10.72	29%
(5,3)	3230.76	0.01	3949.64	0.26	22 %	4036.69	22.91	25%
(7,5)	3150.00	0.01	3731.69	0.56	18 %	3859.12	43.61	22%
(10,5)	3292.60	0.02	3779.02	0.75	15 %	3893.12	41.40	18%
(20,10)	3187.05	0.02	3573.50	5.84	12 %	3701.69	110.75	16%
(30,20)	3224.22	0.02	3563.18	30.66	10%	3682.52	2454.31	14 %

Table 2: Computational results with different sizes for problem (4)

Data	Expected value approach	Individual constraints	Joint constraints
(3,2)	80	8	4
(5,3)	75	15	3
(7,5)	95	25	5
(10,5)	94	16	5
(20,10)	100	26	4
(30,20)	100	34	2

Table 3: Number of violated scenarios

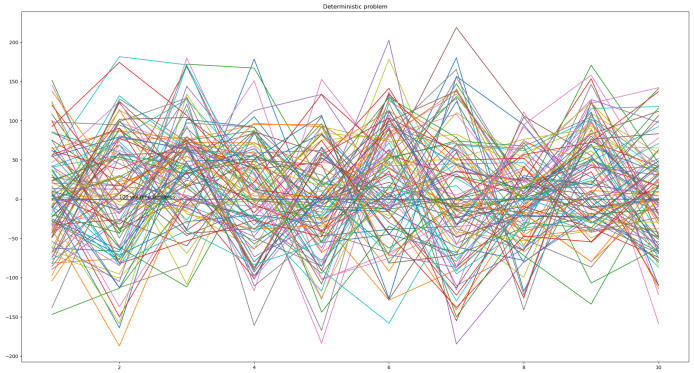


Figure 2: Out of 100 scenarios the constraints were violated 100 times

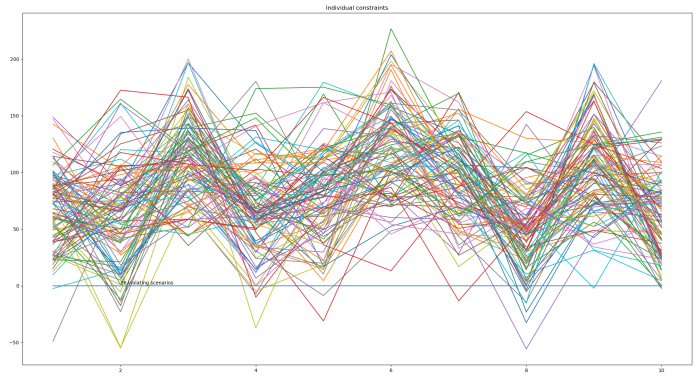


Figure 3: Out of 100 scenarios the constraints were violated 26 times

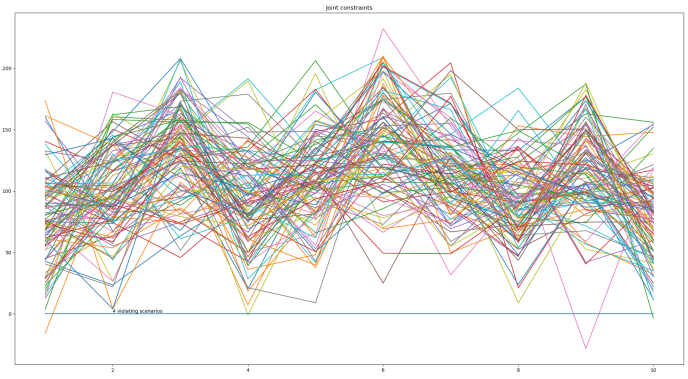


Figure 4: Out of 100 scenarios the constraints were violated 4 times

Table 2 shows that the solution obtained using the expected value approach is the lowest compared to the approaches involving individual or joint chance constraints. We remark that the gaps between the objective function

values obtained decrease as the problem becomes large. Even if the approach using joint constraints is more conservative, it covers better the risk region.

We then generate 100 scenarios for each problem of Table 2 with 10 independent normal vectors T_k, $k = 1, .., 10$ of mean μ_k and covariance matrix Σ_k. Those vectors are generated using the function *normal* of the package *numpy.random*. We note how many times the constraints were violated for the two approaches. The results are given in Table 3.

We remark that the number of violated scenarios is the largest in the case of the expected value approach and the lowest for the approach with joint constraints.

