UNIVERSIDADE FEDERAL FLUMINENSE

JOSÉ ANGEL RIVEAUX MERINO

AN EXACT ALGORITHM FOR THE MAXIMUM QUASI-CLIQUE PROBLEM

NITERÓI 2017

UNIVERSIDADE FEDERAL FLUMINENSE

JOSÉ ANGEL RIVEAUX MERINO

AN EXACT ALGORITHM FOR THE MAXIMUM QUASI-CLIQUE PROBLEM

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização

Orientador: CELSO C. RIBEIRO

> NITERÓI 2017

JOSÉ ANGEL RIVEAUX MERINO

AN EXACT ALGORITHM FOR THE MAXIMUM QUASI-CLIQUE PROBLEM

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização

Aprovada em Julho de 2017.

BANCA EXAMINADORA

Prof. Celso da Cruz Carneiro Ribeiro - Orientador, UFF

Profa. Isabel Rosseti, UFF

Profa. Lilian Markenzon, UFRJ

Prof. Luis Marti Orosa, UFF

Niterói 2017

Dedicatória(s): A mi famila, professores e amigos...

Agradecimentos

Obrigado a: Dios, Egda Caridad Toca Saldinha, Raul Saldivar Pena, meus pais e familia, Celso C. Ribeiro, Isabel Rosseti, Lilian Markenzon, Luis Marti Orosa, Teresa Cancela, Sergio Prado Pires, Ruben Interian Kavaliof, Ruslan Guerra Marzo, Jorge Moreno Ramirez, Bruno Queiroz Pinto, Alberto Reyes, Isela Mendoza, José Ramón González, a todos meus professores, amigos e companheiros da UFF que de contribuiram e apoiaram na realização de este trabalho.

Resumo

Dado um grafo G = (V, E) e um limitante $\gamma \in (0, 1]$, uma γ -clique é qualquer subconjunto $C \in V$ tal que a densidade do grafo induzido em G por C é maior ou igual a γ . O problema da γ -clique de cardinalidade máxima consiste em determinar um subconjunto de cardinalidade máxima C^* dos nós de V tal que a densidade do grafo induzido em Gpor C^* seja maior ou igual ao limitante. Na primeira parte dessa dissertação, propçõe-se um algoritmo exato para o problema da quase-clique máxima, baseado em uma propriedade quase-hereditária. Experimentos computacionais mostram que o novo algoritmo é competitivo com as melhores formulações exatas da literatura resolvidas pelo CPLEX. O algoritmo também fornece uma nova cota superior que é consistentemente mais justa do que as cotas já conhecidas. Na segunda parte da dissertação, propõe-se a hibridização de um algoritmo genético com chaves aleatórias tendenciosas como algoritmo exato desenvolvido na primeira parte da dissertação. Resultados computacionais mostram que o enfoque híbrido tem melhor desempenho do que o algoritmo genético original.

Palavras-chave: Problema de quasi-clique de cardinalidade máxima; problema de máximo quasi-clique; grafos; algoritmo genético baseado em chaves aleatórias; metaheurísticas; densidade do grafo.

Abstract

Given a graph G = (V, E) and a threshold $\gamma \in (0, 1]$, a γ -clique is any subset C of V such that the density of the graph induced in G by C is greater than or equal to γ . The maximum cardinality γ -clique problem amounts to finding a maximum cardinality subset C^* of the vertices in V such that the density of the graph induced in G by C^* is greater than or equal to the threshold γ . In the first part of this dissertation, we propose an exact algorithm for solving the maximum quasi-clique problem, based on a quasi-hereditary property. Computational experiments show that the new algorithm is competitive with the best formulations in the literature solved by CPLEX. The algorithm also provides a new upper bound that is consistently tighter than previously existing bounds. In the second part of this dissertation, we propose a hybridization of a biased random-key genetic algorithm with the exact algorithm developed in the first part of the dissertation. Computational results show that the hybrid approach outperforms the original genetic algorithm.

Keywords: Maximum cardinality quasi-clique problem; maximum quasi-clique problem; maximum γ -clique problem; maximum clique problem; graphs; biased random-key genetic algorithm; metaheuristics; graph density.

Sumário

Resumo Estendido	6
Referências	8
Apêndice A - An exact algorithm for the maximum quasi-clique problem	9
Apêndice B – Improving a biased random-key genetic algorithm for the maximum quasi-clique problem with an exact local search	28

Resumo Estendido

Seja G = (V, E) um grafo definido por um conjunto de vértices V e um conjunto de arestas $E \subseteq V \times V$. Diz-se que G é um grafo completo se existe uma aresta em E para cada par de vértices de V. O grafo G' = (V', E') é um subgrafo de G se $V' \subseteq V$ e $E' \subseteq E$, o que é denotado como $G' \subseteq G$. O grafo G(V') induzido em G por $V' \subseteq V$ é aquele com o conjunto de vértices V' e com o subconjunto de todas as arestas de E que tem as duas extremidades em V'. Para qualquer $V' \subseteq V$, o subconjunto $E(V') \subseteq E$ é formado por todas as arestas de E com ambas extremidades em V' (i.e., E(V') é o conjunto de arestas do grafo induzido em G por V').

A densidade do grafo G é dada por $dens(G) = |E|/(|V| \times (|V| - 1)/2)$. Nota-se que a densidade de um gráfo completo é igual a um.

Um subconjunto $C \subseteq V$ é uma clique de G se o subgrafo G(C) inducido em G por Cé completo. Então, dado um grafo G = (V, E), o problema da clique máxima consiste em encontrar a clique de cardinalidade máxima de G. Este problema está entre os primeiros problemas que foram provados serem NP-difíceis em [2].

Muitas vezes, em vez de se conhecer a clique máxima, o que se deseja é ter o maior conjunto de vértices tal que o grafo induzido seja próximo de uma clique. Logo, dado um grafo G = (V, E) e um limite $\gamma \in (0, 1]$, uma γ -clique é qualquer subconjunto $C \subseteq V$ tal que a densidade de G(C) é maior ou igual a γ . A γ -clique C é maximal se não existe outra γ -clique C' tal que C esteja estritamente contida em C'. Então, o problema da quasi-clique de cardinalidade máxima (MQCP) consiste em encontrar o subconjunto C^* de vértices de V de maior cardinalidade tal que a densidade do grafo induzido em G por C^* seja maior ou igual a γ . Este problema também NP-difícil, já que admite o problema da clique máxima como um caso especial onde $\gamma = 1$; ver [4]. Esse problema tem aplicações em mineração de dados, e.g. em redes sociais ou grafos de chamadas telefônicas [1].

O presente trabalho faz um estudo do MQCP e propõe um algoritmo exato e enumerativo como solução do problema, que constitui a principal contribuição da pesquisa. O algoritmo é baseado em uma propriedade de quasi-hereditariedade, ver [4]. Esta propriedade leva ao teorema que é apresentado e provado na primeira parte da dissertação. Experimentos computacionais mostram que o novo algoritmo é competitivo com as melhores formulações em [4, 8] resolvidas pelo CPLEX e com o algoritmo *branch and bound* proposto em [3]. O algoritmo também fornece um novo limite superior, melhor do que os limites existentes anteriormente na literatura.

Como parte da pesquisa é apresentado também uma hibridação entre o algoritmo genético de chave aleatórias tendenciosas e o algoritmo exato descrito anteriormente. O algoritmo exato é aplicado como uma busca local no processo de reconstrução do decodificador. Esta proposta é comparada com o algoritmo BRKGA-IG^{*} de [6], que era a melhor heurística existente na literatura até o momento da redação dessa dissertação. Os resultados computacionais mostram que o BRKGA híbrido supera BRKGA-IG^{*}.

