Neurocomputing 198 (2016) 171–179

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Polynomial time solvable algorithms to a class of unconstrained and linearly constrained binary quadratic programming problems

Shenshen Gu*, Rui Cui, Jiao Peng

School of Mechatronic Engineering and Automation, Shanghai University, 149 Yanchang Road, Shanghai 200072, PR China

ARTICLE INFO

Article history:
Received 1 April 2015
Received in revised form
16 July 2015
Accepted 28 September 2015
Available online 10 March 2016

Keywords:
Binary quadratic programming
Polynomial time solvable algorithm
Dynamic programming

ABSTRACT

Binary quadratic programming (*BQP*) is a typical integer programming problem widely applied in the field of signal processing, economy, management and engineering. However, it is NP-hard and lacks efficient algorithms. Due to this reason, in this paper, some novel polynomial algorithms are proposed to solve a class of unconstrained and linearly constrained binary quadratic programming problems. We first deduce the polynomial time solvable algorithms to the unconstrained binary quadratic programming problems with Q being a seven-diagonal matrix (*UBQP*7) and a five-diagonal matrix (*UBQP*5) respectively with two different approaches. Then, the algorithm to unconstrained problem is combined with the dynamic programming method to solve the linearly constrained binary quadratic programming problem with Q being a five-diagonal matrix (*LCBQP*5). In addition, the polynomial solvable feature of these algorithms is analyzed and some specific examples are presented to illustrate these new algorithms. Lastly, we demonstrate their polynomial feature as well as their high efficiency.

1. Introduction

In this paper, we consider the following binary quadratic programming (*BQP*):

$$\min_{x\in\{0,1\}^n} \frac{1}{2}x^TQx+c^Tx \tag{1}$$

where $Q=(q_{ij})_{n\times n}$ is a symmetric matrix with zero elements in the main diagonal, $c\in\mathbb{R}^n$ is a real vector. As the binary quadratic programming problem, x_i takes only 0 or 1. As a result, there is no loss of generality in assuming the zero diagonal because $x_i^2=x_i(1\le i\le n)$. More specifically, the above problem is also called unconstrained binary quadratic programming (*UBQP*). For problem (1) with linear constraint $a^Tx\le b(a\in\mathbb{Z}_+^n, b\in\mathbb{Z}_+)$, it is called linearly constrained binary quadratic programming (*LCBQP*). *BQP* are typical optimization problems which is well known as NP-hard problem (see [1]).

Many applications of *BQP* problem exist such as signal processing, financial data analysis [2], molecular conformation [3] and cellular radio channel assignment [4]. Various exact solution methods for solving *BQP* and its variants have been proposed in the literature (see, e.g., [4–10] and the references therein). Some important methods include the decomposition method [4], semidefinite relaxations and cutting planes method to improve the quality of the bounds [8], and semidefinite and polyhedral relaxation method [10].

We focus in this paper on a class of polynomially solvable cases of the quadratic binary programming problems. These cases include the unconstrained binary quadratic programming problems with Q being a five-diagonal matrix and a seven-diagonal matrix (denoted by *UBQP*5 and *UBQP*7 respectively), and the linearly constrained binary quadratic programming problems with Q being a five-diagonal matrix (denoted by *LCBQP*5). Identifying polynomially solvable subclasses of problem *BQP* not only offers theoretical insight into the complicated nature of the problem, but also provides useful information for designing efficient algorithms for finding optimal solution to problem *BQP*. More specifically, the properties of the polynomially solvable subclasses provide hints and facilitate the derivation of efficient relaxations for the general form of *BQP*.

The remainder of this paper is organized as follows: In Section 2, the algorithm to problem *UBQP*7 is designed based on the property of matrix Q. In Section 3, the algorithm to problem *UBQP* 5 is designed based on famous basic algorithm [11,12]. In Section 4, the algorithm to solve *LCBQP*5 is proposed by combining the basic algorithm and the dynamic programming method. And in Section 5, we make an analysis on the polynomial feature of these algorithms. Next, some examples are given in Section 6 to illustrate these algorithms. Then, in order to demonstrate the effectiveness and efficiency of these novel algorithms, the simulation experiments are performed in Section 7. And finally, Section 8 concludes this paper.

