European Journal of Operational Research 260 (2017) 494–506

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Markov Chain methods for the Bipartite Boolean Quadratic Programming Problem

Daniel Karapetyan [a,b,c,*], Abraham P. Punnen [c], Andrew J. Parkes [b]

[a] Institute for Analytics and Data Science, University of Essex, Colchester CO4 3SQ, UK
[b] ASAP Research Group, School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
[c] Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450 102nd AV, Surrey, British Columbia V3T 0A3, Canada

ARTICLE INFO

Article history:
Received 27 April 2016
Accepted 2 January 2017
Available online 6 January 2017

Keywords:
Artificial intelligence
Bipartite Boolean quadratic programming
Automated heuristic configuration
Benchmark

ABSTRACT

We study the Bipartite Boolean Quadratic Programming Problem (BBQP) which is an extension of the well known Boolean Quadratic Programming Problem (BQP). Applications of the BBQP include mining discrete patterns from binary data, approximating matrices by rank-one binary matrices, computing the cut-norm of a matrix, and solving optimisation problems such as maximum weight biclique, bipartite maximum weight cut, maximum weight induced sub-graph of a bipartite graph, etc. For the BBQP, we first present several algorithmic components, specifically, hill climbers and mutations, and then show how to combine them in a high-performance metaheuristic. Instead of hand-tuning a standard metaheuristic to test the efficiency of the hybrid of the components, we chose to use an automated generation of a multi-component metaheuristic to save human time, and also improve objectivity in the analysis and comparisons of components. For this we designed a new metaheuristic schema which we call Conditional Markov Chain Search (CMCS). We show that CMCS is flexible enough to model several standard metaheuristics; this flexibility is controlled by multiple numeric parameters, and so is convenient for automated generation. We study the configurations revealed by our approach and show that the best of them outperforms the previous state-of-the-art BBQP algorithm by several orders of magnitude. In our experiments we use benchmark instances introduced in the preliminary version of this paper and described here, which have already become the de facto standard in the BBQP literature.

1. Introduction

The (Unconstrained) Boolean Quadratic Programming Problem (BQP) is to

$$\begin{aligned} &\text{maximise} && f(x) = x^T Q' x + c' x + c'_0 \\ &\text{subject to} && x \in \{0,1\}^n, \end{aligned}$$

where Q' is an $n \times n$ real matrix, c' is a row vector in $\mathbb{R}^n$, and c'_0 is a constant. The BQP is a well-studied problem in the operational research literature (Billionnet, 2004). The focus of this paper is on a problem closely related to BQP, called the *Bipartite (Unconstrained) Boolean Quadratic Programming Problem* (BBQP) (Punnen, Sripratak, & Karapetyan, 2015b). BBQP can be defined as follows:

$$\begin{aligned} &\text{maximise} && f(x,y) = x^T Q y + cx + dy + c_0 \\ &\text{subject to} && x \in \{0,1\}^m, \quad y \in \{0,1\}^n, \end{aligned}$$

where $Q = (q_{ij})$ is an $m \times n$ real matrix, $c = (c_1, c_2, \ldots, c_m)$ is a row vector in $\mathbb{R}^m$, $d = (d_1, d_2, \ldots, d_n)$ is a row vector in $\mathbb{R}^n$, and c_0 is a constant. Without loss of generality, we assume that $c_0 = 0$, and $m \le n$ (which can be achieved by simply interchanging the rows and columns if needed). In what follows, we denote a BBQP instance built on matrix Q, row vectors c and d and $c_0 = 0$ as BBQP(Q, c, d), and (x, y) is a feasible solution of the BBQP if $x \in \{0, 1\}^m$ and $y \in \{0, 1\}^n$. Also x_i stands for the ith component of the vector x and y_j stands for the jth component of the vector y.