O trabalho desenvolvido está apresentado em dois artigos colocados como apêndices ao corpo da presente disertação. O primeiro artigo [7] apresenta a bibliografia sobre métodos exatos, a proposta do algoritmo exato e os experimentos que mostram os resultados obtidos e as conclusões do artigo. A segunda parte do trabalho é dedicada ao BRKGA híbrido e é mostrada no segundo artigo [5]. O mesmo contém um resumo dos trabalhos relacionados, a proposta do algoritmo híbrido, os resultados e conclusões.

Referências

- ABELLO, J., RESENDE, M., SUDARSKY, S. Massive quasi-clique detection. In Proceedings of the 5th Latin American Symposium on the Theory of Informatics, J. Abello and J. Vitter, Eds., vol. 2286 of Lecture Notes in Computer Science. Springer, Berlin, 2002, p. 598–612.
- [2] KARP, R. M. Reducibility among combinatorial problems. In Complexity of Computer Computations (New York, 1972), R. E. Miller and J. W. Thatcher, Eds., Plenum, p. 85–103.
- [3] PAJOUH, F. M., MIAO, Z., BALASUNDARAM, B. A branch-and-bound approach for maximum quasi-cliques. Annals of Operations Research 216 (2014), 145–161.
- [4] PATTILLO, J., VEREMYEV, A., BUTENKO, S., BOGINSKI, V. On the maximum quasi-clique problem. *Discrete Applied Mathematics 161* (2013), 244–257.
- [5] PINTO, B. Q., RIBEIRO, C. C., RIVEAUX, J. A., ROSSETI, I. Improving a biased random-key genetic algorithm for the maximum quasi-clique problem with an exact local search. *International Transactions in Operational Research* (2017). Submitted.
- [6] PINTO, B. Q., RIBEIRO, C. C., ROSSETI, I., PLASTINO, A. A biased random-key genetic algorithm for solving the maximum quasi-clique problem. *European Journal* of Operational Research (2017). Submitted.
- [7] RIBEIRO, C. C., RIVEAUX, J. A. An exact algorithm for the maximum quasi-clique problem. *Journal of Global Optimization* (2017). Submitted.
- [8] VEREMYEV, A., PROKOPYEV, O. A., BUTENKO, S., PASILIAO, E. L. Exact mipbased approaches for finding maximum quasi-clique and dense subgraph. *Computati*onal Optimization and Applications 64 (2016), 177–214.

APÊNDICE A – An exact algorithm for the maximum quasi-clique problem

An Exact Algorithm for the Maximum Quasi-Clique Problem

Celso C. Ribeiro · Jose A. Riveaux

Received: June 1, 2017 / Accepted: date

Abstract Given a graph G = (V, E) and a threshold $\gamma \in (0, 1]$, the maximum quasi-clique problem amounts to finding a maximum cardinality subset C^* of the vertices in V such that the density of the graph induced in G by C^* is greater than or equal the threshold. This problem is NP-hard, since it admits the maximum clique problem as a special case. It has a number of applications in data mining, e.g. in social networks or phone call graphs. In this work, we present an exact algorithm to solve this problem, based on a quasi-hereditary property. We also propose a new upper bound that is used for pruning the search tree. Numerical results show that the new approach is competitive and outperforms the best integer programming approaches in the literature. The new upper bound is consistently tighter than previously existing bounds.

Keywords Maximum cardinality quasi-clique problem \cdot maximum quasiclique problem \cdot maximum γ -clique problem \cdot maximum clique problem \cdot graphs \cdot graph density

1 Introduction

Let G = (V, E) be a graph defined by a vertex set V and an edge set $E \subseteq V \times V$. G is a complete graph if there is an edge in E connecting every two different vertices in V. A graph G' = (V', E') is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$, which is denoted by $G' \subseteq G$. The graph G(V') induced in G by $V' \subseteq V$ is that with vertex set V' and edge set formed by all edges of E with both ends

Celso C. Ribeiro

Jose A. Riveaux

Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil. E-mail: celso@ic.uff.br

Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil. E-mail: jangel.riveaux@gmail.com

in V'. For any $V' \subseteq V$, the subset $E(V') \subseteq E$ is formed by all edges of G with both ends in V' (i.e., E(V') is the edge set of the graph induced in G by V').

The density of graph G is given by $dens(G) = |E|/(|V| \times (|V| - 1)/2)$.

A subset $C \subseteq V$ is a clique of G if the graph G(C) induced in G by C is complete. Given a graph G = (V, E), the maximum clique problem consists in finding a maximum cardinality clique of G. It was proved to be NP-hard by Karp (1972).

Given a graph G = (V, E) and a threshold $\gamma \in (0, 1]$, a γ -clique is any subset $C \subseteq V$ such that the density of G(C) is greater than or equal to γ . A γ -clique C is maximal if there is no other γ -clique C' such that C is strictly contained in C'. Therefore, the maximum quasi-clique problem (MQCP) amounts to finding a maximum cardinality subset C^* of the vertices in V such that the density of the graph induced in G by C^* is greater than or equal to the threshold γ . This problem is also NP-hard, since it admits the maximum clique problem as a special case in which $\gamma = 1$; see (Pattillo et al, 2013). The problem has many applications in data mining, e.g. in social networks or phone call graphs (Abello et al, 2002).

In this work, we present an exact algorithm for solving the maximum quasiclique problem, based on a quasi-hereditary property. Section 2 reviews exact formulations and algorithms for the problem. Section 3 presents the exact algorithm and demonstrates its correctness. We also propose a new upper bound that is used for pruning the search tree. Computational experiments are reported in Section 4, in which the proposed approach is compared with existing exact methods. Concluding remarks are drawn in the last section. The new algorithm is competitive with the best formulations in (Pattillo et al, 2013; Veremyev et al, 2016) solved by CPLEX and with the branch-and-bound algorithm in (Pajouh et al, 2014). The new upper bound is consistently tighter than previously existing bounds.

2 Related work and mathematical formulations

Pattillo et al (2013) introduced some properties and upper bounds of γ -cliques. Property 1 below will be used in the design of the exact algorithm proposed in Section 3.

Definition 1 Consider a graph G = (V, E) that satisfies a property P. If there exists a vertex $v \in V$ such that the induced graph $G(V \setminus \{v\})$ also satisfies property P, then we say that P is a quasi-hereditary property and that property P displays quasi-heredity or quasi-inheritance.

Property 1 The graph property of having edge density greater than or equal to γ displays quasi-inheritance, i.e., any γ -clique with s > 1 vertices is a strict superset of a γ -clique with s - 1 vertices.

We summarize below the two best mixed integer programming formulations for the maximum γ -clique problem reported in (Pattillo et al, 2013; Veremyev et al, 2016). The first formulation is the one that obtains the best results for small values of γ and for dense graphs among those in (Pattillo et al, 2013). The second formulation appeared in (Veremyev et al, 2016) and presents the best results and the tighter relaxation for sparse graphs. It turned out to be the most consistent model in their experiments.