* Corresponding author.
E-mail address: gushenshen@shu.edu.cn (S. Gu).

http://dx.doi.org/10.1016/j.neucom.2015.09.130
0925-2312/

2. Polynomial solvable algorithm to UBQP7

First, in this section, we consider *U*BQP7 where Q is a seven-diagonal symmetric matrix with zero diagonal elements which takes

$$\begin{pmatrix} 0 & q_{12} & q_{13} & q_{14} & \cdots & 0 & 0 & 0 & 0 \\ q_{12} & 0 & q_{23} & q_{24} & \cdots & 0 & 0 & 0 & 0 \\ q_{13} & q_{23} & 0 & q_{34} & \cdots & 0 & 0 & 0 & 0 \\ q_{14} & q_{24} & q_{34} & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & q_{n-3,n-2} & q_{n-3,n-1} & q_{n-3,n} \\ 0 & 0 & 0 & 0 & \cdots & q_{n-3,n-2} & 0 & q_{n-2,n-1} & q_{n-2,n} \\ 0 & 0 & 0 & 0 & \cdots & q_{n-3,n-1} & q_{n-2,n-1} & 0 & q_{n-1,n} \\ 0 & 0 & 0 & 0 & \cdots & q_{n-3,n} & q_{n-2,n} & q_{n-1,n} & 0 \end{pmatrix}$$

For each x_i, it can be set to either 0 or 1. That is to say, if we use enumeration method, there will be 2^n possibilities. With the increasing of the number of dimension, calculation will grow exponentially.

Here, we proposed a novel algorithm to solve *BQP*7 in polynomial time based on the property of matrix Q. In this algorithm, each time when we set x_i to 0 or 1, at most eight forms of $f(x)$ existed. This can lead to limited calculation, and make the algorithm effective and efficient.

In our proposed algorithm, we first let $i=n$ and set x_i, x_{i-1} and x_{i-2} to 0 or 1, generating eight states of x ($x=(x_1,\ldots,x_n)$) and the corresponding eight forms of $f(x)$. For every two adjacent states of x (only x_i are different, like *state* (0) and *state* (1), *state* (2) and *state* (3) in Table 1), only two terms in $f(x)$ are different. One is the term contains variable x_{i-3} and the other is the constant term. When we further set x_{i-3} to 0 and 1, these different two terms can be compared. Thus, we eliminate the bad state and keep the good one for next calculations. We can apply the similar process for the remaining seven states. Therefore, each time when we assign 0 or 1 to x_i, there will be only eight forms of $f(x)$ left. The flow chart of our algorithm is shown in Fig. 1 and the procedures of our algorithm are given in detail as follows:

Procedure 2.1. *Generate eight forms of $f(x)$ by assigning 0 or 1 to x_i, x_{i-1}, x_{i-2}.*

Let $i=n$ at the first round of calculation.

Assign (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), *and* (1,1,1) *to* (x_{i-2},x_{i-1},x_i) *to generate eight forms of $f(x)$, which is summarized in Table 1.*

Procedure 2.2. *Set x_{i-3} to 0 and 1 respectively to generate the new states. This is the key procedure of the whole algorithm.*

Step 1: *Set x_{i-3} to 0. For state* (0) *and state* (1) *in Table 1, they are only different in the linear term coefficient of x_{i-3} and the constant term. Therefore we set $x_{i-3}=0$, calculate $f(x)$ of state* (0) *and state* (1) *respectively. Compare these two results to eliminate the bad state and preserve the good one as the new state* (0). *Similarly, the new state* (1) *can be generated from state* (2) *and state* (3) *by setting $x_{i-3}=0$, the new state* (2) *is from state* (4) *and state* (5) *and the new state* (3) *is from state* (6) *and state* (7).

Step 2: *Set x_{i-3} to 1. By applying the same updating process, we set x_{i-3} to 1 and calculate $f(x)$ of state* (0) *and state* (1). *Compare the results and choose the good one as the new state* (4). *Similarly, we can obtain the new state* (5) *from state* (2) *and state* (3); *the new state* (6) *from state* (4) *and state* (5); *and the new state* (7) *from state* (6) *and state* (7). *By applying the above two steps, we will get a new table of eight forms of $f(x)$ and its corresponding states of x.*

Procedure 2.3. *Set the remaining x_i to 0 or 1.*

Set i to $i-1$ and repeat Procedure 2.3, we can generate a table of eight states of $f(x)$ and its corresponding states of x for each time when we set x_i to 0 or 1. After every variable in x is being set to 0 or 1, the final table contains only constant term of $f(x)$ and its corresponding states of x. Choose the optimal value and the corresponding value of x to get the optimal solution.

