Some heuristic approaches for solving extended geometric programming problems

R. Toscano[1], S. B. Amouri

Université de Lyon
Laboratoire de Tribologie et de Dynamique des Systèmes CNRS UMR5513 ECL/ENISE
58 rue Jean Parot 42023 Saint-Etienne cedex 2

Abstract. In this paper we introduce an extension of standard geometric programming (GP) problems which we call quasi geometric programming (QGP) problems. The idea behind QGP is very simple, it means that a problem become GP when some variables are kept constants. The consideration of this particular kind of nonlinear and possibly non smooth optimization problem is motivated by the fact that many engineering problems can be formulated, or well approximated, as a QGP. However, solving a QGP remains a difficult task due to its intrinsic non-convex nature. This is why we introduce some simple approaches to easily solve this kind of non-convex problems. The interesting thing is that the proposed methods does not require the development of a customized solver and works well with any existing solver able to solve conventional geometric programs. Some considerations on the robustness issue are also presented. Various optimization problems are considered to illustrate the ability of the proposed methods for solving a QGP problem. Comparison with previously published works are also given.

Keywords: Non-convex optimization, Geometric Programming, Quasi Geometric Programming, GP-solver, Robust Optimization.

1 Introduction

Geometric programming (GP) has proved to be a very efficient tool for solving various kinds of engineering problems. This efficiency comes from the fact that geometric programs can be transformed to convex optimization problems for which powerful global optimization methods have been developed. As a result, globally optimal solution can be computed with great efficiency, even for problems with hundreds of variables and thousands of constraints, using recently developed interior-point algorithms. A detailed tutorial of GP and comprehensive survey of its recent applications to various engineering problems can be found in [1].

[1]E-mail address: toscano@enise.fr, Tel.:+33 477 43 84 84; fax: +33 477 43 84 99

An important extension of GP is signomial[2] geometric programming (SGP). Various approaches have been proposed to solve SGP, and a lot of specific algorithms, not always available, have been developed (see for instance [11] and references therein). However, despite these various contributions, solving a SGP problem remain an open issue. This is mainly due to the fact that a SGP is inherently non-convex. Indeed, unlike GP problems, SGP problems remain non-convex in both their primal and dual forms and there is no transformation able to convexify them. Consequently, only a locally optimal solution of a SGP can be computed efficiently[3].

In this paper we introduce a particular type of nonlinear program which we call quasi geometric programming (QGP) problems. The idea behind QGP is very simple, it means that a problem become GP when some variables are kept constants[4]. For this kind of problems (QGP), we introduce two approaches for their resolution. The interesting thing is that the proposed approaches does not require development of new solvers and work well with any existing solver that is able to solve conventional convex programs (for instance cvx see [4]). From a practical point of view this is very interesting because the engineers often do not have the time to develop specific algorithm for solving particular problems. Finally, one of the main objectives of this work is to introduce some procedures that are easy to use for solving engineering problems.

The rest of this paper is organized as follows. In section 2, we provide a short introduction to GP. Section 3 is the main part of this paper. It introduces the notion of quasi geometric programming problems as well as their resolutions using available convex solvers. In many practical problems, some parameters are not precisely known, this aspect is discussed in section 4 which is devoted to the robustness issue. In section 5 various

[2]A signomial is sum of terms of the form $c_i x_1^{a_i^1} x_2^{a_i^2} \cdots x_n^{a_i^n}$ where the coefficients c_i are allowed to be negative.

[3]It is possible to compute the globally optimal solution of a SGP, but this can requires prohibitive computation, even for relatively small problems.

[4]As we will see later, QGP include some SGP problems.

optimization problems are considered to show how the concept of QGP can be used to solve them efficiently. We give concluding remarks in section 6.

2 Geometric Programming

GP is a special type of nonlinear, non-convex optimisation problems. A useful property of GP is that it can be turned into a convex optimization problem and thus a local optimum is also a global one, which can be computed very efficiently. Since QGP is based on the resolution of GP, this section gives a short presentation of GP both in standard and convex form.

2.1 Standard formulation

Monomials are the basic elements for formulating a geometric programming problem. A monomial is a positive function f defined by:

$$f(x) = c x_1^{a^1} x_2^{a^2} \cdots x_n^{a^n} \tag{1}$$

where $x_1, \cdots x_n$ are n positive variables, c is a positive multiplicative constant and the exponentials a^i, $i = 1, \cdots, n$ are real numbers. We will denote by x the vector $(x_1, \cdots, x_n)$. A sum of monomial is called a posynomial:

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_k^1} x_2^{a_k^2} \cdots x_n^{a_k^n} \tag{2}$$

Minimizing a posynomial subject to posynomial upper bound inequality constraints and monomial equality constraints is called GP in standard form:

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 1, & i = 1, \cdots, m \\ & g_i(x) = 1, & i = 1, \cdots, p \end{array} \tag{3}$$

where f_i, $i = 0, \cdots, m$, are posynomials and g_i, $i = 1, \cdots, p$, are monomials. Note that monomials and posynomials are always assumed to be positive functions of positive variables.

2.2 Convex formulation

GP in standard form is not a convex optimisation problem[5], but it can be transformed to a convex problem by an appropriate change of variables and a log transformation of the objective and constraint functions. Indeed, if we introduce the change of variables $y_i = \log x_i$ (and so $x_i = e^{y_i}$), the posynomial function (2) becomes:

$$f(y) = \sum_{k=1}^{K} c_k \exp\left(\sum_{i=1}^{n} a_k^i y_i\right) = \sum_{k=1}^{K} \exp(a_k^T y + b_k) \tag{4}$$

where $b_k = \log c_k$, taking the log we obtain $\bar{f}(y) = \log\left(\sum_{k=1}^{K} \exp(a_k^T y + b_k)\right)$, which is a convex function of the new variable y. Applying this change of variable and the log transformation to the problem (3) gives the following equivalent optimization problem:

$$\begin{array}{ll} \text{minimize} & \bar{f}_0(y) = \log\left(\sum_{k=1}^{K_0} \exp(a_{0k}^T y + b_{0k})\right) \\ \text{subject to} & \bar{f}_i(y) = \log\left(\sum_{k=1}^{K_i} \exp(a_{ik}^T y + b_{ik})\right) \leq 0, \quad i = 1, \cdots, m \\ & \bar{g}_j(y) = a_j^T y + b_j = 0, \quad j = 1, \cdots, p \end{array} \tag{5}$$

Since the functions $\bar{f}_i$ are convex, and $\bar{g}_j$ are affine, this problem is a convex optimization problem, called geometric program in convex form. However, in some practical situations, it is not possible to formulate the problem in standard geometric form, the problem is then not convex. In this case the problem is generally difficult to solve even approximately. In these situations, it seems very useful to introduce simple approaches that are able to compute a good suboptimal solution (if not the global optimum). In this spirit, we are now ready to introduce the concept of quasi geometric programming.

[5] A convex optimization problem consists in minimizing a convex function subject to convex inequality constraints and linear equality constraints.

3 Quasi Geometric Programming (QGP)

Consider the nonlinear program defined by

$$\begin{array}{ll} \text{minimize} & f_0(z) \\ \text{subject to} & f_i(z) \leq 0, \quad i = 1, \cdots, m \\ & g_j(z) = 0, \quad j = 1, \cdots, p \end{array} \tag{6}$$

where the vector[6] $z \in \mathbf{R}^n_{++}$ include all the optimization variables, $f_0 : \mathbf{R}^n_{++} \to \mathbf{R}$ is the objective function or cost function, $f_i : \mathbf{R}^n_{++} \to \mathbf{R}$ are the inequality constraint functions and $g_j : \mathbf{R}^n_{++} \to \mathbf{R}$ are the equality constraint functions. This nonlinear optimization problem is called a quasi geometric programming problem if it can be formulated into the following form:

$$\begin{array}{ll} \underset{x,\,\xi}{\text{minimize}} & \varphi_0(x,\xi) - Q_0(\xi) \\ \text{subject to} & \varphi_i(x,\xi) \leq Q_i(\xi), \quad i = 1, \cdots, m \\ & h_j(x,\xi) = Q'_j(\xi), \quad j = 1, \cdots, p \end{array} \tag{7}$$

where $x \in \mathbf{R}^{n_x}_{++}$ and $\xi \in \mathbf{R}^{n_\xi}_{++}$ with $n_x + n_\xi = n$, $(x,\, \xi)$ is a partition of vector z. The functions $\varphi_i(x,\xi)$, $i = 0, \cdots, m$ are posynomials and $h_j(x,\xi)$, $j = 1, \cdots, p$ are monomials. With respect to the nature of the functions Q_0, Q_i and Q'_j, we consider two cases for the solution of the QGP (7).