A graph theoretic interpretation of the BBQP can be given as follows (Punnen et al., 2015b). Let $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$. Consider a bipartite graph $G = (I, J, E)$. For each node $i \in I$ and $j \in J$, respective costs c_i and d_j are prescribed. Furthermore, for each $(i, j) \in E$, a cost q_{ij} is given. Then the *Maximum Weight Induced Subgraph Problem* on G is to find a subgraph $G' = (I', J', E')$ such that $\sum_{i \in I'} c_i + \sum_{j \in J'} d_j + \sum_{(i,j) \in E'} q_{ij}$ is maximised, where $I' \subseteq I$, $J' \subseteq J$ and G' is induced by $I' \cup J'$. The Maximum Weight Induced Subgraph Problem on G is precisely the BBQP, where $q_{ij} = 0$ if $(i, j) \notin E$.

* Corresponding author at: Institute for Analytics and Data Science, University of Essex, Colchester CO4 3SQ, UK.
E-mail addresses: daniel.karapetyan@gmail.com (D. Karapetyan), apunnen@sfu.ca (A.P. Punnen), andrew.parkes@nottingham.ac.uk (A.J. Parkes).

http://dx.doi.org/10.1016/j.ejor.2017.01.001

There are some other well known combinatorial optimisation problems that can be modelled as a BBQP. Consider the bipartite graph $G = (I, J, E)$ with w_{ij} being the weight of the edge $(i, j) \in E$. Then the *Maximum Weight Biclique Problem* (MWBP) (Ambühl, Mastrolilli, & Svensson, 2011; Tan, 2008) is to find a biclique in G of maximum total edge-weight. Define

$$q_{ij} = \begin{cases} w_{ij} & \text{if } (i, j) \in E, \\ -M & \text{otherwise,} \end{cases}$$

where M is a large positive constant. Set c and d as zero vectors. Then BBQP(Q, c, d) solves the MWBP (Punnen et al., 2015b). This immediately shows that the BBQP is NP-hard and one can also establish some approximation hardness results with appropriate assumptions (Ambühl et al., 2011; Tan, 2008). Note that the MWBP has applications in data mining, clustering and bioinformatics (Chang, Vakati, Krause, & Eulenstein, 2012; Tanay, Sharan, & Shamir, 2002) which in turn become applications of BBQP.

Another application of BBQP arises in approximating a matrix by a rank-one binary matrix (Gillis & Glineur, 2011; Koyutürk, Grama, & Ramakrishnan, 2005; 2006; Lu, Vaidya, Atluri, Shin, & Jiang, 2011; Shen, Ji, & Ye, 2009). For example, let $H = (h_{ij})$ be a given $m \times n$ matrix and we want to find an $m \times n$ matrix $A = (a_{ij})$, where $a_{ij} = u_i v_j$ and $u_i, v_j \in \{0, 1\}$, such that $\sum_{i=1}^{m} \sum_{j=1}^{n} (h_{ij} - u_i v_j)^2$ is minimised. The matrix A is called a rank one approximation of H and can be identified by solving the BBQP with $q_{ij} = 1 - 2h_{ij}$, $c_i = 0$ and $d_j = 0$ for all $i \in I$ and $j \in J$. Binary matrix factorisation is an important topic in mining discrete patterns in binary data (Lu et al., 2011; Shen et al., 2009). If u_i and v_j are required to be in $\{-1, 1\}$ then also the resulting factorisation problem can be formulated as a BBQP.

The Maximum Cut Problem on a bipartite graph (MaxCut) can be formulated as BBQP (Punnen et al., 2015b) and this gives yet another application of the model. BBQP can also be used to find approximations to the cut-norm of a matrix (Alon & Naor, 2006).

For theoretical analysis of approximation algorithms for BBQP, we refer to Punnen, Sripratak, and Karapetyan (2015a).