Model F1 (Pattillo et al, 2013) has a binary variable x_i associated to each vertex of the graph:

$$x_i = \begin{cases} 1, & \text{if vertex } v_i \in V \text{ belongs to the solution,} \\ 0, & \text{otherwise.} \end{cases}$$

It also considers a variable $y_{ij} = x_i \cdot x_j$ associated to each pair of vertices $i, j \in V$, with i < j, that is linearized as below:

Model F1:
$$\max_{i \in V} x_i$$
 (1)

subject to:

$$\sum_{(i,j)\in E: i < j} y_{ij} \ge \gamma \cdot \sum_{i,j\in V: i < j} y_{ij} \tag{2}$$

$$y_{ij} \le x_i, \qquad \forall i, j \in V, \quad i < j,$$
(3)

$$y_{ij} \le x_j, \qquad \forall i, j \in V, \quad i < j,$$
(4)

$$y_{ij} \ge x_i + x_j - 1, \qquad i, j = 1, \dots, n, \quad i < j,$$
(5)

$$x_i \in \{0, 1\}, \qquad \forall i \in V, \tag{6}$$

$$y_{ij} \ge 0, \qquad \forall i, j \in V, \quad i < j$$

$$\tag{7}$$

The objective function (1) maximizes the number of vertices in the solution. If two vertices i, j belong to a solution, then $x_i = x_j = 1$ and $y_{ij} = x_i \cdot x_j = 1$. If edge $(i, j) \in E$, then it contributes to the density of the quasi-clique. Therefore, constraint (2) ensures that the density of the solution is greater than or equal to γ . Constraints (3) and (4) ensure that any edge may contribute to the density of a solution only if both of its ends are chosen to belong to this solution. Constraints (5) ensure that any existing edge $(i, j) \in E$ will contribute to the solution if both of its ends are chosen. Constraints (6) and (7) impose the integrality and non-negativity requirements on the problem variables, respectively.

To introduce the second formulation, let ω^u and ω^ℓ be, respectively, any upper and lower bounds on the size of a maximum γ -clique in G. The lower bound can be set to 1 if there is no information available about the sizes of γ -cliques in G. Its value can be increased using the size of any heuristically identified quasi-clique, e.g., it can be set to be the size of any clique in G. The upper bound can be simply set to the trivial value |V|, or we can use e.g. the result of (Pattillo et al, 2013):

$$\omega^{u} = \lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1 + 8\frac{|E|}{\gamma}} \rfloor,\tag{8}$$

or, if the graph is connected,

$$\omega^{u} = \lfloor \frac{1}{2} + \frac{2 + \sqrt{(\gamma + 2)^{2} + 8\gamma(|E| - |V|)}}{2\gamma} \rfloor.$$
(9)

Model F3 (Veremyev et al, 2016) also makes use of the binary variables x_i . New variables y_{ij} , with i < j, are defined as:

$$y_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \in E, \text{ with } i < j, \text{ belongs to the quasi-clique,} \\ 0, & \text{otherwise.} \end{cases}$$

In addition, let z_k be a binary variable that determines the size of the current solution, namely, $z_k = 1$ if and only if the γ -clique has k vertices, 0 otherwise. Using this notation, the following formulation was proposed for finding a maximum γ -clique based on the classical value-disjunction idea (see, e.g., (Nemhauser and Wolsey, 1988)) that says that only one of a set of values can be taken by some variable, which is applied to the size of a maximum γ -clique:

Model F3:
$$\max \sum_{i \in V} x_i$$
 (10)

subject to:

$$\sum_{(i,j)\in E: i < j} y_{ij} \ge \gamma \cdot \sum_{k=\omega^l}^{\omega^u} \frac{k(k-1)}{2} z_k, \tag{11}$$

$$y_{ij} \le x_i, \qquad \forall (i,j) \in E, \quad i < j,$$
 (12)

$$y_{ij} \le x_j, \qquad \forall (i,j) \in E, \quad i < j,$$

$$(13)$$

$$\sum_{i \in V} x_i = \sum_{k=\omega^l} k \cdot z_k, \tag{14}$$

$$\sum_{k=\alpha^{d}}^{\omega^{u}} z_{k} = 1, \tag{15}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in V, \tag{16}$$

$$y_{ij} \ge 0, \qquad \forall i, j \in V, \quad i < j, \tag{17}$$

$$z_k \ge 0, \qquad \forall k \in \{\omega^l, \dots, \omega^u\}.$$
 (18)

Constraints (12) and (13) ensure that y_{ij} can be set to 1 only if both vertices i and j belong to the γ -clique, i.e., $x_i = x_j = 1$. Constraint (11) represents the edge density requirements for the subgraph induced by the k chosen vertices, while constraints (14) and (15), together, enforce the appropriate value in the right hand-side of (11). As for model F1, variables y_{ij} can be relaxed to be continuous in constraint (17) due to the structure of constraints (11) to (13) and to the fact that the objective function maximizes a linear function with positive costs. Also, the binary restrictions for the z_k variables may be replaced by the non-negativity constraints (18).

Pajouh et al (2014) proposed an exact depth-first search based enumeration algorithm for the maximum quasi-clique problem (Kreher and Stinson (1998), chapter 4).

3 Exact algorithm based on the quasi-hereditary property

From Property 1, if a γ -clique of size k does not exist in graph G, then no larger γ -clique exists as well. The exact algorithm proposed in this work starts from a lower-bound ω^{ℓ} and iteratively searches for a γ -clique of size $k = \omega^{\ell+1}, \omega^{\ell+2}, \ldots$ until it fails. The initial lower bound could be set to $\ell = 1$ or calculated by any heuristic. The largest value of k for which a γ -clique of size k is found is the optimal value. The following theorem holds:

Theorem 1 Let C^* be a γ -clique of cardinality k of graph G. Then, there exists a permutation Π of the vertices in C^* such that if Π_i denotes the first i vertices in permutation Π (i.e., Π_i is the prefix of Π of size i), $G(\Pi_i)$ is the subgraph induced in G by the vertices in Π_i and k > 1, then $dens(G(\Pi_i)) \ge dens(G(\Pi_{i+1}))$, for all $i = 1, \ldots, k - 1$.

Proof: This result follows as a consequence of the quasi-inheritance Property 1. It can be proved by a constructive algorithm that builds the permutation from scratch. First, select the vertex v' of C^* with minimum degree in $G(C^*)$ to be the last vertex in Π and remove it from C^* . Then, the density of the new induced graph $G(C^* \setminus \{v'\})$ is greater than that of $G(C^*)$, because the degree of the removed vertex does not increase the average degree of the graph (Pattillo et al, 2013). This step can be repeated iteratively and the algorithm finishes when all vertices have been selected and the resulting sequence meets the condition of Theorem 1.



Fig. 1 Illustration of Theorem 1 on a graph G with five vertices when a maximum γ -clique for $\gamma = 0.7$ is sought. $C^* = V = \{1, 2, 3, 4, 5\}$ is a maximum 0.7-clique, since $dens(G(C^*)) = 7/10 \ge \gamma$.

Figure 1 illustrates Theorem 1 on a graph G with five vertices when a maximum γ -clique for $\gamma = 0.7$ is sought. $C^* = V = \{1, 2, 3, 4, 5\}$ is a maximum 0.7-clique, since $dens(G(C^*)) = 7/10 \ge \gamma$. In Figure 1(a), vertex 1 is selected as that with the smallest degree in G and placed in the last position of the permutation. The resulting subgraph has density $0.8\overline{3}$ and $\Pi = <1 >$. Next,

vertex 2 is selected in Figure 1(b) and the resulting permutation becomes $\Pi = < 2, 1 >$. The remaining subgraph in Figure 1(c) is a clique and any order of the vertices will meet the conditions of the theorem. In particular, the final permutation could be e.g. $\Pi = < 3, 4, 5, 2, 1 >$. The subgraphs induced by $\Pi_1 = < 3 >$, $\Pi_2 = < 3, 4 >$, and $\Pi_3 = < 3, 4, 5 >$ have their densities equal to 1, the subgraph induced by $\Pi_4 = < 3, 4, 5, 2, 1 >$ has its density equal to $0.8\overline{3}$, and that induced by $\Pi_5 = \Pi = < 3, 4, 5, 2, 1 >$ has density 0.7.