Now, we check that the solution obtained by the neural network is partial KKT feasible. Figure 5 shows that the increase of the number of iterations leads to more tight partial KKT feasibility. Furthermore, we remark that the convergence towards the partial KKT solution becomes faster in terms of the number of iterations as the problem size increases.

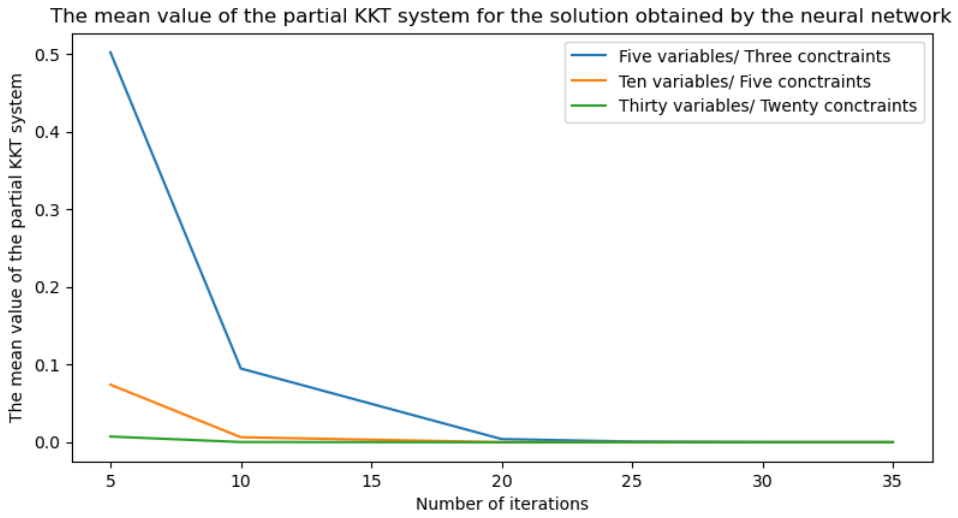


Figure 5: The mean value of the partial KKT system in function of the number of iterations

Although the neural network approach converges to a near optimal solution, it remains a time consuming method. As the size of the problem increases, the matrix $\Phi(z)$ of the neural network becomes large and the ODE solver takes more time to find the solution. Faster ODE solvers might decrease significantly the total CPU time.

5. Conclusion

We propose a dynamical neural network to solve a joint chance constrained problem with multivariate normal distribution. We show the convergence and the stability of such an approach.

We compare the results of our algorithm with those obtained by the expected value model and the individual chance constrained optimization problem. We show that the joint chance constrained approach outperforms individual chance constraint and expected value models.

References

[1] H. M. W. Bruce L. Miller, Chance-constrained programming with joint constraints, Operations Research 13 (1965) 879–1060. doi:10.1287/opre.13.6.930.

[2] A. Prékopa, On probabilistic constrained programming, Proceedings of the Princeton Symposium on Mathematical Programming, Princeton University Press (1970) 113–138.

[3] J. Luedtke, S. Ahmed, A sample approximation approach for optimization with probabilistic constraints, SIAM Journal on Optimization 19 (2) (2008) 674–699. doi:10.1137/070702928.

[4] S. J.Luedtke, G.Nemhauser, An integer programming approach for linear programs with probabilistic constraints, Mathematical Programming (2010) 247–272doi:10.1007/s10107-008-0247-4.

[5] H. H. Lukáš Adam, Martin Branda, R. Henrion, Solving joint chance constrained problems using regularization and benders' decomposition, Annals of Operations Researchdoi:10.1007/s10479-018-3091-9.

[6] W. Chen, M. Sim, J. Sun, C.-P. Teo, From cvar to uncertainty set: Implications in joint chance-constrained optimization, Operations Research 58 (2) (2010) 470–485.
URL http://www.jstor.org/stable/40605930

[7] D. K. Steve Zymler, B. Rustem, Distributionally robust joint chance constraints with second-order moment information, Mathematical Programming 137 (1) (2013) 167–198.

[8] D. Reich, A linear programming approach for linear programs with probabilistic constraints, European Journal of Operational Research 230 (3) (2013) 487–494. doi:https://doi.org/10.1016/j.ejor.2013.04.049.

[9] A. M. W. van Ackooij, R. Henrion, R. Zorgati, Joint chance constrained programming for hydro reservoir management, Optimization and Engineering 15 (2) (2014) 509–531. doi:10.1007/s11081-013-9236-4.