Based on the above procedures, we can demonstrate the polynomial solvable algorithm for *UBQP*7 as follows:

Algorithm 2.1. Polynomial solvable algorithm to *UBQP*7.

Step 1: Let $i=n$ and set x_i, x_i-1, and x_i-2 to 0 or 1.
Step 2: Calculate $f(x)$ separately when (x_{i-2},x_{i-1},x_i) is assigned to (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), generating eight states of $f(x)$.
Step 3: Let $i=n-3$.

Table 1
Different states of $f(x)$.

States	(x_{i-2},x_{i-1},x_i)	$f(x_1,\ldots,x_n)$
state (0)	(0,0,0)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-4}x_{i-4}+c_{i+1}x_{i+1}+\cdots+c_nx_n+c_{i-3}x_{i-3}$
state (1)	(0,0,1)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-4}x_{i-4}+c_{i+1}x_{i+1}+\cdots+c_nx_n+(c_{i-3}+q_{i-3,i})x_{i-3}+c_i$
state (2)	(0,1,0)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-5}x_{i-5}+c_{i+1}x_{i+1}+\cdots+c_nx_n$ $+(c_{i-4}+q_{i-4,i-1})x_{i-4}+(c_{i-3}+q_{i-3,i-1})x_{i-3}+c_{i-1}$
state (3)	(0,1,1)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}n+c_1x_1+\cdots+c_{i-5}x_{i-5}+c_{i+1}x_{i+1}+\cdots+c_nx_n$ $+(c_{i-4}+q_{i-4,i-1})x_{i-4}+(c_{i-3}+q_{i-3,i-1}+q_{i-3,i})x_{i-3}+c_{i-1}+c_i$
state (4)	(1,0,0)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-6}x_{i-6}+c_{i+1}x_{i+1}+\cdots+c_nx_n+(c_{i-5}+q_{i-5,i-2})x_{i-5}$ $+(c_{i-4}+q_{i-4,i-2})x_{i-4}+(c_{i-3}+q_{i-3,i-2})x_{i-3}+c_{i-2}$
state (5)	(1,0,1)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-4}x_{i-4}+c_{i+1}x_{i+1}+\cdots+c_nx_n+(c_{i-5}+q_{i-5,i-2})x_{i-5}$ $+(c_{i-4}+q_{i-4,i-2})x_{i-4}+(c_{i-3}+q_{i-3,i-2}+q_{i-3,i})x_{i-3}+c_{i-2}+q_{i-2,i}+c_i$
state (6)	(1,1,0)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-6}x_{i-6}+c_{i+1}x_{i+1}+\cdots+c_nx_n+(c_{i-5}+q_{i-5,i-2})x_{i-5}+(c_{i-4}+$ $q_{i-4,i-2}+q_{i-4,i-1})x_{i-4}+(c_{i-3}+q_{i-3,i-2}+q_{i-3,i-1})x_{i-3}+c_{i-2}+q_{i-2,i-1}+c_{i-1}$
state (7)	(1,1,1)	$q_{12}x_1x_2+\cdots+q_{i-4,i-3}x_{i-4}x_{i-3}+q_{i+1,i+2}x_{i+1}x_{i+2}+\cdots+q_{n-1,n}x_{n-1}x_n+c_1x_1+\cdots+c_{i-6}x_{i-6}+c_{i+1}x_{i+1}+\cdots+c_nx_n+(c_{i-5}+q_{i-5,i-2})x_{i-5}+(c_{i-4}+$ $q_{i-4,i-2}+q_{i-4,i-1})x_{i-4}+(c_{i-3}+q_{i-3,i-2}+q_{i-3,i-1}+q_{i-3,i})x_{i-3}+c_{i-2}+q_{i-2,i-1}+q_{i-2,i}+c_{i-1}+q_{i-1,i}+c_i$