1. In the first case, called quasi geometric programming in posynomial form, the functions $Q_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$ are ratios of posynomial functions and thus are positive functions.

2. In the second case, called quasi geometric programming in general form, except their positivity, nothing special is assumed about $Q_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$, these functions can be even non-smooth.

[6] In our notations, $\mathbf{R}_{++}$ represents the set of positive real numbers.

It is important to insist on the fact that in these two cases, the problem (7) cannot be converted into a GP in the standard form (3) and thus the problem is not convex. As a consequence, no approach exists for finding quickly even a sub optimal solution by using available convex solvers. Although specific algorithms can be designed to find out a sub optimal solution to problem (7) both in posynomial or general form, we think that it could be very interesting to solve these problems by using standard convex solvers. Indeed, this should be interesting for at least two reasons. Firstly the ability of solving problem (7) using available convex solvers allows time saving; the development of a specific algorithm is always a long process and there are often no resources to develop customized approaches in industry. Secondly, the available convex solvers are efficient and easy to use. Problems involving tens of variables and hundreds of constraints can be solved on a small current workstation in less than one second. All these reasons justify the approaches presented in sections 3.1 and 3.2. Indeed, these methods does not require the development of particular algorithms and are based on the use of available convex solvers.

3.1 Solution of a QGP in posynomial form

The QGP (7) can be reformulated as follows:

$$\begin{array}{ll} \underset{\lambda,\, x,\, \xi}{\text{minimize}} & \lambda^{-1} \\ \text{subject to} & \lambda + \varphi_0(x,\xi) \leq Q_0(\xi) \\ & \varphi_i(x,\xi) \leq Q_i(\xi), \quad i = 1, \cdots, m \\ & h_j(x,\xi) = Q'_j(\xi), \quad j = 1, \cdots, p \end{array} \tag{8}$$

where $\lambda \in \mathbf{R}_{++}$ is an additional decision variable. Note that $\lambda + \varphi_0(x,\xi)$ is a posynomial function. This problem is not solvable as a GP since $Q_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$ are not monomials. However, for a fixed value of ξ this problem becomes a GP in standard form. This suggests that the quasi geometric problem (8) can be solved by considering a succession of GP. The details of the proposed approach for solving a QGP in posynomial form are

presented in the procedure below.

P1: Procedure for solving QGP in posynomial form

1. *Solve the following standard GP problem:*

$$\begin{aligned}
\underset{\lambda,\, x,\, \xi}{\text{minimize}} \quad & \lambda^{-1} \\
\text{subject to} \quad & \frac{(\lambda + \varphi_0(x,\xi))\mathcal{D}(Q_0(\xi))}{\Gamma(\mathcal{N}(Q_0(\xi)))} \leq 1 \\
& \frac{\varphi_i(x,\xi)\mathcal{D}(Q_i(\xi))}{\Gamma(\mathcal{N}(Q_i(\xi)))} \leq 1, \quad i = 1, \cdots, m \\
& \frac{h_j(x,\xi)\mathcal{D}(Q'_j(\xi))}{\Gamma(\mathcal{N}(Q'_j(\xi)))} \leq 1 \\
& \frac{\mathcal{N}(Q'_j(\xi))}{h_j(x,\xi)\Gamma(\mathcal{D}(Q'_j(\xi)))} \leq 1, \quad j = 1, \cdots, p
\end{aligned} \tag{9}$$

   This gives the solution denoted (x', ξ^*).

2. *For the value* ξ^* *solve the following standard GP problem:*

$$\begin{aligned}
\underset{\lambda,\, x}{\text{minimize}} \quad & \lambda^{-1} \\
\text{subject to} \quad & \frac{\lambda + \varphi_0(x,\xi^*)}{Q_0(\xi^*)} \leq 1 \\
& \frac{\varphi_i(x,\xi^*)}{Q_i(\xi^*)} \leq 1, \quad i = 1, \cdots, m \\
& \frac{h_j(x,\xi^*)}{Q'_j(\xi^*)} = 1, \quad j = 1, \cdots, p
\end{aligned} \tag{10}$$

   this gives the optimal solution x^* *w.r.t* ξ^**. The final suboptimal solution is given by* (x^*, ξ^*).

As this procedure shows, the solution of QGP (7) is decomposed into two main steps. In the first step, an approximate problem of QGP (7) is solved. The approximation is based on an optimal lower approximation of a posynomial by a monomial i.e.: $\sum_i u_i(x_1, \cdots, x_n) \geq cx_1^{a_1} \cdots x_n^{a_n}$, where u_i are monomials, c is a positive constant and the exponentials $a^1 \cdots a^n$, are real numbers. In problem (9), $\Gamma(.)$ represents the optimal lower monomial approximation obtained from the posynomial given in argument. The optimal lower monomial approximation of a given posynomial is presented in appendix A2. In our notations, $\mathcal{N}(.)$ and $\mathcal{D}(.)$ are, respectively, the numerator and the denominator of the rational posynomial function passed in argument. The details for the derivation of problem (9) are given in appendix A1.

Since lower monomial approximations are used, the set of feasible solutions of problem (9) is a convex subset of the set of feasible solutions of QGP (7) (see the details in appendix A1). Therefore, the value ξ^* found by solving problem (9) is always feasible for QGP problem (7). In the second step, QGP (7) is solved with the value of $\xi = \xi^*$ found at the first step. QGP (7) with ξ kept constant becomes a standard GP, its solution leads to the optimal value x^*. The solution thus obtained (x^*, ξ^*) is a good suboptimal solution of QGP (7) in the sense that $\varphi(x^*, \xi^*) \leq \varphi(x', \xi^*)$ where (x', ξ^*) is the global solution of problem (9).

Remark 1. As already said, the optimal solution, say (x'_-, ξ^*_-), of GP (9) is also a feasible solution of QGP (8) and since the domain of feasible solutions of (9) is a subset of the domain of feasible solutions of QGP (8) we have $\varphi(x_{opt}, \xi_{opt}) \leq \varphi(x'_-, \xi^*_-)$, where (x_{opt}, ξ_{opt}) is the global solution of QGP (8). If instead of lower-monomial approximations we utilize in (9) upper-monomial approximations, the solution, say (x'_+, ξ^*_+), of GP (9) satisfies $\varphi(x'_+, \xi^*_+) \leq \varphi(x_{opt}, \xi_{opt})$, this is because in this case, the set of feasible solutions of GP (9) with upper-monomial approximations includes the set of feasible solutions of QGP (8). Finally, by considering the GP (9) with upper-lower monomial approximations, it is

guaranteed that the global optimum of QGP (8) is within the bounds:

$$\varphi(x'_+, \xi^*_+) \leq \varphi(x_{opt}, \xi_{opt}) \leq \varphi(x'_-, \xi^*_-)$$

where (x'_+, ξ^*_+) is the global solution of GP (9) with upper-monomial approximations and (x'_-, ξ^*_-) is the global solution of GP (9) with lower-monomial approximations.