[10] H. T.González Grandón, R.Henrion, A joint model of probabilistic/robust constraints for gas transport management in stationary networks, Computational Management Science 14 (3) (2017) 443–460. doi:10.1007/s10287-017-0284-7.

[11] A. Hyunhee, J. W. Eheart, A screening technique for joint chance-constrained programming for air-quality management, Operations Research 55 (4) (2007) 792–798.
URL www.jstor.org/stable/25147119

[12] A. T.Arnold, R.Henrion, S.Vigerske, A mixed-integer stochastic nonlinear optimization problem with joint probabilistic constraints, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Mathematik, 2013. doi:http://dx.doi.org/10.18452/8435.

[13] 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.

[14] L. H. K.J. Arrow, H. Uzawa, Studies in Linear and Non-Linear Programming, Stanford University Press, California, 1958.

[15] L. Jin, L.-W. Zhang, X.-T. Xiao, Two differential equation systems for equality-constrained optimization, Applied Mathematics and Computation 190 (2) (2007) 1030–1039. doi:https://doi.org/10.1016/j.amc.2006.11.041.
URL https://www.sciencedirect.com/science/article/pii/S0096300306015426

[16] L. Adam, M. Branda, Machine learning approach to chance-constrained problems: An algorithm based on the stochastic gradient descent (2019). arXiv:1905.10986.

[17] S. Zhao, F. You, Distributionally robust chance constrained programming with generative adversarial networks (gans), AIChE Journal 66 (6). doi:10.1002/aic.16963.
URL http://dx.doi.org/10.1002/aic.16963

[18] 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.

[19] A. Nazemi, A dynamical model for solving degenerate quadratic minimax problems with constraints, Journal of Computational and Applied Mathematics 236 (6) (2011) 1282–1295.

doi:https://doi.org/10.1016/j.cam.2011.08.012.
URL https://www.sciencedirect.com/science/article/pii/S0377042711004584

[20] A. Nazemi, E. Sharifi, Solving a class of geometric programming problems by an efficient dynamic model, Communications in Nonlinear Science and Numerical Simulations 18 (3) (2013) 692–709. doi:10.1016/j.cnsns.2012.07.016.

[21] A. Nazemi, F. Omidi, A capable neural network model for solving the maximum flow problem, Journal of Computational and Applied Mathematics 236 (14) (2012) 3498–3513. doi:https://doi.org/10.1016/j.cam.2012.03.001.
URL https://www.sciencedirect.com/science/article/pii/S0377042712001185

[22] A. Nazemi, F. Omidi, An efficient dynamic model for solving the shortest path problem, Transportation Research Part C: Emerging Technologies 26 (2013) 1–19. doi:https://doi.org/10.1016/j.trc.2012.07.005.
URL https://www.sciencedirect.com/science/article/pii/S0968090X12000964

[23] J. Cheng, C. Gicquel, A. Lisser, A second-order cone programming approximation to joint chance-constrained linear programs, in: A. R. Mahjoub, V. Markakis, I. Milis, V. T. Paschos (Eds.), Combinatorial Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 71–80.

[24] M. Jiang, Z. Meng, R. Shen, Partial exactness for the penalty function of biconvex programming, Entropy 23 (2). doi:10.3390/e23020132. URL https://www.mdpi.com/1099-4300/23/2/132

[25] R. K.Miller, A. N. Michel, Ordinary Differential Equations, Academic Press, 1981.

[26] P. F. Gorski Jochen, K. Kathrin, Biconvex sets and optimization with biconvex functions: a survey and extensions, Mathematical Methods of Operations Research (2007) 373–467doi:10.1007/s00186-007-0161-1.

[27] C. Foias, A. E. Frazho, Positive Definite Block Matrices, Birkhäuser Basel, Basel, 1990, pp. 547–586. doi:10.1007/978-3-0348-7712-1_16. URL https://doi.org/10.1007/978-3-0348-7712-1_16

[28] R. J.-B. W. R. Tyrrell Rockafellar, Variational Analysis, Springer, Berlin, Heidelberg, 1998. doi:10.1007/978-3-642-02431-3.

[29] R. M. Murray, S. S. Sastry, L. Zexiang, A Mathematical Introduction to Robotic Manipulation, 1st Edition, CRC Press, Inc., USA, 1994.

[30] S. J.-J. E., Applied nonlinear control / Jean-Jacques E. Slotine,... Weiping Li,..., Prentice Hall, Englewood Cliffs (N.J.), right 1991.