EVOLUTIONARY ALGORITHMS FOR SOLVING QUASI GEOMETRIC PROGRAMMING PROBLEMS

R. Toscano and P. Lyonnet
Université de Lyon, Laboratoire de Tribologie et de Dynamique des Systèmes, CNRS UMR5513 ECL/ENISE
58 rue Jean Parot 42023 Saint-Etienne cedex 2, France

Keywords: Geometric Programming (GP), Quasi Geometric Programming (QGP), Evolutionary Algorithm (EA), Interior point method.

Abstract: In this paper we introduce an extension of standard geometric programming (GP) problems which we call quasi geometric programming (QGP) problems. 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 as a QGP. However, solving a QGP remains a difficult task due to its intrinsic non-convex nature. This is why we investigate the possibility of using evolutionary algorithms (EA) for solving a QGP problem. The main idea developed in this paper is to combine evolutionary algorithms with interior point method for efficiently solving QGP problems. An interesting feature of the proposed approach is that it does not need to develop specific program solver and works well with any existing EA and available solver able to solve conventional GP. Some considerations on the robustness issue are also presented. Numerical experiments are used to validate the proposed method.

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 (Boyd et al., 2007).

In this paper we introduce an important extension of GP which we call quasi geometric programming (QGP) problems. The motivation of introducing this particular kind of nonlinear program comes from the fact that a lot of engineering problems can be formulated as a QGP. Thus, solving a given QGP appears of great practical importance.

A major difficulty is that QGP is a nonlinear non-convex and possibly non smooth optimization problem. This is a major source of computational intractability and conservatism (Rockafellar, 1993), (Boyd and Vandenberghe, 2004). Indeed, unlike GP problems, QGP 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 QGP can be computed efficiently[1].

On the other hand, evolutionary algorithms (EAs) have demonstrated a strong ability to deal with non-convex optimization problems. Indeed, EAs maintain several solutions simultaneously, which provide a significantly better foundation for escaping from local optima. In addition, solutions are not necessarily created as neighbors of existing solutions and thus, this increases the probability of finding the global optimum. Over the years, a lot of algorithms have been suggested, however the basic principles remain the same and a good overview can be found in (Back and Schwefel., 1993), (Michalewicz and Schoenauer., 1996).

The main idea developed in this paper is to associate the interior point method, able to solve very efficiently a GP problem, with the ability of evolutionary

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

algorithms of dealing with non-convex optimization problems. This is possible because a QGP becomes a GP when some variables are kept constant. The interesting thing is that the proposed approach does not need to develop specific program solver and works well with any existing evolutionary algorithm (for instance a standard genetic algorithm) and available solver able to solve conventional geometric programs (for instance cvx (Grant and Boyd, 2010)). From a practical point of view this is very interesting because the engineers often have not so much time to develop specific algorithm for solving particular 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 EAs and GP 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 some optimization problems are considered to illustrate the validity of the proposed method for solving a QGP problem. We give concluding remarks in section 6.

2 GEOMETRIC PROGRAMMING

GP is a special type of nonlinear, non-convex optimisation problems. An 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 function $f : \mathbf{R}^n_{++} \to \mathbf{R}$ defined by[2]:

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

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:

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

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

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

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

where f_i, $i = 0, \cdots, m$, are posynomials and g_i, $i = 1, \cdots, p$, are monomials.

2.2 Convex Formulation

GP in standard form is not a convex optimisation problem[3], 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:

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

where $\underline{i} = 1, \cdots, m$ and $j = 1, \cdots, p$. 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 able to give if not the optimum, at least a good near-optimum. In this spirit, we are now ready to introduce the concept of quasi geometric programming.

[3]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

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

where the vector $z \in \mathbf{R}^n_{++}$ include all the optimization variables, $f_0 : \mathbf{R}^n_{++} \rightarrow \mathbf{R}$ is the objective function or cost function, $f_i : \mathbf{R}^n_{++} \rightarrow \mathbf{R}$ are the inequality constraint functions and $g_j : \mathbf{R}^n_{++} \rightarrow \mathbf{R}$ 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:

$$\begin{array}{ll} \text{minimize} & \varphi_0(x,\xi) - \varphi'_0(\xi) \\ \text{subject to} & \varphi_i(x,\xi) \leqslant Q_i(\xi), \quad i = 1, \cdots, m \\ & h_j(x,\xi) = Q'_j(\xi), \quad j = 1, \cdots, p \end{array} \tag{7}$$

where $x \in \mathbf{R}^{n_x}_{++}$ and $\xi \in \mathbf{R}^{n_\xi}_{++}$ with $n_x + n_\xi = n$, are sub-vector of the optimization variable $z \in \mathbf{R}^n_{++}$. The functions $\varphi_i(x,\xi)$, $i = 0, \cdots, m$ are posynomials and $h_j(x,\xi)$, $j = 1, \cdots, p$ are monomials. The only particular assumption made about the functions $\varphi'_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$, is that they are positives. Except for their positivity, no other particular assumption is made; these functions can be even non-smooth.