3.1 Enumeration tree: Backtracking

The enumeration procedure amounts to constructing the optimal vertex permutation sequence that leads to the maximum γ -clique. Although there are k!vertex permutations associated with any γ -clique with k vertices, considering the demonstration of Theorem 1, the vertex to be selected and removed in each step is always that with the minimum degree. Therefore, the last vertex in every prefix of the permutation is always that with minimum degree. In case there are ties and two or more vertices have minimum degree, the optimal sequence might not be unique. To avoid this situation, we suppose that the vertices of V are labeled with $1, 2, \ldots, |V|$ and, in case of ties, the enumeration always selects that with the largest label. Therefore, the generated sequence will be unique.

Algorithm 1 below is a backtracking strategy that enumerates all γ -cliques in G without repetition. The first call to the algorithm is performed with $\Pi = \emptyset$. The vertices that can be inserted at the end of the current permutation Π form the candidate list CL. In line 1, function getCandidates generates the candidate list CL. The candidate list returned after the call to getCandidates for Π_0 is CL = V. The recursion is interrupted in line 3 when the candidate list becomes empty. The loop in lines 5 to 8 creates a new permutation Π' in line 6 for each candidate $j \in CL$ and makes a recursive call in line 7.

Algorithm 1 Backtracking

ingonitini i Dachtraching
Require: G, Π, γ
1: $CL \leftarrow getCandidates(G, \Pi, \gamma)$
2: if $CL = \emptyset$ then
3: return
4: end if
5: for all $j \in CL$ do
6: $\Pi' \leftarrow \Pi \oplus \{j\}$
7: Backtracking (G, Π', γ)
8: end for

Algorithm 2 called in line 1 of Algorithm 1 builds the candidate list CL formed by the vertices that can be inserted at the end of a prefix in line 7. Line 1 initializes CL as empty and line 2 initializes i as the size of the current permutation Π . The loop in lines 3 to 11 considers the addition to the current permutation Π of all vertices in $V \setminus \Pi$. We denote by $deg_{G(\Pi)}(j)$ the number of vertices of the current permutation Π that are adjacent to vertex j. Line 4 checks if, for a candidate vertex j, the subgraph induced by $\Pi \cup \{j\}$ is a γ -clique. Recall that $E(\Pi)$ denotes the edge set of the graph induced in G by Π . Line 5 creates a tentative sequence Π' by appending vertex j at the end of sequence Π . Lines 6 determines if there is another vertex $w \in \Pi$ with degree smaller than j or with the same degree as j but with a larger label. If this is not the case, then vertex j is definitely added to the candidate list CL in line 7. The candidate list CL is returned in line 11.

Algorithm 2 getCandidates

Require: G, Π, γ $1 \colon CL \leftarrow \emptyset$ $2:\ i \leftarrow |\Pi|$ 3: for all $j \in V \setminus \Pi$ do if $deg_{G(\Pi)}(j) + |E(\Pi)| \ge \gamma(i+1)i/2$ then 4: $\Pi' \leftarrow \Pi \oplus \{j\}$ 5:if $\{w \in \Pi : deg_{G(\Pi')}(w) < deg_{G(\Pi')}(j) \text{ or } (deg_{G(\Pi')}(w) = deg_{G(\Pi')}(j) \text{ and } w \in \Pi : deg_{G(\Pi')}(w) < deg_{G(\Pi')}(j) \}$ 6: w > j = \emptyset then 7: $\mathit{CL} \leftarrow \mathit{CL} \cup \{j\}$ 8: end if 9: end if 10: end for 11: return CL

3.2 Pruning and a new upper bound

Algorithm 1 produces an enumeration tree. Each of its non-root nodes corresponds to a feasible solution to MQCP. If at some time of the search the incumbent (i.e., the best known solution) has size LB, a necessary condition for a larger solution to exist is that at least a solution of size k = LB + 1 exists. Assume that the current node of the search tree represents a solution Π of size $i \leq LB$ and let $R = V \setminus \Pi$. Then, a solution with k vertices and prefix Π exists if and only if there exists a subset of k - i vertices in R whose union with the vertices in Π induces a γ -clique in G. If such a subset of vertices does not exist in R, then the tree is pruned at the current node.

If, for any vertex $v \in R$, $\deg_{G(\Pi)}(v)$ exceeds the degree of the last vertex of Π in $G(\Pi)$ by more than k units, then v can not be the last vertex in any possible other permutation with prefix Π and size smaller than or equal to k. Therefore, if w is the vertex with minimum degree in $G(\Pi)$, R can be reduced to:

$$R = \{ v \in V \setminus \Pi : \deg_{G(\Pi)}(v) \le \deg_{G(\Pi)}(w) + k \}.$$

$$\tag{19}$$

We recall that E(V') denotes the set of edges of E with both ends in $V' \subseteq V$. Furthermore, let $E(V', \Pi)$ be the set of edges with one extremity

in V' and the other in $\varPi.$ If $V'\subseteq R$ has k-i vertices, then $G(\varPi\cup V')$ is a $\gamma\text{-clique if}$

$$|E(\Pi) \cup E(V') \cup E(V',\Pi)| \ge \gamma \cdot \binom{k}{2}.$$
(20)

 $|E(\Pi)|$ is a known value in equation (20). Let F be the set formed by the k-i vertices in R with the largest degrees in $G(\Pi)$ and D be the set of k-i vertices in R with the largest degrees in G. Then, for any $V' \subseteq R$:

$$|E(V',\Pi)| \le \sum_{v \in F} \deg_{G(\Pi)}(v), \text{and}$$
(21)

$$|E(V',\Pi)| + 2 \cdot |E(V')| \le \sum_{v \in D} \deg_G(v).$$
(22)

Assuming that $E(V', \Pi)$ is maximized, we obtain

$$|E(V')| \le \min\{\frac{\sum_{v \in D} \deg_G(v) - \sum_{v \in F} \deg_{G(\Pi)}(v)}{2}, \binom{|V'|}{2}\}.$$
 (23)

Replacing $|E(V', \Pi)|$ and |E(V')| in equation (20) by their upper bounds in inequalities (21) and (23), respectively, we obtain:

$$|E(\Pi)| + \sum_{v \in F} \deg_{G(\Pi)}(v) + \min\{\frac{\sum_{v \in D} \deg_G(v) - \sum_{v \in F} \deg_{G(\Pi)}(v)}{2}, \binom{|V'|}{2}\} \ge \gamma \binom{k}{2}.$$
(24)

Therefore, if inequality (24) is not satisfied then it does not exist $V' \subseteq R$ such that $G(V' \cup \Pi)$ is a γ -clique. If the same holds for any $k' \in [i+1, k-i]$, then there is no γ -clique of size k with Π as a prefix.