Remark 2. Problem (9) is a convex restriction of the original problem (7). Therefore, problem (9) can be infeasible even if (7) is feasible. In this situation, we can solve a relaxed problem using upper monomials approximation (see appendix A2)), if the solution thus found is feasible for the original problem (7), this means that we have found the optimal solution. In the case where the solution of the relaxed problem is infeasible for the original problem it is always possible to solve (7) using the method for solving a QGP in general form (see section 3.2).

The solution (x^*, ξ^*) obtained via procedure P1 can be further improved by looking for a better one in the vicinity of (x^*, ξ^*). A good way to do that is to iteratively solve a linear approximation of the original problem (7) around the solution found so far. The principle is to select a starting point (in our case the solution found via procedure P1), and the original problem (7) is linearized about this point to give a linear problem which can be efficiently solved using any convex solver. The solution thus found is then used as a new point to linearize the problem (7). This process of resolution continues until no further improvement can be found. More precisely, the principle of the method is presented in the following iterative procedure.

P2: Algorithm for the improvement of a given suboptimal solution

1. *Let* (x', ξ') *be the solution found using the procedure P1 and select the precision tolerance* ϵ *(e.g.* 10^{-4}*).*

2. *Solve the following linear problem:*

$$
\begin{aligned}
\underset{x,\,\xi}{\text{minimize}} \quad & \varphi_0(x',\xi') - Q_0(\xi') + \nabla_x\varphi_0(x',\xi')^T(x-x') \\
& + \left(\nabla_\xi\varphi_0(x',\xi') - \nabla_\xi Q_0(\xi')\right)^T(\xi-\xi') \\
\\
\text{subject to} \quad & \varphi_i(x',\xi') - Q_i(\xi') + \nabla_x\varphi_i(x',\xi')^T(x-x') \\
& + \left(\nabla_\xi\varphi_i(x',\xi') - \nabla_\xi Q_i(\xi')\right)^T(\xi-\xi') \le 0, \quad i=1,\cdots,m \\
\\
& h_j(x',\xi') - Q'_j(\xi') + \nabla_x h_j(x',\xi')^T(x-x') \\
& + \left(\nabla_\xi h_j(x',\xi') - \nabla_\xi Q'_j(\xi')\right)^T(\xi-\xi') = 0, \quad j=1,\cdots,p \\
\\
& (x-x',\,\xi-\xi') \in \Delta
\end{aligned}
\tag{11}
$$

This gives the current local solution denoted (x^*,ξ^*).

3. *If* $\|(x^*,\xi^*)-(x',\xi')\| > \epsilon$, *set* $(x',\xi') := (x^*,\xi^*)$ *go to 2, else end of the algorithm. The final solution is given by* (x^*,ξ^*).

It is usually necessary to bound the steps taken in the iterations to ensure that the decision variables remain in the feasible domain. These bound are the additional constraints $(x-x',\,\xi-\xi') \in \Delta$, where the "size" of Δ defines the extent of the domain in which the linear approximation can be considered as a valid one. However, since we start with a good suboptimal solution, the choice of Δ is usually easy to do.

3.2 Solution of a QGP in general form

In this section, the only particular assumption made about the functions $Q_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$, is that they are positive. Except for their positivity, no other particular assumption is made; these functions can be even non-smooth.

To solve this kind of problem, we can see the QGP (7) like a function of ξ that we want to minimize:

$$\begin{array}{ll} \underset{\xi}{\text{minimize}} & F(\xi) = J(\xi) - Q_0(\xi) \\ \text{subject to} & \underline{\xi} \leq \xi \leq \bar{\xi} \end{array} \tag{12}$$

where $\underline{\xi}$ and $\bar{\xi}$ are simple bound constraints on the decision variable ξ, and the function $J(\xi)$ is defined as follows:

$$\begin{array}{rll} J(\xi) = & \underset{x}{\min} & \varphi_0(x,\xi) \\ & \text{s. t.} & \varphi_i(x,\xi) \leq Q_i(\xi), \quad i = 1, \cdots, m \\ & & h_j(x,\xi) = Q'_j(\xi), \quad j = 1, \cdots, p \end{array} \tag{13}$$

Problem (12) is a non-convex unconstrained optimization problem[7] and can be solved using well known zero order algorithms[8] such as: Nelder-Mead simplex method (NMSM) [8], simulated annealing (SA) [10], genetic algorithm (GA) [5], particle swarm optimization (PSO) [9] or Heuristic Kalman Algorithm (HKA) [15]. In the case where the objective function $F(\xi)$ in (12) is differentiable, we can also use a multi-start first order or second order methods to solve this problem. The code associated to these various algorithms are easily available and thus don't need to be programmed[9]. When ξ is kept constant, problem (13) is a standard GP which can be solved very efficiently using available convex solvers. This suggests that we can solve the QGP problem in general form with a two levels procedure. At the first level, the chosen search algorithm (eg. NMSM) is used to select a value of ξ within the bounds. For the selected value of ξ, the standard GP (13) is solved using available solvers. This procedure is continued until some stopping rule is satisfied. The suggested procedure is formalized more precisely in the following algorithm.

[7] We have only simple bound constraints on the decision variable ξ.

[8] Zero order algorithms does not require the knowledge of the derivatives of the objective function. Thus smoothness is not required.

[9] For instance, the Nelder-Mead simplex method is available in MatLab trough the function `fminsearch`.

P3: Algorithm for solving QGP in general form

1. *If a good initial guess* (x', ξ') *is available set* $F_{best} := \varphi_0(x', \xi') - Q_0(\xi')$*, else set* $F_{best} :=$ `inf`.

2. *Using a zero order algorithm (ZOA), generate* ξ^* *such that* $\underline{\xi} \leq \xi^* \leq \bar{\xi}$.

3. *For the value* ξ^* *solve the standard GP problem (13). This gives the optimal solution* x^* *w.r.t* ξ^*.

4. *If problem (13) is not feasible, then set* $F :=$ `inf` *and goto 2. Else set* $F := \varphi_0(x^*, \xi^*) - Q_0(\xi^*)$.

5. *If* $F \geq F_{best}$ *then goto 2. Else set* $F_{best} := F$, $x_{best} := x^*$, $\xi_{best} := \xi^*$ *and goto 2.*

6. *At the end of the ZOA, the optimal solution is given by* (x_{best}, ξ_{best}).

In this algorithm, `inf` represents the IEEE arithmetic representation for positive infinity, and F_{best} is a variable containing the current best objective function. Note that the use of "global optimization methods" like SA, GA, PSO or HKA, increases the probability of finding a global optimum but this is not guaranteed, except perhaps if the search space of problem (12) is explored very finely, but this cannot be done in a reasonable time.

Remark 3. To speed up the convergence, it is desirable to start this algorithm with a good initial guess (x', ξ'). Indeed, in the first level, a value of ξ is selected within the bounds $\underline{\xi}$, $\bar{\xi}$, but this value of ξ is not necessarily feasible for the GP problem. As a consequence, a very long time can be spent to find a feasible ξ. Thus the computation time can be strongly improved by using a good starting point. A good initial guess can be

obtained by solving the following GP problem:

$$\begin{array}{ll} \underset{\xi}{\text{minimize}} & \lambda^{-1} \\ \text{subject to} & \lambda + \varphi_0(x,\xi) \leq \Gamma_\epsilon(Q_0(\xi)) \\ & \varphi_i(x,\xi) \leq \Gamma_\epsilon(Q_i(\xi)), \quad i = 1, \cdots, m \\ & h_j(x,\xi) = \Gamma_\epsilon(Q'_j(\xi)), \quad j = 1, \cdots, p \end{array} \tag{14}$$

which is an approximation of the considered QGP problem. The notation $\Gamma_\epsilon(.)$ represents the ϵ-approximation of the function passed in argument (see appendix A2).