The QGP (7) can be reformulated as follows:

$$\begin{array}{ll} \text{maximize} & \lambda \\ \text{subject to} & \lambda \leqslant -\varphi_0(x,\xi) + \varphi'_0(\xi) \\ & \varphi_i(x,\xi) \leqslant Q_i(\xi), \quad i = 1, \cdots, m \\ & h_j(x,\xi) = Q'_j(\xi), \quad j = 1, \cdots, p \end{array} \tag{8}$$

where $\lambda \in \mathbf{R}_{++}$ is an additional decision variable. It is important to insist on the fact that the problem (7), or equivalently (8), 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 GP solvers. Although specific algorithms can be designed to find out a sub optimal solution to problem (8), we think that it could be very interesting solving these problems by using available EA and standard GP solvers. Indeed, this could be interesting for at least two reasons. Firstly the ability of solving problem (8) using available GP solvers allows time saving; the development of a specific algorithm is always a long process and in an industrial context of great concurrency there is often no time to do that. Secondly, the available GP solvers like for instance cvx are very easy to use and, which is most important, are very very efficient. 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 approach presented here after. Indeed, this method does not require the development of particular algorithms and is based on the use of available EA and GP solvers.

The QGP (8) is not at all easy to solve when $\varphi'_0(\xi)$, $Q_i(\xi)$ and $Q'_j(\xi)$ have no particular form. In this case indeed, the problem is intrinsically non-convex, and thus, in general, there is no obvious transformation allowing to solve (8) via the resolution of a sequence of standard GP. To solve this kind of problem, we can see the QGP (7), or equivalently (8), as a function of ξ, denoted $J(\xi)$, that we want to maximize:

$$\begin{array}{ll} \text{maximize} & J(\xi) \\ \text{subject to} & \underline{\xi} \leqslant \xi \leqslant \overline{\xi} \end{array} \tag{9}$$

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

$$\begin{array}{ll} J(\xi) = & \max\limits_{x,\lambda} \lambda \\ \text{subject to} & \dfrac{\lambda + \varphi_0(x,\xi)}{\varphi'_0(\xi)} \leqslant 1 \\ & \dfrac{\varphi_i(x,\xi)}{Q_i(\xi)} \leqslant 1, \quad i = 1, \cdots, m \\ & \dfrac{h_j(x,\xi)}{Q'_j(\xi)} = 1, \quad j = 1, \cdots, p \end{array} \tag{10}$$

Problem (9) is a non-convex unconstrained optimization problem[4] and can be solved using evolutionary algorithms[5] like, for instance, a standard genetic algorithm (GA). The code associated to this kind of algorithms is easily available and thus don't need to be programmed.

When ξ is kept constant, problem (10) is a standard GP which can be solved very efficiently using available GP solvers.

This suggests that we can solve the QGP problem (8) with a two levels procedure. At the first level, the chosen EA (eg. the standard GA) is used to select values of ξ within the bounds. For each value of ξ, the standard GP (10) is solved using available solvers. As we can see, (10) is our fitness function which is evaluated by solving a standard GP problem. This procedure is continued until some stopping rule is satisfied. The suggested procedure is formalized more precisely in the following algorithm.

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

[5]Note that EAs does not require the knowledge of the derivatives of the objective function. Thus, smoothness is not required.

Algorithm for Solving a QGP Problem (EA-QGP)

1. *Set* $k := 0$, `best` $:= -$`inf`, $\xi_{opt} := -1$ *and* $x_{opt} := -1$ *or other infeasible values.*

2. *Using an EA, generate a population* $P(k) = \left\{\xi_i^{(k)}\right\}_{i=1}^{i=I}$*, such that* $\underline{\xi} \leqslant \xi_i^{(k)} \leqslant \overline{\xi}$*, for all individuals i; I is the size of the population.*