Inequality (24) can be used in the computation of a new upper bound UB to MQCP. First note that if $\Pi_0 = \emptyset$, then $|E(\Pi_0)| = 0$, $deg_{G(\Pi_0)}(v) = 0$ for all $v \in R$, and |V'| = k. Then UB is the largest value of $k = 1, \ldots, |V(G)|$ such that

$$\frac{\sum_{v \in D} \deg_G(v)}{2} \ge \gamma \cdot \binom{k}{2}.$$
(25)

3.3 Algorithm QClique

The pseudo-code of the QClique algorithm for the maximum quasi-clique problem is presented in Algorithm 3. The initial parameters are the graph G = (V(G), E(G)), the solution Π which is initially an empty set, the threshold γ , a lower bound LB and the incumbent (best solution) Π^* . Line 1 initializes *i* as the size of the current permutation Π . If line 2 determines by applying inequality (24) that a feasible solution with size greater than that of the current lower bound LB can not be obtained by extending the current solution Π , then this node is pruned and the search backtracks in line 3. In line 5, function getCandidates generates the candidate list CL. If the candidate list is empty in line 6, then the search also backtracks in line 7. If the candidate list is not empty, line 9 checks if the extension of the current solution by any of its members improves the current lower bound. If this is the case, the lower bound LB is updated in line 10, a node $v \in CL$ is randomly selected in line 11, and the incumbent Π^* is updated by appending node v at the end of sequence Π . The loop in lines 14 to 17 creates a new permutation Π' for each candidate $j \in CL$ and makes a recursive call to QClique in line 16.

Algorithm 3 QClique

Require: $G, \Pi, \gamma, LB, \Pi^*$ 1: $i \leftarrow |\Pi|$ 2: if $|E(\Pi)| + \sum_{v \in F} \deg_{G(\Pi)}(v) + \min\{\frac{\sum_{v \in D} \deg_{G}(v) - \sum_{v \in F} \deg_{G(\Pi)}(v)}{2}, \binom{|V'|}{2}\} < \gamma \cdot \binom{k}{2}$ for every $k = i + 1, \ldots, LB + 1$ then 3: return 4: end if 5: $CL \leftarrow getCandidates(G, \Pi, \gamma)$ 6: if $CL = \emptyset$ then return 7: 8: end if 9: if $|\Pi| + 1 > LB$ then 10: $LB \leftarrow |\Pi| + 1$ Randomly select $v \in CL$ 11: $\Pi^* \leftarrow \Pi \oplus \{v\}$ 12:13: end if 14: for all $j \in CL$ do $\Pi' \leftarrow \Pi \oplus \{j\}$ 15:16: $\operatorname{QClique}(G, \Pi', \gamma, LB, \Pi^*)$ 17: end for

Example 1 Consider the graph G = (V, E) in Figure 1 (a) and let $\gamma = 0.8$. The enumeration tree produced by Algorithm 1 is represented in Figure 2. The root node represents the empty set. Each path from the root to a node of the tree represents a feasible solution, i.e. a γ -clique and a vertex permutation that satisfies the conditions of Theorem 1. Paths from the root to red nodes represent the maximum γ -clique. A depth-first search strategy was used and the nodes are visited according to the order they are placed in the candidate list (which is the same of their labels, in this case). Figure 3 shows the enumeration tree produced by algorithm QClique that explores pruning. Immediately after algorithm QClique discovers a solution of size 4, the value of LB is set to 5 in line 10. Next, all nodes of the search tree that are not able to lead to a solution of size 5 are pruned.

4 Computational experiments

All algorithms have been implemented using version 19.00.23504 of the Microsoft C/C++ Optimizing compiler. The computational experiments have been performed on an Intel Core i5-5200 processor with 2.20 GHz and 8 GB



Fig. 2 Enumeration tree produced by Algorithm 1.



Fig. 3 Enumeration tree by Algorithm 3 (QClique).

of RAM running under Windows 10. The test problems involved 50 randomly generated graphs, 25 instances derived from graph coloring problems, 16 instances derived from maximum clique problems of the Second DIMACS Implementation Challenge (Johnson, 1996) and four miscelaneous graphs from (Rossi and Ahmed, 2015).

The exact QClique algorithm is compared with the two best formulations in (Pattillo et al, 2013; Veremyev et al, 2016) solved by CPLEX and with the

branch-and-bound algorithm in (Pajouh et al, 2014). The MIP formulations were solved with version 12.6.2 of the CPLEX library for Visual Studio 2010. The solver was used with the default settings for preprocessing, branching strategies, node algorithms, heuristics and cutting planes. The upper bound proposed in Section 3.2 was used in the algorithm of Pajouh et al (2014) and in model F3.

4.1 Experiments on randomly generated graphs with 100 vertices

We have randomly generated ten instances with 100 vertices each with densities $\rho = 0.05$, 0.10, 0.25, 0.50, and 0.75. Each instance was solved by the branch and bound algorithm (B&B), by the two MIP models (F1 and F3) and by the proposed QClique algorithm. The time limit for each algorithm and instance was one hour.

The numerical results are presented in Table 1. Whenever any of the exact methods failed to prove optimality within the time limit for at least one instance in the group of same density, we indicate it by "> 3600". For each group of same density instances, the table shows the average running time in seconds over the ten instances, the number of instances solved to proved optimality within the time limit, and the average size of the best solution found within the time limit. The best average times and the best average solution sizes are highlighted in bold face.

The numerical results for randomly generated graphs show that algorithm QClique is much faster than the others and obtain same quality solutions for the instances with small densities. For the problems with larger densities, algorithm QClique outperformed the other approaches and found larger γ -cliques in smaller computation times, except for the case $\gamma = 0.85$ and $\rho = 0.75$.

4.2 Experiments on literature instances

We consider in this section numerical experiments on 25 instances derived from graph coloring problems available at http://mat.gsia.cmu.edu/COLORO4/, 16 instances derived from maximum clique problems of the Second DIMACS Implementation Challenge (Johnson, 1996; DIMACS, 2016) and four miscelaneous graphs from (Rossi and Ahmed, 2015). Except for the last nine coloring instances, all other instances were also used in the computational experiments reported in (Pattillo et al, 2013; Pajouh et al, 2014).

The number of vertices, edges, and the density of each instance appear in Table 2, together with the upper bound UB proposed in Section 3.2 and the upper bound ω^u given by equation (9) (Pattillo et al, 2013) for the connected graphs and by equation (8) (Pattillo et al, 2013) for instance miles250, which is not a connected graph. The best upper bounds for each instance are highlighted in bold face. The new pruning upper bound UB is shown to be

γ	ρ	B&B		F1		F3		QCliqu	ıe	B&B	F1	F3	QClique
		avg.time	opt.	avg.time	opt.	avg.time	opt.	avg.time	opt.	avg.size	avg.size	avg.size	avg.size
0.85	0.05	1552.036	10	40.508	10	0.120	10	0.003	10	4.8	4.8	4.8	4.8
	0.10	> 3600	0	247.763	10	3.425	10	0.027	10	7.7	7.7	7.7	7.7
	0.25	> 3600	0	> 3600	0	> 3600	0	1.599	10	11.0	10.7	9.6	11.2
	0.50	> 3600	0	> 3600	0	> 3600	0	422.690	10	14.8	13.3	10.5	16.1
	0.75	> 3600	0	> 3600	0	> 3600	0	> 3600	0	45.1	41.8	28.7	29.3
0.95	0.05	542.767	10	34.755	10	0.127	10	0.002	10	3.8	3.8	3.8	3.8
	0.10	2663.052	6	105.422	10	1.459	10	0.007	10	5.6	5.6	5.6	5.6
	0.25	> 3600	0	405.256	10	> 3600	0	0.071	10	7.8	7.8	7.0	7.8
	0.50	> 3600	0	> 3600	0	> 3600	0	1.411	10	10.1	10.2	7.1	10.6
	0.75	> 3600	0	> 3600	0	> 3600	0	> 3600	0	20.9	20.3	10.7	21.9
1.00	0.05	542.062	10	0.052	10	0.131	10	0.002	10	3.8	3.8	3.8	3.8
	0.10	2699.658	6	0.053	10	1.383	10	0.007	10	5.4	5.4	5.4	5.4
	0.25	> 3600	0	0.366	10	> 3600	0	0.064	10	7.1	7.1	6.5	7.1
	0.50	> 3600	0	17.159	10	> 3600	0	1.000	10	8.8	9.2	6.2	9.2
	0.75	> 3600	0	78.966	10	> 3600	0	1249.99	10	14.8	16.5	7.9	16.5

Table 1 Average results for randomly generated graphs with 100 vertices (ten instances for each threshold γ and each density ρ), times in seconds.

significantly tighter for all but one instance, for which the two bounds match (karate, for $\gamma = 0.85$).