4 Robustness issue

Until now it was implicitly assumed that the parameters (i.e. the problem data) which enter in the formulation of a QGP problem are precisely known. However, in many practical applications some of these parameters are subject to uncertainties. It is then important to be able to calculate solutions that are insensitive to parameters uncertainties; this leads to the notion of optimal robust design. We say that the design is robust, if the various specifications (i.e. the constraints) are satisfied for a set of values of the parameters uncertainties. In this section we show how to use the methods presented above to develop designs that are robust with respect to some parameters uncertainties.

Let $\theta = [\theta_1\ \theta_2\ \cdots\ \theta_q]^T$ be the vector of uncertain parameters. It is assumed that θ (also called the parameter box) lie in a bounded set Θ defined as follows:

$$\Theta = \left\{\theta \in \mathbf{R}^q : \underline{\theta} \preceq \theta \preceq \bar{\theta}\right\}, \tag{15}$$

where the notation $\preceq$ denotes the componentwise inequality between two vectors: $v \preceq w$ means $v_i \leqslant w_i$ for all i. The vectors $\underline{\theta} = [\underline{\theta}_1 \cdots \underline{\theta}_q]^T$, $\bar{\theta} = [\bar{\theta}_1 \cdots \bar{\theta}_q]^T$ are the bounds of uncertainty of the parameters vector θ. Thus, the uncertain vector belong to the q-dimensional hyperrectangle Θ also called the parameter box. In these conditions, the QGP problem (7), or equivalently (8), must be expressed in term of functions of (x, ξ), the

design variables, and θ the vector of uncertain parameters. The robust version of the quasi geometric problem (8) is then written as follows:

$$\begin{array}{ll} \underset{\lambda,\, x,\, \xi}{\text{minimize}} & \lambda^{-1} \\ \text{subject to} & \lambda + \varphi_0(x,\xi,\theta) \leqslant \varphi_0'(\xi,\theta), \quad \forall \theta \in \Theta \\ & \varphi_i(x,\xi,\theta) \leqslant Q_i(\xi,\theta), \quad i = 1,\cdots,m, \quad \forall \theta \in \Theta \\ & h_j(x,\xi,\theta) = Q_j'(\xi,\theta), \quad j = 1,\cdots,p, \quad \forall \theta \in \Theta \end{array} \tag{16}$$

where the functions φ_i, $i = 0,\cdots,m$, are posynomial functions of (x,ξ), for each value of θ, and the functions h_j, $j = 1,\cdots,p$, are monomial functions of (x,ξ), for each value of θ. In the case of a QGP in posynomial form, the function φ_0', is posynomial for each value of θ, and the functions Q_i and Q_j' are ratio of posynomial functions for each θ. In the case of a QGP in general form, the functions φ_0', Q_i and Q_j' are only assumed to be positive for each θ. The approach proposed in this section apply both in the case of a robust QGP in posynomial form and robust QGP in general form.

We consider the resolution of the robust QGP problem in the case of a finite set. Let $\Theta_N = \{\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(N)}\}$ be a finite set of possible vector parameter values. This finite set may (can) be imposed by the problem itself or can be obtained by sampling the continuous set Θ defined in (15). For instance, we might sample each interval $[\underline{\theta}_i,\ \bar{\theta}_i]$ with three values: $\underline{\theta}_i$, $\frac{\underline{\theta}_i + \bar{\theta}_i}{2}$ and $\bar{\theta}_i$, and form every possible combination of parameter values, this lead to $N = 3^q$ different vector parameters.

Whatever how the finite set is obtained, we have to determine a solution $(x,\ \xi)$ that satisfy the QGP problem for all possible vector parameters. To do that, we have only to replicate the constraints for all possible vector parameters. Thus, in the case of a finite set Θ_N, the robust QGP problem is formulated as follows:

$$
\begin{array}{ll}
\underset{\lambda, x, \xi}{\text{minimize}} & \lambda^{-1} \\
\text{subject to} & \lambda + \varphi_0(x, \xi, \theta^{(k)}) \leqslant \varphi_0'(\xi, \theta^{(k)}), \quad k = 1, \cdots, N \\
& \varphi_i(x, \xi, \theta^{(k)}) \leqslant Q_i(\xi, \theta^{(k)}), \quad i = 1, \cdots, m, \quad k = 1, \cdots, N \\
& h_j(x, \xi, \theta^{(k)}) = Q_j'(\xi, \theta^{(k)}), \quad j = 1, \cdots, p, \quad k = 1, \cdots, N
\end{array}
\tag{17}
$$

Whit respect to the nature of the functions φ_0', Q_i and Q_j, problem (17) can be solved as a QGP problem in posynomial form (see section 3.1) or as a QGP problem in general form (see section 3.2).

5 Numerical examples

In this section we illustrate the applicability of the proposed methods through some numerical examples (which are all come from the literature). For all these examples we have used the GP solver cvx [4], and the implementation has been done on a Intel core 2 CPU 2GHz with 512MB memory microcomputer. The computational results are compared with several other existing methods to show the practical interest of the proposed approach. In our experiments, the number of calls relative to QGP represents the number of time that the solver is launched. In the other cases, the number of calls represents the number of evaluations of all constraints and objective function.

5.1 Example 1

This first example is borrowed from [11] in which the problem was solved using a global optimization algorithm via Lagrangian relaxation.

$$
\begin{array}{ll}
\text{minimize} & 0.5 x_1 x_2^{-1} - x_1 - 5 x_2^{-1} \\
\text{subject to} & 0.01 x_2 x_3^{-1} + 0.01 x_2 + 0.0005 x_1 x_3 \leq 1 \\
& 70 \leq x_1 \leq 150, \;\; 1 \leq x_2 \leq 30, \;\; 0.5 \leq x_3 \leq 21
\end{array}
$$

This problem can be rewritten as follows:

$$\begin{aligned} \text{minimize} \quad & \lambda^{-1} \\ \text{subject to} \quad & \frac{\lambda + 0.5x_1x_2^{-1}}{x_1 + 5x_2^{-1}} \leq 1 \\ & 0.01x_2x_3^{-1} + 0.01x_2 + 0.0005x_1x_3 \leq 1 \\ & 70 \leq x_1 \leq 150, \;\; 1 \leq x_2 \leq 30, \;\; 0.5 \leq x_3 \leq 21 \end{aligned}$$

which is QGP in the variables $(x_1,\, x_2)$. The method of section 3.1, was applied to solve this optimization problem and the solution found is presented in Table 1. It can be seen that the solution thus obtained is very significantly better than that found using the method described in [11]. Note that the QGP-solution cannot be improved since the variables x_1 and x_2 are on the bounds. We can then say that the solution found is global.

Table 1: Comparison of the solutions found via the method in [11] and QGP.

Method	x_1	x_2	x_3	f_0	Nb of calls
[11]	88.6274	7.9621	1.3215	-83.6898	1754
QGP	149.9999	29.9999	2.0269	-147.6666	2

5.2 Example 2

This second example is borrowed from [17] in which the problem was solved using a specific optimization algorithm for generalized geometric programming problems.

$$\begin{aligned} \text{minimize} \quad & x_1 \\ \text{subject to} \quad & 3.7x_1^{-1}x_2^{0.85} + 1.985x_1^{-1}x_2 + 700.3x_1^{-1}x_3^{-0.75} \leq 1 \\ & 0.7673x_3^{0.05} \leq 1 + 0.05x_2 \\ & 5 \leq x_1 \leq 15, \; 0.1 \leq x_2 \leq 5, \; 380 \leq x_3 \leq 450 \end{aligned}$$

This problem is QGP in x_2 and thus can be solved using the method of section 3.1. The solution found is presented in Table 2 which shows also the result obtained in [17].

Table 2: Comparison of the solutions found via the method in [17] and QGP.

Method	x_1	x_2	x_3	f_0	Nb of calls
[17]	11.9538	0.8150	445.1249	11.9538	67
QGP	12.0097	0.8369	449.9999	12.0097	2

It can be seen that the obtained result is a very good suboptimal solution and was found without any particular effort: only two GP-solver calls.