3. *For each individual* $\xi_i^{(k)}$ *solve the standard GP problem (10) w.r.t* λ *and x. This gives, w.r.t* $\xi_i^{(k)}$*, the optimal solution denoted* $\lambda_i^{(k)}$ *and* $x_i^{(k)}$*. This step represents the evaluation of the population* $P(k)$*. If problem (10) is not feasible, then set* $J(\xi^{(k)}) := -$`inf` *else set* $J(\xi^{(k)}) := \lambda^{(k)}$*.*

4. *If* $J(\xi_b^{(k)}) >$ `best`*, then set:* `best` $:= J(\xi_b^{(k)})$, $\xi_{opt} := \xi_b^{(k)}$ *and* $x_{opt} := x_b^{(k)}$*, where* $\xi_b^{(k)}$ *represents the best individual of the population* $P(k)$ *and* $x_b^{(k)}$ *the solution to the corresponding GP problem.*

5. *If the termination condition is satisfied, go to step 7 (the termination condition can be, for instance, a defined number of iterations).*

6. *From the results obtained step 3, generate a new population* $P(k)$ *where* $k := k+1$ *(this is done by using the usual operators of EA, i.e. selection, crossover and mutation operators), go to step 3.*

7. *The optimal solution is given by* (x_{opt}, ξ_{opt})*, stop.*

In this algorithm, `inf` represents the IEEE arithmetic representation for positive infinity, and `best` is a variable containing the current best objective function. Note that the use of " global optimization methods " like, for instance GA, increases the probability of finding a global optimum but this is not guaranteed, except perhaps if the search space of problem (9) is explored very finely, but this cannot be done in a reasonable time.

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 lie in a bounded set Θ defined as follows:

$$\Theta = \left\{\theta \in \mathbf{R}^q : \underline{\theta} \preceq \theta \preceq \overline{\theta}\right\}, \qquad (11)$$

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 $\underline{\theta} = [\underline{\theta}_1 \cdots \underline{\theta}_q]^T$, $\overline{\theta} = [\overline{\theta}_1 \cdots \overline{\theta}_q]^T$ 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:

$$\begin{array}{ll} \text{maximize} & \lambda \\ \text{subject to} & \lambda + \varphi_0(x,\xi,\theta) \leqslant \varphi_0'(\xi,\theta) \\ & \varphi_i(x,\xi,\theta) \leqslant Q_i(\xi,\theta), \quad i = 1,\cdots,m \\ & h_j(x,\xi,\theta) = Q_j'(\xi,\theta), \quad j = 1,\cdots,p \end{array} \qquad (12)$$

for all $\theta \in \Theta$. 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. The functions φ_0', Q_i and Q_j' are only assumed to be positive for each θ.

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 can be imposed by the problem itself or can be obtained by sampling the continuous set Θ defined in (11). For instance, we might sample each interval $[\underline{\theta}_i,\ \overline{\theta}_i]$ with three values: $\underline{\theta}_i$, $\frac{\underline{\theta}_i+\overline{\theta}_i}{2}$ and $\overline{\theta}_i$, 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, ξ) that satisfy the QGP problem for all possible vector parameters. To do so, 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:

$$\begin{array}{ll} \text{maximize} & \lambda \\ \text{subject to} & \lambda + \varphi_0(x,\xi,\theta^{(k)}) \leqslant \varphi_0'(\xi,\theta^{(k)}) \\ & \varphi_i(x,\xi,\theta^{(k)}) \leqslant Q_i(\xi,\theta^{(k)}) \\ & h_j(x,\xi,\theta^{(k)}) = Q_j'(\xi,\theta^{(k)}) \end{array} \qquad (13)$$

where $i = 1,\cdots,m$ and $j = 1,\cdots,p$ and $k = 1,\cdots,N$. As we can see, problem (13) can be solved as a standard QGP problem using the method presented in section 3.

5 NUMERICAL EXAMPLES

In this section we illustrate the applicability of the proposed method through three numerical examples. In these examples, the EA-QGP has been implemented using the GA toolbox (Chipperfield et al., 1995) and the GP solver cvx (Grant and Boyd, 2010). The following parameters were used: binary coded-GA 16 bits, number of generations 30 (used as stopping rule), population size 20, roulette wheel selection, one point crossover with probability of 0.7 and a probability of mutation 0.07. The number of generations has been chosen deliberately small to avoid a too long computation time. Indeed, the time cost for a GP-solver call (in the three examples this time cost is about 0.5s) is generally higher than the time cost of the objective function. The price to pay is that the solution thus found is not necessarily the best possible.