Instance	vertices	edges	density	$\gamma = 1$		$\gamma =$	$\gamma = 0.95$		0.85
				ω^{u}	UB	ω^{u}	UB	ω^u	UB
1-FullIns_3	30	100	0.23	13	10	13	10	14	11
1 -FullIns_4	52	201	0.15	33	21	34	22	36	24
2-FullIns_3	80	346	0.11	18	13	19	13	20	14
3-FullIns_3	114	541	0.08	24	15	25	16	26	18
4-FullIns_3	154	792	0.07	30	18	31	19	33	21
5 -FullIns_3	93	593	0.14	37	21	38	22	40	24
games120	120	638	0.09	33	13	34	14	36	15
$mug88_1$	88	146	0.04	12	5	12	5	13	5
$mug88_25$	88	146	0.04	12	5	12	5	13	5
myciel3	11	20	0.36	6	5	6	5	6	5
myciel4	23	71	0.28	11	9	11	9	12	10
myciel5	47	236	0.22	21	16	21	16	22	17
myciel6	95	755	0.17	37	27	38	28	41	31
$queen5_5$	25	160	0.53	18	14	18	15	19	16
queen6_6	36	290	0.46	24	18	24	19	26	20
$queen7_7$	49	776	0.66	30	21	31	22	33	25
queen8_8	64	728	0.36	37	25	38	26	41	29
$queen 8_12$	96	1368	0.30	51	32	53	33	56	37
queen9_9	81	1056	0.32	45	29	46	30	49	34
queen10_10	100	1470	0.29	53	33	55	34	58	38
miles 250	128	387	0.047	28	14	29	14	30	16
miles500	128	1170	0.143	47	32	48	34	51	37
miles750	128	2113	0.25	64	49	66	51	70	56
miles1000	128	3216	0.39	80	67	82	70	86	76
miles1500	128	5198	0.63	102	94	104	98	110	107
c-fat200-1	200	1534	0.08	53	18	54	18	57	21
c-fat200-2	200	3235	0.16	79	34	81	36	86	40
c-fat200-5	200	8473	0.43	130	86	133	91	141	101
hamming6-4	64	704	0.35	37	23	38	24	40	26
hamming8-4	256	20864	0.64	204	164	209	172	221	192
johnson8-2-4	28	210	0.56	20	16	21	16	22	18
johnson8-4-4	70	1855	0.77	61	54	62	56	66	63
brock200-2	200	9876	0.50	140	104	144	110	152	122
brock200-1	200	14834	0.75	172	151	177	159	187	177
brock200-3	200	12048	0.61	155	125	159	131	168	146
brock200-4	200	13089	0.66	162	135	166	141	175	157
brock400-2	400	59786	0.75	346	303	355	318	375	354
brock400-4	400	59765	0.75	346	303	355	318	375	354
keller4	171	9435	0.65	137	115	141	120	149	133
p-hat300-1	300	10933	0.24	147	100	151	104	159	115
p-hat300-2	300	21928	0.49	209	177	214	185	227	201
adjnoun	112	425	0.07	26	19	27	20	28	21
dolphins	62	159	0.08	15	10	15	11	16	12
karate	34	78	0.14	11	10	11	10	11	11
polbooks	105	441	0.08	27	19	28	20	29	21

Table 2 Results for literature graphs: new pruning upper bound UB is significantly tighter.

Table 3 shows the experimental results for each instance for the threshold $\gamma = 0.85$. For each approach, the table displays the best solution found within

the time limit of one hour and the running time in seconds. As before, we indicate by "> 3600" whenever any of the exact methods failed to prove optimality within the time limit for some instance. We indicate by "< ϵ " whenever a running time is too close to zero. Concerning the coloring instances, QClique was faster than the other approaches for all but the four last instances, for which it did not reach the optimal solution within the time limit. QClique was also faster for the four last miscelaneous graphs. However, QClique did not perform well for the three last coloring instances (miles750, miles1000, miles1500) and for some clique instances (hamming-8-4, all brock instances but brock200-2, keller4, p-hat300-2).

Table 4 displays the same results for each instance for the threshold $\gamma = 0.95$. We notice that as the threshold γ increases, the sizes of the maximum γ -cliques and the running times decrease. The performance of algorithm QClique improves and it clearly outperforms the others. QClique found the best solutions or obtained the smallest running times for all but seven instances (miles750, miles1000, miles1500, brock200-1, brock400-2, brock400-4, p-hat300-2)

Finally, Table 5 gives the numerical results for each instance for the threshold $\gamma = 1.00$. The relative performance of algorithm QClique is even better than in the two previous cases. QClique has the smallest running times for all but four instances (c-fat200-5, johnson8-4-4, hamming8-4, miles500, miles750, miles1000, miles1500) and for only two instances another algorithm found a better solution than QClique (miles1000, p-hat300-2).

In general, we observe that the performance of algorithm QClique improves with the increase of the threshold γ , due to the effectiveness of the pruning procedure. When the graph density increases, the upper bounds provided by inequality (24) become less tight and the number of nodes that are pruned in the search tree decrease.

5 Concluding remarks

Given a graph G = (V(G), E(G)) and a threshold $\gamma \in (0, 1]$, the maximum quasi-clique problem consists in finding a maximum cardinality subset C^* of the vertices in V(G) such that the density of the graph induced in G by C^* is greater than or equal to the threshold γ .

We proposed an exact algorithm to solve this problem, based on a quasihereditary property. We also proposed a new upper bound that is used for pruning the search tree. Numerical results showed that the new approach is competitive with the best integer programming formulations in (Pattillo et al, 2013; Veremyev et al, 2016) solved by CPLEX and with the branch-and-bound algorithm proposed by Pajouh et al (2014), in terms of both solution quality and running time. In addition, the new upper bound is consistently tighter than previously existing bounds and leads to significant reductions in the enumeration tree.