5.3 Example 3

This third example is borrowed from [17, 11]. The problem to solve is defined as:

$$\begin{aligned} \text{minimize} \quad & x_3^{0.8} x_4^{1.2} \\ \text{subject to} \quad & x_1 x_4^{-1} + x_2^{-1} x_4^{-1} \leq 1 \\ & x_3 \leq x_1^{-2} + x_2 \\ & 0.1 \leq x_1 \leq 1,\ 5 \leq x_2 \leq 10,\ 8 \leq x_3 \leq 15,\ 0.01 \leq x_4 \leq 1 \end{aligned}$$

which is QGP in (x_1, x_2) and thus can be solved using the method of section 3.1. The solution found is presented in Table 3 which shows also the results obtained in [17, 11].

Table 3: Comparison of the solutions found via the methods in [17, 11] and QGP.

Method	x_1	x_2	x_3	x_4	f_0	Nb of calls
[17]	0.1358	9.9324	8.6973	0.2365	1.0000	171
[11]	0.1015	7.3197	8.0169	0.2395	0.9514	175
QGP	0.1000	9.9999	8.0000	0.1999	0.765	2

It can be seen that the obtained solution is much better than that found in [17, 11].

5.4 Example 4

This example is borrowed from [6] in which the problem was solved using a co-evolutionary particle swarm optimization. In this problem, the objective is to minimize the total cost including the cost of the material, forming and welding of a cylindrical vessel.

$$\begin{aligned}
\text{minimize} \quad & 0.6224x_1x_3x_4 + 1.7778x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
\text{subject to} \quad & -x_1 + 0.0193x_3 \leq 0 \\
& -x_2 + 0.009543x_3 \leq 0 \\
& -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
& 1 \leq x_1,\, x_2 \leq 99,\; 10 \leq x_3,\, x_4 \leq 200
\end{aligned}$$

By introducing an additional variable x_5, this problem can be reformulated as follows:

$$\begin{aligned}
\text{minimize} \quad & 0.6224x_1x_3x_4 + 1.7778x_2x_3^2 + 3.1661x_1^2x4 + 19.84x_1^2x_3 \\
\text{subject to} \quad & 0.0193x_3/x_1 \leq 1, \quad 0.009543x_3/x_2 \leq 1 \\
& \frac{1296000}{\pi x_3^3(x_5 + 4/3)} \leq 1, \quad \frac{x_4}{x_3x_5} = 1 \\
& 1 \leq x_1,\, x_2 \leq 99, \;\; 10 \leq x_3,\, x_4 \leq 200, \;\; \tfrac{1}{20} \leq x_5 \leq 20
\end{aligned}$$

which is QGP in x_5. The method of section 3.1, was applied to solve this optimization problem and the solution found is presented in Table 4. It can be seen that the solution thus obtained is significantly better than that found using the method described in [6].

Table 4: Comparison of the solutions found via the method in [6] and QGP.

Method	x_1	x_2	x_3	x_4	f_0	Nb of calls
[6]	0.8125	0.4375	42.0913	176.7465	6061.0777	32500
QGP	0.7785	0.3848	40.3366	199.7631	5885.8942	2

5.5 Example 5

A welded beam is designed for minimum cost subject to constraints on shear stress $\tau(x)$, bending stress in the beam $\sigma(x)$, buckling load on the bar Pc, end deflection of the beam

$\delta(x)$, and side constraints. The problem can be mathematically formulated as follows [12]:

$$\begin{aligned}
\text{minimize} \quad & (1+c_1)x_1^2x_2 + c_2x_3x_4(L+x_2) \\
\text{subject to} \quad & \tau(x) - \tau_{max} \leq 0, \quad \sigma(x) - \sigma_{max} \leq 0 \\
& g_3(x) = x_1 - x_4 \leq 0, \quad h_{min} - x_1 \leq 0 \\
& g_4(x) = c_1x_1 + c_2x_3x_4(L+x_2) - 5 \leq 0 \\
& \delta(x) - \delta_{max} \leq 0, \quad P - P_c(x) \leq 0 \\
& 0.1 \leq x_1,\, x_4 \leq 2, \quad 0.1 \leq x_2,\, x_3 \leq 10
\end{aligned} \tag{18}$$

where

$$\begin{aligned}
& \tau(x) = \sqrt{\tau_1^2 + 2\tau_1\tau_2\frac{x_2}{2R} + \tau_2^2}, \;\; \tau_1 = \frac{P}{\sqrt{2}x_1x_2}, \;\; \tau_2 = \frac{MR}{I} \\
& M = P\left(L + \frac{x_2}{2}\right), \;\; R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, \\
& I = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}, \;\; \sigma(x) = \frac{6PL}{x_4x_3^2}, \\
& \delta(x) = \frac{4PL^3}{Ex_3^2x_4}, \;\; P_c(x) = \frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)
\end{aligned} \tag{19}$$

and

$$\begin{aligned}
& c_1 = 0.10471, \;\; c_2 = 0.04811, \;\; P = 6\times10^3, \;\; L = 14, \\
& E = 3\times10^7, \;\; G = 1.2\times10^7, \;\; h_{min} = 0.125, \\
& \delta_{max} = 0.25, \;\; \tau_{max} = 1.36\times10^4, \;\; \sigma_{max} = 3\times10^4
\end{aligned} \tag{20}$$

In [2] this problem has been solved using a multi-objective genetic algorithm. More recently, in [6], this problem has been solved using a co-evolutionary particle swarm optimization. This problem is QGP in the variables (x_1, x_2, x_3) and thus can be solved using the method of section 3.1. From Table 5, it can be seen that the solution thus obtained is better than those previously obtained.

Table 5: Comparison of the solutions found via the methods in [2, 6] and QGP.

Method	x_1	x_2	x_3	x_4	f_0	Nb of calls
[2]	0.2059	3.4713	9.0202	0.2064	1.7282	80000
[6]	0.2023	3.5442	9.0482	0.2057	1.7280	200000
QGP	0.2057	3.2718	9.0366	0.2057	1.6978	2

5.6 Example 6

This example is borrowed from [3] in which the problem was solved using a constrained particle swarm optimizer. This problem is related to the design of a speed reducer. The objective is to minimize the weight of the speed reducer subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft. The corresponding optimization problem is formulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
& -1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \\
\text{subject to} \quad & \frac{27}{x_1x_2^2x_3} \leq 1, \quad \frac{397.5}{x_1x_2^2x_3^2} \leq 1, \quad \frac{1.93x_4^3}{x_2x_3x_6^4} \leq 1, \quad \frac{1.93x_5^3}{x_2x_3x_7^4} \leq 1 \\
& \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} \leq 1 \\
& \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} \leq 1 \\
& \frac{x_2x_3}{40} \leq 1, \quad \frac{5x_2}{x_1} \leq 1, \quad \frac{x_1}{12x_2} \leq 1 \\
& \frac{1.5x_6 + 1.9}{x_4} \leq 1, \quad \frac{1.1x_7 + 1.9}{x_5} \leq 1 \\
& 2.6 \leq x_1 \leq 3.6,\ 0.7 \leq x_2 \leq 0.8,\ 17 \leq x_3 \leq 28,\ 7.3 \leq x_4 \leq 8.3 \\
& 7.8 \leq x_5 \leq 8.3,\ 2.9 \leq x_6 \leq 3.9,\ 5.0 \leq x_7 \leq 5.5
\end{aligned}
$$