5.1 Example 1

This first example is borrowed from (Qu et al., 2007) in which the problem was solved using a global optimization algorithm via lagrangian relaxation (GDCAB).

$$\begin{array}{ll} \text{min.} & 0.5t_1t_2^{-1} - t_1 - 5t_2^{-1} \\ \text{s. t.} & 0.01t_2t_3^{-1} + 0.01t_2 + 0.0005t_1t_3 \leqslant 1 \\ & 70 \leqslant t_1 \leqslant 150,\ 1 \leqslant t_2 \leqslant 30,\ 0.5 \leqslant t_3 \leqslant 21 \end{array}$$

This problem can be rewritten as follows:

$$\begin{array}{ll} \text{max.} & \lambda \\ \text{s. t.} & \dfrac{\lambda + 0.5t_1t_2^{-1}}{t_1 + 5t_2^{-1}} \leqslant 1 \\ & 0.01t_2t_3^{-1} + 0.01t_2 + 0.0005t_1t_3 \leqslant 1 \\ & 70 \leqslant t_1 \leqslant 150,\ 1 \leqslant t_2 \leqslant 30,\ 0.5 \leqslant t_3 \leqslant 21 \end{array}$$

which is QGP in (t_1, t_2). The method EA-QGP described section 3, was applied to solve this optimization problem and the solution found is presented in Table 1. It can be seen that the solution found using EA-QGP is very significantly better than that found using the method described in (Qu et al., 2007).

Note also that the Number of iterations (in fact, the number of GP-solver call) is small compared to the number of iterations required by GDCAB. However, recall that the time cost of a GP-solver call is generally higher than the computation time of the objective function.

Table 1: Comparison of the solution found by EA-QGP and GDCAB see (Qu et al., 2007)

	GDCAB	EA-QGP
t_1	88.6274	149.9999
t_2	7.9621	18.9558
t_3	1.3215	1.6973
objective function	-83.6898	-146.3064
NbIter	1754	600

5.2 Example 2

This second example is borrowed from (He and Wang., 2007) in which the problem was solved using a co-evolutionary particle swarm optimization (CPSO). In this problem, the objective is to minimize the total cost including the cost of the material, forming and welding of a cylindrical vessel.

$$\begin{array}{ll} \text{min.} & 0.6224x_1x_3x_4 + 1.7778x_2x_3^2 \\ & +3.1661x_1^2x4 + 19.84x_1^2x_3 \\ \text{s. t.} & -x_1 + 0.0193x_3 \leqslant 0 \\ & -x_2 + 0.009543x_3 \leqslant 0 \\ & -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leqslant 0 \\ & 1 \leqslant x_1, x_2 \leqslant 99,\ 10 \leqslant x_3, x_4 \leqslant 200 \end{array}$$

This problem can be reformulated as follows:

$$\begin{array}{ll} \text{min.} & 0.6224x_1x_3x_4 + 1.7778x_2x_3^2 \\ & +3.1661x_1^2x4 + 19.84x_1^2x_3 \\ \text{s. t.} & -0.0193x_3/x_1 \leqslant 1 \\ & 0.009543x_3/x_2 \leqslant 1 \\ & \dfrac{1296000}{\pi x_3^3(z+4/3)} \leqslant 1 \\ & \dfrac{x_4}{x_3z} = 1, \quad \dfrac{1}{20} \leqslant z \leqslant 20 \\ & 1 \leqslant x_1, x_2 \leqslant 99,\ 10 \leqslant x_3, x_4 \leqslant 200 \end{array}$$

which is QGP in z. The solution found using the EA-QGP method is presented in Table 2.

Table 2: Comparison of the solution found by EA-QGP and CPS0 see (He and Wang., 2007)

	CPSO	EA-QGP
x_1	0.8125	0.7782
x_2	0.4375	0.3846
x_3	42.0913	40.3197
x_4	176.7465	199.9993
objective function	6061.0777	5885.3336
NbFuncEval	32500	600

From Table 2, it can be seen that the solution found using EA-QGP is significantly better than that found using the method described in (He and Wang., 2007). Regarding the number of function evaluations (NbFuncEval) the same remark as in example 1 applies.