		solution	size			time (se	conds)	
Instance	B&B	F1	F3	QC	B&B	F1	F3	QC
1-FullIns_3	3	3	3	3	14.438	1.516	0.094	$< \epsilon$
1-FullIns_4	3	3	3	3	1348.382	87.453	166.781	0.004
2-FullIns_3	5	5	5	5	> 3600	8.625	0.516	0.002
3-FullIns_3	7	7	7	7	> 3600	30.469	1.500	0.003
4-FullIns_3	8	8	8	8	> 3600	121.969	2.172	0.008
5-FullIns_3	9	10	10	10	> 3600	456.219	29.813	0.027
games120	10	10	10	10	> 3600	233.156	3.344	0.036
mug88_1	3	3	3	3	1076.580	30.625	0.125	0.001
mug88_25	3	3	3	3	1076.840	25.500	0.203	0.001
myciel3	2	2	2	2	0.022	0.047	0.016	$< \epsilon$
myciel4	2	2	2	2	0.923	0.672	0.063	$< \epsilon$
myciel5	2	2	2	2	8.605	8.266	0.578	$< \epsilon$
myciel6	2	2	2	2	64.520	429.578	328.984	0.001
queen5_5	6	6	6	6	272.487	1.453	2.359	0.005
queen6_6	7	7	7	7	> 3600	6.656	25.500	0.015
queen7_7	7	8	8	8	> 3600	20.906	> 3600	0.038
queen8_8	7	9	9	9	> 3600	69.500	> 3600	0.104
queen8_12	7	13	13	13	> 3600	3123.094	> 3600	1.398
queen9_9	7	10	10	10	> 3600	253.156	> 3600	0.300
queen10_10	7	11	11	11	> 3600	1105.188	> 3600	0.767
miles250	11	11	11	11	> 3600	149.234	0.125	0.038
miles500	30	30	30	30	0.085	555.422	0.688	> 3600
miles750	46	46	46	25	0.074	244.578	3.359	> 3600
miles1000	64	64	64	37	0.064	117.531	27.219	> 3600
miles1500	104	104	104	89	0.055	5.000	0.953	> 3600
c-fat200-1	14	14	14	14	> 3600	1637.370	9.047	6.487
c-fat200-2	29	29	29	29	> 3600	3476.420	235.453	> 3600
c-fat200-5	70	70	70	70	> 3600	> 3600	> 3600	> 3600
hamming6-4	4	4	4	4	> 3600	28.190	145.093	0.007
hamming8-4	35	11	8	23	> 3600	> 3600	> 3600	> 3600
johnson8-2-4	4	4	4	4	1.396	3.656	0.547	0.002
johnson8-4-4	21	22	21	27	> 3600	> 3600	> 3600	> 3600
brock200-2	15	13	7	17	> 3600	> 3600	> 3600	> 3600
brock200-1	63	36	21	27	> 3600	> 3600	> 3600	> 3600
brock200-3	25	17	11	24	> 3600	> 3600	> 3600	> 3600
brock200-4	34	14	9	24	> 3600	> 3600	> 3600	> 3600
brock400-2	94	0	1	16	> 3600	> 3600	> 3600	> 3600
brock400-4	93	0	1	16	> 3600	> 3600	> 3600	> 3600
keller4	23	26	10	24	> 3600	> 3600	> 3600	> 3600
p-hat300-1	11	7	7	12	> 3600	> 3600	> 3600	220.098
p-hat300-2	85	63	2	24	> 3600	> 3600	> 3600	> 3600
adjnoun	7	7	7	7	> 3600	109.609	0.922	0.007
dolphins	6	6	6	6	> 3600	12.328	0.125	0.002
karate	6	6	6	6	2857.420	1.703	0.016	$< \epsilon$
polbooks	9	9	9	9	> 3600	85.547	0.344	0.019

Table 3 Results for literature graphs with $\gamma = 0.85$.

Acknowledgements Work of Celso C. Ribeiro was partially supported by CNPq research grant 303958/2015-4 and by FAPERJ research grant E-26/201.198/2014. Work of Jose Angel Riveaux was supported by a CAPES scholarship. The authors are thankful to Ruben Interian for a critical revision of a preliminary version of this manuscript.

	solution size time (seconds)										
Instance	B&B	F1	F3	QC	B&B	F1	F3	QC			
1-FullIns_3	3	3	3	3	14.238	1.047	0.141	< 6			
1-FullIns_4	3	3	3	3	1323.496	53.969	145.484	0.002			
2-FullIns_3	4	4	4	4	1365.687	5.969	0.328	0.001			
3-FullIns_3	5	5	5	5	> 3600	29.578	3.922	0.001			
4-FullIns_3	7	7	7	7	> 3600	79.844	1.824	0.005			
5-FullIns_3	8	8	8	8	> 3600	246.375	6.172	0.008			
games120	7	9	9	9	> 3600	102.922	14.641	0.018			
$mug88_1$	3	3	3	3	1067.006	26.484	0.047	0.001			
mug88_25	3	3	3	3	1066.575	29.953	0.047	0.001			
myciel3	2	2	2	2	0.023	$< \epsilon$	0.016	$< \epsilon$			
myciel4	2	2	2	2	0.901	0.594	0.109	$< \epsilon$			
myciel5	2	2	2	2	8.665	5.063	0.719	$< \epsilon$			
myciel6	2	2	2	2	63.951	55.938	776.344	0.002			
queen5_5	5	5	5	5	95.780	1.219	1.047	0.002			
queen6_6	6	6	6	6	> 3600	4.016	12.656	0.006			
queen7_7	6	7	7	7	> 3600	8.828	353.781	0.015			
queen8_8	6	8	8	8	> 3600	22.844	> 3600	0.029			
queen8_12	7	12	12	12	> 3600	171.687	> 3600	0.225			
queen9_9	7	9	9	9	> 3600	58.359	> 3600	0.077			
queen10_10	7	10	10	10	> 3600	113.422	> 3600	0.247			
miles250	8	8	8	8	> 3600	83.563	0.297	0.008			
miles500	25	25	25	25	> 3600	175.859	2.531	381.172			
miles750	39	39	39	30	> 3600	230.375	5.063	> 3600			
miles1000	52	52	52	34	> 3600	55.937	2652.531	> 3600			
miles1500	90	90	90	88	0.061	4.859	129.922	> 3600			
c-fat200-1	12	12	12	12	> 3600	799.187	14.950	0.658			
c-fat200-2	25	25	25	25	> 3600	1132.500	386.421	1334.637			
c-fat200-5	61	61	61	61	> 3600	2145.391	> 3600	> 3600			
hamming6-4	4	4	4	4	> 3600	19.781	0.516	0.007			
hamming8-4	17	7	5	17	> 3600	> 3600	> 3600	> 3600			
johnson8-2-4	4	4	4	4	0.0160	2.156	0.625	0.002			
johnson8-4-4	12	15	8	15	> 3600	132.320	> 3600	191.986			
brock200-2	10	10	5	13	> 3600	> 3600	> 3600	105.461			
brock200-1	26	23	4	25	> 3600	> 3600	> 3600	> 3600			
brock200-3	15	16	7	17	> 3600	> 3600	> 3600	> 3600			
brock200-4	18	17	9	20	> 3600	> 3600	> 3600	> 3600			
brock400-2	34	8	1	31	> 3600	> 3600	> 3600	> 3600			
brock400-4	32	0	1	31	> 3600	> 3600	> 3600	> 3600			
keller4	10	13	7	15	> 3600	> 3600	> 3600	> 3600			
p-hat300-1	8	8	7	9	> 3600	> 3600	> 3600	3.456			
p-hat300-2	39	30	8	25	> 3600	> 3600	> 3600	> 3600			
adjnoun	5	5	5	5	> 3600	69.516	2.125	0.003			
dolphins	5	5	5	5	> 3600	11.656	0.110	0.001			
karate	5	5	5	5	711.772	1.969	0.031	$< \epsilon$			
polbooks	7	7	7	7	> 3600	58.234	0.438	0.007			

Table 4 Results for literature graphs with $\gamma = 0.95$.