This problem can be rewritten into the following form:

$$
\begin{aligned}
\text{minimize} \quad & \lambda^{-1} \\
\text{subject to} \quad & (\lambda+\varphi_0)/x_1 \leq \varphi_0' \\
& \frac{27}{x_1x_2^2x_3} \leq 1, \quad \frac{397.5}{x_1x_2^2x_3^2} \leq 1, \quad \frac{1.93x_4^3}{x_2x_3x_6^4} \leq 1, \quad \frac{1.93x_5^3}{x_2x_3x_7^4} \leq 1 \\
& \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2+16.9\times10^6} \leq 1 \\
& \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2+157.5\times10^6} \leq 1 \\
& \frac{x_2x_3}{40} \leq 1, \quad \frac{5x_2}{x_1} \leq 1, \quad \frac{x_1}{12x_2} \leq 1 \\
& \frac{1.5x_6+1.9}{x_4} \leq 1, \quad \frac{1.1x_7+1.9}{x_5} \leq 1 \\
& 2.6 \leq x_1 \leq 3.6,\ 0.7 \leq x_2 \leq 0.8,\ 17 \leq x_3 \leq 28,\ 7.3 \leq x_4 \leq 8.3 \\
& 7.8 \leq x_5 \leq 8.3,\ 2.9 \leq x_6 \leq 3.9,\ 5.0 \leq x_7 \leq 5.5
\end{aligned}
$$

where φ_0 and φ_0' are defined as follows:

$$
\begin{aligned}
\varphi_0 = \ & 0.7854x_1x_2^2x_3(3.3333x_3+14.9334)+7.4777(x_6^3+x_7^3)+0.7854(x_4x_6^2+x_5x_7^2) \\
\varphi_0' = \ & 33.8456x_2^2+1.508(x_6^2+x_7^2)
\end{aligned}
$$

Thus, this equivalent problem is QGP in (x_2, x_6, x_7). The method of section 3.1 was applied to solve this optimization problem and the solution found is presented in Table 6.

Table 6: Comparison of the solutions found via the method in [3] and QGP.

Method	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f_0	Nb of calls
[3]	3.5000	0.7	17	7.3000	7.8000	3.3502	5.2867	2996.3481	24000
QGP	3.5000	0.7	17	7.3000	7.8000	3.3498	5.2867	2996.2376	5

It is observed that the solution found is very very close to the solution obtained in [3] with however a slightly lower cost function value. It is worth noting that the best solution found in [3] was obtained in 30 runs with 24000 function evaluations per run.

5.7 Example 7

In this last example, we consider the optimal design of a spiral inductor on silicon. The problem to solve can be formulated as follows [7]:

$$\begin{aligned} \text{minimize} \quad & Q^{-1} \\ \text{subject to} \quad & L_s = L_{req} \\ & \omega_{sr} \geq \omega_{sr,min} \\ & d_{in} + 2n(w+s) \leq d_{out} \\ & s \geq s_{min},\ w \geq w_{min} \\ & d_{in} \geq d_{in,min},\ d_{out} \leq d_{out,max} \end{aligned} \tag{21}$$

where L_{req} id the desired inductance value, L is the inductance expression which depends uppon the geometry of the inductor, namely: the number of turns n, the turn width w, the turn spacing s, the inner diameter d_{in} and the outer diameter d_{out}. These parameters are typically the design variables of the inductor. The quantities Q and ω_{sr} represents, respectively, the quality factor and the self-resonance frequency:

$$Q = \frac{\omega L_s}{R_s} \frac{R_p \left[1 - \frac{R_s^2(C_s+C_p)}{L_s} - \omega^2 L_s(C_s+C_p)\right]}{R_p + \left[\left(\frac{\omega L_s}{R_s}\right)^2 + 1\right] R_s}, \quad \omega_{sr} = \sqrt{\frac{1 - \frac{R_s^2(C_s+C_p)}{L_s}}{L_s(C_s+C_p)}}$$

The inductance L_s, and the resistances and capacitances R_s, C_s, R_p, C_p are defined as follows:

$$\begin{aligned} & L_s = k_1 n^2 z(d_{in}, d_{out}),\ \ R_s = k_2 n(d_{in}+d_{out})/w,\ \ C_s = k_3 n w^2 \\ & R_p = 2k_7/(nw(d_{in}+d_{out})),\ \ C_p = (k_8+k_9)nw(d_{in}+d_{out})/2 \end{aligned} \tag{22}$$

The function $z(d_{in}, d_{out})$ and the constants k_1, k_2, k_3, k_7, k_8 and k_9 are given by:

$$\begin{aligned} & z(d_{in}, d_{out}) = c_1(\ln(c_2/r) + c_3 r + c_4 r^2),\ \ r = (d_{out}-d_{in})/(d_{out}+d_{in}) \\ & k_1 = 2\pi 10^{-7},\ k_2 = \eta\rho/(d(1-e^{-t/\delta})),\ \eta = c_5 \tan(\pi/c_5),\ \delta = \sqrt{5\times 10^6 \rho/(\pi\omega)} \\ & k_3 = \epsilon_{ox}/t_{ox,M_1-M_2},\ \ k_4 = \eta\epsilon_{ox}/(2t_{ox}),\ \ k_5 = \eta C_{sub}/2,\ \ k_6 = 2/(\eta G_{sub}) \\ & k_7 = 1/(\omega^2 k_4^2 k_6) + k_6(k_4+k_5)^2/k_4^2,\ \ k_8 = k_4/(1+\omega^2(k_4+k_5)^2 k_6^2) \\ & k_9 = k_4\omega^2(k_4+k_5)k_5 k_6^2/(1+\omega^2(k_4+k_5)^2 k_6^2) \end{aligned}$$

where the parameters c_1, c_2, c_3, c_4, c_5 depend upon the shape of the inductor (square, hexagonal, octagonal or circular); the parameters ρ, t, ϵ_{ox}, t_{ox}, t_{ox,M_1-M_2}, C_{sub}, G_{sub} are technology dependent, and ω is the working frequency of the inductor.

Problem (21) can be formulated as a QGP problem. Indeed, after some basic manipulations we get the following equivalent problem:

$$\begin{array}{ll} \text{minimize} & q^{-1} \\ \text{subject to} & \frac{qR_s}{\omega L_s R_p}\left(\frac{\omega^2 L_s^2}{R_s}+R_s+R_p\right)+(C_s+C_p)\left(\frac{R_s^2}{L_s}+\omega^2 L_s\right)\leq 1 \\ & L_s = L_{req} \\ & \omega_{sr,min}^2 L_s(C_s+C_p)+R_s^2(C_s+C_p)/L_s \leq 1 \\ & d_{in}+2n(w+s)\leq d_{out} \\ & s\geq s_{min},\ w\geq w_{min} \\ & d_{in}\geq d_{in,min},\ d_{out}\leq d_{out,max} \end{array} \tag{23}$$

where q is an additional variable, L_s, R_s, C_s, R_p and C_p are given by (22). Thus formulated, the problem (23) is QGP in the design variables d_{in} and d_{out} and so can be efficiently solved using the approach described in section 3.2.

Problem (23) has been solved using the Nelder-Mead simplex method based QGP (NMSM-QGP), the results thus obtained were then compared to those obtained using a standard genetic algorithm (GA). In our experiments, the following parameters have been used:

$$\begin{array}{l} c_1 = 1.27,\ c_2 = 2.07,\ c_3 = 0.18,\ c_4 = 0.13,\ c_5 = 4,\ \rho = 2\times 10^{-8}\Omega\text{m} \\ t = 10^{-6}\text{m},\ \omega = 3\pi\times 10^9\text{rad/s},\ \epsilon_{ox} = 3.45\times 10^{-11}\text{F/m},\ t_{ox} = 4.5\times 10^{-6}\text{m} \\ t_{ox,M_1-M_2} = 1.3\times 10^{-6}\text{m},\ C_{sub} = 1.6\times 10^{-6}\text{F/m}^2,\ G_{sub} = 4\times 10^4\text{S/m}^2 \\ s_{min} = w_{min} = 1.9\times 10^{-6}\text{m},\ d_{in,min} = 10^{-4}\text{m},\ d_{out,max} = 4\times 10^{-4}\text{m} \\ \omega_{sr,min} = 8\pi\times 10^9\text{rad/s},\ L_{req} = 30\times 10^{-9}\text{H}. \end{array}$$

The solutions found via NMSM-QGP and GA are presented in Table 7. As we can see, the result obtained using NMSM-QGP is significantly better than the solution found by GA.