5.3 Example 3

This third and last example is borrowed from (Cagnina et al., 2008) in which the problem was solved using a constrained particle swarm optimizer (SiC-PSO). 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 were formulated as follows:

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

To apply the proposed approach, this problem can be rewritten into the following form:

$$\begin{aligned}
\text{max. } & \lambda \\
\text{s. t. } & \lambda+\varphi_0\leqslant\varphi_0' \\
& \frac{27}{x_1x_2^2x_3}\leqslant 1, \quad \frac{397.5}{x_1x_2^2x_3^2}\leqslant 1 \\
& \frac{1.93x_4^3}{x_2x_3x_6^4}\leqslant 1, \quad \frac{1.93x_5^3}{x_2x_3x_7^4}\leqslant 1 \\
& \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2+16.9\times 10^6}\leqslant 1 \\
& \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2+157.5\times 10^6}\leqslant 1 \\
& \frac{x_2x_3}{40}\leqslant 1, \quad \frac{5x_2}{x_1}\leqslant 1, \quad \frac{x_1}{12x_2}\leqslant 1 \\
& \frac{1.5x_6+1.9}{x_4}\leqslant 1, \quad \frac{1.1x_7+1.9}{x_5}\leqslant 1 \\
& 2.6\leqslant x_1\leqslant 3.6,\ 0.7\leqslant x_2\leqslant 0.8,\ 17\leqslant x_3\leqslant 28 \\
& 7.3\leqslant x_4\leqslant 8.3,\ 7.8\leqslant x_5\leqslant 8.3,\ 2.9\leqslant x_6\leqslant 3.9 \\
& 5.0\leqslant x_7\leqslant 5.5
\end{aligned}$$

where φ_0 and φ_0' are defined as follows:

$$\begin{aligned}
\varphi_0 = \; & 0.7854x_1x_2^2x_3(3.3333x_3+14.9334) \\
& +7.4777(x_6^3+x_7^3)+0.7854(x_4x_6^2+x_5x_7^2) \\
\varphi_0' = \; & 33.8456x_1x_2^2+1.508x_1(x_6^2+x_7^2)
\end{aligned}$$

Thus, this equivalent problem is QGP in (x_1,x_2,x_6,x_7). The solution found using the EA-QGP method is presented in Table 3. Note that in spite of the square roots, the two equation are still posynomials. From Table 3, it can be seen that the solution found using EA-QGP is better than that found using the method described in (Cagnina et al., 2008). Here also, regarding the number of function evaluations (NbFuncEval) the same remark as in example 1 applies.

Table 3: Comparison of the solution found by EA-QGP and SiC-PS0 see (Cagnina et al., 2008)

	SiC-PSO	EA-QGP
x_1	3.5000	3.5531
x_2	0.7000	0.6684
x_3	17	17
x_4	7.3000	7.3000
x_5	7.8000	7.8000
x_6	3.3502	3.3509
x_7	5.2867	5.2868
objective function	2996.3481	2876.4999
NbFuncEval	24000	600

6 CONCLUSIONS

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 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 by combining evolutionary algorithms (EA) and interior point methods. In addition, the proposed approach does not need to develop specific program solver and works well with any existing EA and available solver able to solve conventional GP. This feature is important for time saving reasons. Numerical applications have shown that the results obtained by applying the proposed method are better than those obtained via any other approaches. This is not so surprising since we don't use EA 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 suggested a resolution method including two levels. At the first level, the ability of EA to to deal with non-convex problems is exploited and at the second level, the ability of interior point method for solving a standard GP is used. This two levels structure makes the resolution of a QGP efficient and easy to do by using available EA and standard GP solver. However, the main drawback of the proposed approach is that we have to choose a small number of generations to prevent a too long computation time. This is a serious limitation since the solution thus found is not necessarily the best possible. From this point of view, the proposed approach needs to be improved. One possible way is to generate the initial population from a good approximate solution. This approximate solution can be found using EA in a usual way. In this case, the EA-QGP plays the role of a refinement procedure.

REFERENCES

Back, T. and Schwefel., H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23.

Boyd, S., Kim, S.-J., Vandenberghe, L., and Hassibi., A. (2007). A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67–127.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Cagnina, L. C., Esquivel, S. C., and Coello., C. A. C. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica*, 32(3):319–326.

Chipperfield, A., Fleming, P., Pohlheim, H., and Fonseca, C. (1995). *Genetic Algorithm TOOLBOX For Use with MATLAB*.

Grant, M. and Boyd, S. (2010). *CVX: Matlab Software for Disciplined Convex Programming, version 1.21*. http://cvxr.com/cvx.

He, Q. and Wang., L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Application of Artificial Intelligence*, 20(1):89–99.

Michalewicz, Z. and Schoenauer., M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

Qu, S. J., Zhang, K. C., and Ji., Y. (2007). A new global optimization algorithm for signomial geometric programming via lagrangian relaxation. *Applied Mathematics and Computation*, 184(2):886–894.

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM Review*, 35:183–238.