A preliminary version of this paper was made available to the research community in 2012 (Karapetyan & Punnen, 2012). Subsequently Glover, Ye, Punnen, and Kochenberger (2015) and Duarte, Laguna, Martí, and Sánchez-Oro (2014) studied heuristic algorithms for the problem. The testbed presented in our preliminary report (Karapetyan & Punnen, 2012) continues to be the source of benchmark instances for the BBQP. In this paper, in addition to providing a detailed description of the benchmark instances, we refine the algorithms reported in Karapetyan and Punnen (2012), introduce a new class of algorithms and give a methodology for automated generation of a multi-component metaheuristic. By (algorithmic) component we mean a black box algorithm that modifies the given solution. All the algorithmic components can be roughly split into two categories: hill climbers, i.e. components that guarantee that the solution not be worsened, and mutations, i.e. components that usually worsen the solution. Our main goals are to verify that the proposed components are sufficient to build a high-performance heuristic for BBQP and also investigate the most promising combinations. By this computational study, we also further support the ideas in the areas of automated parameter tuning and algorithm configuration (e.g. see Adenso-Díaz & Laguna, 2006; Bezerra, López-Ibáñez, & Stützle, 2015; Hutter, Hoos, Leyton-Brown, & Stützle, 2009; Hutter, Hoos, & Stützle, 2007). Thus we rely entirely on automated configuration. During configuration, we use smaller instances compared to those in our benchmark. This way we ensure that we do not over-train our metaheuristics to the benchmark instances – an issue that is often quite hard to avoid with manual design and configuration. We apply the resulting multi-component metaheuristic to our benchmark instances demonstrating that a combination of several simple components can yield powerful metaheuristics clearly outperforming the state-of-the-art BBQP methods.

The main contributions of the paper include:

- In Section 2, we describe several BBQP algorithmic components, one of which is completely new.
- In Section 3 we take the Markov Chain idea, such as in the Markov Chain Hyper-heuristic (McClymont & Keedwell, 2011), but restrict it to use static weights (hence having no online learning, and so, arguably, not best labelled as a 'hyper-heuristic'), but instead adding a powerful extension to it, giving what we call 'Conditional Markov Chain Search (CMCS)'.
- In Section 4 we describe five classes of instances corresponding to various applications of BBQP. Based on these classes, a set of benchmark instances is developed. These test instances were first introduced in the preliminary version of this paper (Karapetyan & Punnen, 2012) and since then used in a number of papers (Duarte et al., 2014; Glover et al., 2015) becoming de facto standard testbed for the BBQP.
- In Section 5 we use automated configuration of CMCS to demonstrate the performance of individual components and their combinations, and give details sufficient to reproduce all of the generated metaheuristics. We also show that a special case of CMCS that we proposed significantly outperforms several standard metaheuristics, on this problem.
- In Section 6 we show that our best machine-generated metaheuristic is, by several orders of magnitude, faster than the previous state-of-the-art BBQP method.

2. Algorithmic components

In this section we introduce several algorithmic components for BBQP. Except for 'Repair' and 'Mutation-X/Y', these components were introduced in Karapetyan and Punnen (2012). A summary of the components discussed below is provided in Table 1. The components are selected to cover a reasonable mix of fast and slow hill climbing operators for intensification, along with mutation operators that can be expected to increase diversification, and with Repair that does a bit of both. Note that a hill climbing component can potentially implement either a simple improvement move or a repetitive local search procedure with iterated operators that terminates only when a local maximum is reached. However in this project we opted for single moves leaving the control to the metaheuristic framework.

2.1. Components: Optimise-X/Optimise-Y

Observe that, given a fixed vector x, we can efficiently compute an optimal $y = y_{\text{opt}}(x)$:

$$y_{\text{opt}}(x)_j = \begin{cases} 1 & \text{if } \sum_{i \in I} q_{ij} x_i + d_j > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

This suggests a hill climber operator Optimise-Y (OptY) that fixes x and replaces y with $y_{\text{opt}}(x)$. Eq. (1) was first introduced in Punnen et al. (2015b) and then used as a neighbourhood search operator in Karapetyan and Punnen (2012), Duarte et al. (2014) and Glover et al. (2015).

OptY implements a hill climber operator in the neighbourhood $N_{\text{OptY}}(x, y) = \{(x, y') : y' \in \{0, 1\}^n\}$, where (x, y) is the original solution. Observe that the running time of OptY is polynomial and the size of the neighbourhood $|N_{\text{OptY}}(x, y)| = 2^n$ is exponential; hence OptY corresponds to an operator that could be used in a very large-scale neighbourhood search (VLNS), a method that is often considered as a powerful approach to hard combinatorial optimisation problems Ahuja, Ergun, Orlin, Punnen, (2002).

Observe that OptY finds a local maximum after the first application because $N(x, y) = N(x, y_{\text{opt}}(y))$ (that is, it is an 'idempotent