Table 7: Comparison of the solutions found via GA (with $N = 200$, $N_G = 150$, $p_c = 0.7$, $p_m = 0.07$) and NMSM-QGP (with the starting point ($d_{in} = 200$, $d_{out} = 300$).

	n	w	s	d_{in}	d_{out}	L_s	Q	Nb of calls
GA	9.440	4.491	3.73	147.603	309.069	29.99	2.821	30000
NMSM-QGP	10.862	3.683	1.90	111.54	232.824	30.00	3.233	47

6 Conclusion

In this paper an important extension of standard geometric programming (GP), called quasi geometric programming (QGP) problems, was introduced. The consideration of this kind of problems is motivated by the fact that many engineering problems can be formulated, or well approximated, as a QGP. Thus, the problem of solving a given QGP appears of great practical importance. However, the resolution of a QGP is difficult due to its non-convex nature. The main contribution of this paper was to show that a given QGP can be efficiently solved via GP which represent exactly the original problem when some variables are held constant. In addition, the proposed approach does not require the development of new solvers and works well with any existing solver that are able to solve convex problems. This feature is important for time saving reasons. Numerical applications have shown that the results obtained by applying the proposed method are often better than those obtained via any other approaches. Moreover, if the QGP problem is feasible, our approach never fails to find out, in a reasonable time, a very good suboptimal solution. This is not so surprising since we do not solve the QGP problem in a blind manner. On the contrary, the proposed approach takes into account the particular structure of the problem to be solved. Indeed, QGP becomes a standard GP when some variables are kept constant. This important property has been exploited for efficiently solving the considered problem. In fact, the main difficulty we have encountered is to recognize if a given problem is QGP or not. Indeed, very often this is not obvious at the first glance and some mathematical manipulations and/or transformations must be done to see if the underlying problem is

QGP.

Appendix

A1. derivation of problem (9)

The quasi geometric problem (7), can be rewritten as follows:

$$\begin{aligned} \underset{\lambda,\, x,\, \xi}{\text{minimize}} \quad & \lambda^{-1} \\ \text{subject to} \quad & \lambda + \varphi_0(x,\xi) \leq \frac{\mathcal{N}(Q_0(\xi))}{\mathcal{D}(Q_0(\xi))} \\ & \varphi_i(x,\xi) \leq \frac{\mathcal{N}(Q_i(\xi))}{\mathcal{D}(Q_i(\xi))}, \quad i = 1, \cdots, m \\ & h_j(x,\xi) = \frac{\mathcal{N}(Q'_j(\xi))}{\mathcal{D}(Q'_j(\xi))}, \quad j = 1, \cdots, p \end{aligned} \tag{24}$$

where $\mathcal{N}(.)$ and $\mathcal{D}(.)$ are, respectively, the numerator and the denominator of the rational posynomial function passed in argument. Since $\mathcal{D}(Q_0(\xi) > 0$, $\mathcal{D}(Q_i(\xi) > 0$ $i = 1, \cdots, m$ and $\mathcal{D}(Q'_j(\xi) > 0$ $j = 1, \cdots, p$, formulation (24) is equivalent to the following optimization problem:

$$\begin{aligned} \underset{\lambda,\, x,\, \xi}{\text{minimize}} \quad & \lambda^{-1} \\ \text{subject to} \quad & (\lambda + \varphi_0(x,\xi))\mathcal{D}(Q_0(\xi)) \leq \mathcal{N}(Q_0(\xi)) \\ & \varphi_i(x,\xi)\mathcal{D}(Q_i(\xi)) \leq \mathcal{N}(Q_i(\xi)), \quad i = 1, \cdots, m \\ & h_j(x,\xi)\mathcal{D}(Q'_j(\xi)) \leq \mathcal{N}(Q'_j(\xi)) \\ & h_j(x,\xi)\mathcal{D}(Q'_j(\xi)) \geq \mathcal{N}(Q'_j(\xi)), \quad j = 1, \cdots, p \end{aligned} \tag{25}$$

Let $\mathcal{S}$ be the set of feasible solutions of (25), i.e. the set of decision variables satisfying the constraints. This set is not convex and thus the underlying optimization problem is very hard to solve. However, problem (25) can be efficiently solved over a convex subset of $\mathcal{S}$. To do that, we can use the optimal lower monomial approximation of a posynomial (see appendix A2). We denote by $\Gamma(p(x))$ the optimal lower monomial approximation of the posynomial $p(x)$, i.e. we have: $\Gamma(p(x)) \leq p(x)$ and the difference $p(x) - \Gamma(p(x))$ is the

smallest possible. This led to the following convex optimization problem[10]:

$$\begin{aligned}
\underset{\lambda,\,x,\,\xi}{\text{minimize}} \quad & \lambda^{-1} \\
\text{subject to} \quad & \frac{(\lambda+\varphi_0(x,\xi))\mathcal{D}(Q_0(\xi))}{\Gamma(\mathcal{N}(Q_0(\xi)))} \leq 1 \\
& \frac{\varphi_i(x,\xi)\mathcal{D}(Q_i(\xi))}{\Gamma(\mathcal{N}(Q_i(\xi)))} \leq 1, \quad i=1,\cdots,m \\
& \frac{h_j(x,\xi)\mathcal{D}(Q'_j(\xi))}{\Gamma(\mathcal{N}(Q'_j(\xi)))} \leq 1 \\
& \frac{\mathcal{N}(Q'_j(\xi))}{h_j(x,\xi)\Gamma(\mathcal{D}(Q'_j(\xi)))} \leq 1, \quad j=1,\cdots,p
\end{aligned} \tag{26}$$

This standard GP is convex under a log transformation and since

$$\begin{aligned}
&\Gamma(\mathcal{N}(Q_0(\xi))) \leq \mathcal{N}(Q_0(\xi)) \\
&\Gamma(\mathcal{N}(Q_i(\xi))) \leq \mathcal{N}(Q_i(\xi)), \quad i=1,\cdots,m \\
&\Gamma(\mathcal{N}(Q'_j(\xi))) \leq \mathcal{N}(Q'_j(\xi)) \\
&h_j(x,\xi)\Gamma(\mathcal{D}(Q'_j(\xi))) \leq h_j(x,\xi)\mathcal{D}(Q'_j(\xi)), \quad j=1,\cdots,p
\end{aligned} \tag{27}$$

the set of feasible solutions of (26) is a convex subset of $\mathcal{S}$. The global optimum of problem (26) is then, at least, a suboptimal solution of problem (24).

A2. Optimal lower (upper) monomial approximation and optimal ϵ-approximation

Consider a positive function $f : \mathcal{X} \subset \mathbf{R}^n_{++} \rightarrow \mathbf{R}$ not necessarily in posynomial form. The objective is to find a monomial:

$$\Gamma(f(x)) = cx_1^{a_1}\cdots x_n^{a_n} \tag{28}$$

satisfying:

$$\Gamma(f(x)) \leq f(x), \ \forall x \in \mathcal{X} \tag{29}$$

and in addition, it is required that $\Gamma(f(x))$ must be close as possible to $f(x)$. In the least-squares sense, this problem is solution of a nonlinear constrained 2-norm minimization problem:

$$\begin{aligned}
\text{minimize} \quad & \|f(x)-\Gamma(f(x))\|_2^2 \\
\text{subject to} \quad & \Gamma(f(x)) \leq f(x) \\
& x=(x_1,\cdots,x_n) \in \mathcal{X}
\end{aligned} \tag{30}$$

[10]Strictly speaking this problem is not convex as it is, however, using the log transformation presented in section 2.2 we get a convex formulation.

A good approximate solution to this problem can be found by using data points:

$$(x^{(i)},\, f(x^{(i)})), \quad i = 1, \cdots N \tag{31}$$

drawn within $\mathcal{X}$ according to a uniform probability distribution. Indeed, taking the log of the monomial (28) and the log of the positive function f gives respectively:

$$L = \begin{bmatrix} 1 & \log(x_1^{(1)}) & \cdots & \log(x_n^{(1)}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \log(x_1^{(N)}) & \cdots & \log(x_n^{(N)}) \end{bmatrix}, \quad p = \begin{bmatrix} \log c \\ a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad q = \begin{bmatrix} \log(f(x^{(1)})) \\ \vdots \\ \log(f(x^{(N)})) \end{bmatrix} \tag{32}$$

We want to determine p so that[11] $Lp \preceq q$ with Lp as close as possible to q. This can be done by solving the following 2-norm constrained problem:

$$\begin{aligned} &\text{minimize} && \|Lp - q\|_2^2 \\ &\text{subject to} && Lp \preceq q \end{aligned} \tag{33}$$

This is a convex optimization problem which can be solved using available solver like for instance cvx. In the same way, the optimal upper monomial approximation can be obtained by solving the convex problem:

$$\begin{aligned} &\text{minimize} && \|Lp - q\|_2^2 \\ &\text{subject to} && Lp \succeq q \end{aligned} \tag{34}$$

The main difficulty is how to determine the minimum number of samples N required so that $\Gamma(f(x)) \leq f(x)$ is satisfied for all $x \in \mathcal{X}$ with high probability. It can be shown (see [16]) that, given two positive numbers: ρ close to 1 (e.g. 0.999) and e close to zero (e.g. 10-3), if the number of samples satisfies $N \geq \frac{\log(1-\rho)}{\log(1-e)}$, then inequality $\Gamma(f(x)) \leq f(x)$ is satisfied for all $x \in \mathcal{X}$ with a probability at least ρ, except possibly for some x belonging to a set of measure no larger than e. In our experiments we have used $\rho = 0.999$ and $e = 6 \times 10^{-4}$ which gives $N \geq 11510$, thus the lower (or upper) monomials approximation are satisfied with a probability at least 0.999.

The lower-upper monomial approximation principle can be combined to a form that we call the ϵ-approximation of a positive function by monomials. In this case, the objective is to find a monomial (28) satisfying:

$$\Gamma_\epsilon(f(x)) - \epsilon \leq f(x) \leq \Gamma_\epsilon(f(x)) + \epsilon, \ \forall x \in \mathcal{X} \tag{35}$$

in other words, we want to approximate the function $f(x)$ by a monomial $\Gamma_\epsilon(f(x))$ with a given accuracy ϵ. In the least-squares sense, this problem is solution of a nonlinear

[11] The notation $\preceq$ means a componentwise inequality.

constrained 2-norm minimization problem:

$$\begin{aligned} \text{minimize} \quad & \|f(x) - \Gamma_\epsilon(f(x))\|_2^2 \\ \text{subject to} \quad & |\Gamma_\epsilon(f(x)) - f(x)| \leq \epsilon \\ & x = (x_1, \cdots, x_n) \in \mathcal{X} \end{aligned} \tag{36}$$

However, for small values of ϵ this problem may be infeasible. To make the problem feasible for every value of ϵ it is necessary to relax the constraints. This can be done by introducing additional variables ε^-, ε^+ that indicate that, occasionally, we accept the constraints violation as long as it does not exceed a certain value which must be as small as possible. This can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \|f(x) - \Gamma_\epsilon(f(x))\|_2^2 + \varepsilon^- + \varepsilon^+ \\ \text{subject to} \quad & \Gamma_\epsilon(f(x)) - f(x) \leq \epsilon + \varepsilon^- \\ & -\Gamma_\epsilon(f(x)) + f(x) \leq \epsilon + \varepsilon^+ \\ & \varepsilon^- \geq 0, \;\; \varepsilon^+ \geq 0, \;\; x = (x_1, \cdots, x_n) \in \mathcal{X} \end{aligned} \tag{37}$$

A good approximate solution to this problem can be found by using data points drawn within $\mathcal{X}$ according to a uniform probability distribution. In these conditions, problem (37) is formulated as follows[12]:

$$\begin{aligned} \text{minimize} \quad & \|Lp - q\|_2^2 + \|\varepsilon^-\|_2^2 + \|\varepsilon^+\|_2^2 \\ \text{subject to} \quad & Lp - q \preceq \mathbf{1}\epsilon + \varepsilon^- \\ & -Lp + q \preceq \mathbf{1}\epsilon + \varepsilon^+ \\ & \varepsilon^- \succeq 0, \;\; \varepsilon^+ \succeq 0 \end{aligned} \tag{38}$$

where L, p and q are defined as in (32), $\mathbf{1} = (1, 1, \cdots, 1)$, ε^-, ε^+ are vectors, and $\preceq$, $\succeq$ represents componentwise inequalities. This is a convex optimization problem which can be solved using available solver like for instance cvx.

[12]Note that this formulation is closely related to the so called SVM-regression, see for instance [14].

References

[1] S. Boyd, S.-J. Kim, L. Vandenberghe & A. Hassibi. A Tutorial on Geometric Programming. *Optimization and Engineering*, vol. 8(1), pp. 67-127, 2007.

[2] C.A.C Coello & E.M. Montes. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, Vol. 16, pp. 193-203, 2002.

[3] L. C. Cagnina, S. C. Esquivel & C. A. Coello Coello. Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer. *Informatica*, Vol. 32, pp. 319-326, 2008.

[4] M. Grant & S. Boyd. CVX: Matlab Software for Disciplined Convex Programming, version 1.21. http://cvxr.com/cvx, 2010.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Kluwer Academic Publishers, Boston, MA, 1989.

[6] Q. He & L. Wang. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, Vol. 20, pp. 89-99, 2007.

[7] M. Hershenson, S. Mohan, S. Boyd, & T. Lee. Optimization of Inductor Circuits via Geometric Programming. *Proceedings of IEEE Design Automation Conference*, pp. 994-998, 1999.

[8] C. T. Kelley. *Iterative Methods for Optimization.* SIAM Frontiers in Applied Mathematics, N° 18, 1999.

[9] J. Kennedy, & R. C. Eberhart. Particle swarm optimization. *In Proceedings of IEEE International Conference on Neural Networks* pp. 1942-1948, Piscataway, NJ, USA, IEEE Press, 1995.

[10] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi. Optimization by Simulated Annealing. *Science*, Vol. 220(4598), pp. 671-680, 1983.

[11] S. J. Qu, K. C. Zhang & Y. Ji. A new global optimization algorithm for signomial geometric programming via Lagrangian relaxation. *Applied Mathematics and Computation*, Vol. 184, pp. 886-894, 2007.

[12] S. S. Rao. *Engineering Optimization.* Wiley, New York, 1996.

[13] K. Sedlaczek & P. Eberhard. Using augmented Lagrangian particle swarm optimization for constrained problems in engineering. *Structural and Multidisciplinary Optimization*, Vol. 32, pp. 277-286, 2006.

[14] A. J. Smola & B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, Vol. 14, pp. 199-222, 2004.

[15] R. Toscano & P. Lyonnet. Heuristic Kalman Algorithm for solving optimization problems. *IEEE Transaction on Systems, Man, and Cybernetics, Part B*, Vol. 35, pp. 1231-1244, 2009.

[16] R. Toscano. $\mathcal{H}_2/\mathcal{H}_\infty$ Robust static output feedback control design without solving linear matrix inequalities. *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 129, pp. 860-866, 2007.

[17] Y. Wang & Z. Liang. A deterministic global optimization algorithm for generalized geometric programming. *Appl. Math. Comput.*, Vol. 168, pp. 722-737, 2005.