References

Abello J, Resende M, Sudarsky S (2002) Massive quasi-clique detection. In: Abello J, Vitter J (eds) Proceedings of the 5th Latin American Symposium on the Theory of Informatics, Lecture Notes in Computer Science, vol 2286,

		solutio	n size			time (s	seconds)	
Instance	B&B	F1	F3	QC	B&B	F1	F3	QC
1-FullIns_3	3	3	3	3	0.864	0.016	0.078	$< \epsilon$
1-FullIns_4	3	3	3	3	243.552	0.125	275.813	0.004
2-FullIns_3	4	4	4	4	397.431	0.063	0.314	0.001
3-FullIns_3	5	5	5	5	> 3600	0.063	1.000	0.002
4-FullIns_3	6	6	6	6	> 3600	0.266	2.344	0.004
5-FullIns_3	7	7	7	7	> 3600	0.797	31.281	0.009
games120	7	9	9	9	> 3600	1.656	3.172	0.005
mug88_1	3	3	3	3	734.887	0.703	0.047	0.001
mug88_25	3	3	3	3	736.032	0.844	0.047	0.001
myciel3	2	2	2	2	0.007	$< \epsilon$	$< \epsilon$	$< \epsilon$
myciel4	2	2	2	2	0.307	$< \epsilon$	0.0469	$< \epsilon$
myciel5	2	2	2	2	6.083	0.063	1.172	0.001
myciel6	2	2	2	2	42.778	0.734	311.422	0.001
queen5_5	5	5	5	5	69.067	0.031	0.625	0.002
queen6_6	6	6	6	6	2886.369	0.047	7.750	0.005
$queen7_7$	7	7	7	7	> 3600	0.031	105.578	0.010
queen8_8	7	8	8	8	> 3600	0.156	> 3600	0.018
$queen8_{12}$	7	12	12	12	> 3600	0.453	> 3600	0.038
queen9_9	7	9	9	9	> 3600	0.438	> 3600	0.032
queen10_10	7	1	10	10	> 3600	0.891	> 3600	0.055
miles250	8	8	8	8	> 3600	2.453	0.281	0.006
miles500	20	20	20	20	> 3600	2.625	53.891	15.061
miles750	31	31	31	31	> 3600	2.641	3298.766	> 3600
miles1000	42	42	40	30	> 3600	2.516	> 3600	> 3600
miles1500	73	73	11	73	> 3600	1.766	> 3600	> 3600
c-fat200-1	12	12	12	12	> 3600	12.141	10.891	0.046
c-fat200-2	24	24	24	24	> 3600	11.359	89.109	1.504
c-fat200-5	58	58	58	58	> 3600	42.328	> 3600	> 3600
hamming6-4	4	4	4	4	> 3600	0.734	14.594	0.007
hamming8-4	16	16	5	16	> 3600	30.203	> 3600	> 3600
johnson8-2-4	4	4	4	4	296.073	0.015	0.281	0.002
johnson8-4-4	10	14	10	14	> 3600	0.218	TL	12.4930
brock200-2	9	12	6	12	> 3600	1530.718	> 3600	40.462
brock200-1	15	19	8	19	> 3600	> 3600	>3600	> 3600
brock200-3	10	13	6	15	> 3600	> 3600	> 3600	1154.621
brock200-4	12	15	5	16	> 3600	> 3600	> 3600	> 3600
brock400-2	18	19	1	21	> 3600	> 3600	> 3600	> 3600
brock400-4	18	16	1	22	> 3600	> 3600	> 3600	> 3600
keller4	8	11	5	11	> 3600	971.703	> 3600	420.978
p-hat300-1	8	8	5	8	> 3600	2134.547	> 3600	2.822
p-hat300-2	23	22	4	22	> 3600	> 3600	> 3600	> 3600
adjnoun	5	5	5	5	> 3600	0.453	1.438	0.004
dolphins	5	5	5	5	> 3600	0.094	0.266	0.001
karate	5	5	5	5	510.339	0.016	0.063	$< \epsilon$
polbooks	6	6	6	6	> 3600	0.813	0.828	0.006

Table 5 Results for literature graphs with $\gamma = 1.00$.

Springer, Berlin, pp 598–612

 DIMACS (2016) Dimacs implementation challenges. URL http://dimacs. rutgers.edu/Challenges/, online reference, last access on June 7, 2017.
 Johnson DS (ed) (1996) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Dimacs Series in Discrete Mathematics and Theoretical Computer Science, vol 26. American Mathematical Society, Providence

- Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of Computer Computations, Plenum, New York, pp 85–103
- Kreher DL, Stinson DR (1998) Combinatorial Algorithms: Generation, Enumeration, and Search. CRC Press, New York
- Nemhauser GL, Wolsey LA (1988) Integer programming and combinatorial optimization. Wiley, New York
- Pajouh FM, Miao Z, Balasundaram B (2014) A branch-and-bound approach for maximum quasi-cliques. Annals of Operations Research 216:145–161
- Pattillo J, Veremyev A, Butenko S, Boginski V (2013) On the maximum quasiclique problem. Discrete Applied Mathematics 161:244–257
- Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Association for the Advancement of Artificial Intelligence, Austin, pp 4292–4293, URL http: //networkrepository.com, online reference, last access on June 7, 2017.
- Veremyev A, Prokopyev OA, Butenko S, Pasiliao EL (2016) Exact MIP-based approaches for finding maximum quasi-clique and dense subgraph. Computational Optimization and Applications 64:177–214

APÊNDICE B – Improving a biased random-key genetic algorithm for the maximum quasi-clique problem with an exact local search Intl. Trans. in Op. Res. 25 (2018) 1–20 DOI: xx.xxxx/itor.xxxxx INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Improving a biased random-key genetic algorithm for the maximum quasi-clique problem with an exact local search

Bruno Q. Pinto^a, Celso C. Ribeiro^{a,}, Jose A. Riveaux^{a,*}, and Isabel Rosseti^a

^aInstitute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil. E-mail: jangel.riveaux@ic.uff.br [Jose]; celso@ic.uff.br [Celso]; bruno.queiroz@iftm.edu.br [Bruno]; rosseti@ic.uff.br [Isabel]

Received 30 June 2017; received in revised form XXXX; accepted XXXX

Abstract

Given a graph G = (V, E) and a threshold $\gamma \in (0, 1]$, the maximum cardinality quasi-clique problem consists in finding a maximum cardinality subset C^* of the vertices in V such that the density of the graph induced in G by C^* is greater than or equal to the threshold γ . This problem is NP-hard, since it admits the maximum clique problem as a special case. It has a number of applications in data mining, e.g. in social networks or phone call graphs. In this work, we propose a hybrid biased random-key genetic algorithm (BRKGA) for solving the maximum cardinality quasi-clique problem. The hybrid approach makes use of the QClique algorithm of Ribeiro and Riveaux (2017) to exactly solve the local search procedure in the reconstruction phase of the decoder. The newly proposed approach is compared with algorithm BRKGA-IG^{*} of Pinto et al. (2017), the best heuristic in the literature at the time of writing. Computational results show that the hybrid BRKGA outperforms BRKGA-IG^{*}.

Keywords: maximum cardinality quasi-clique problem; maximum clique problem; maximum clique problem; biased randomkey genetic algorithm; metaheuristics; graph density

1. Introduction

Let G = (V, E) be a graph defined by a vertex set V and an edge set $E \subseteq V \times V$. G is a complete graph if there is an edge in E connecting every two different vertices in V. A graph G' = (V', E') is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$, which is denoted by $G' \subseteq G$. The graph G(V') induced in G by $V' \subseteq V$ is that with vertex set V' and edge set formed by all edges of E with both ends in V'. For any $V' \subseteq V$, the subset $E(V') \subseteq E$ is formed by all edges of E with both ends in V' is the edge set of the graph induced in G by V').

*Author to whom all correspondence should be addressed (e-mail: jangel.riveaux@ic.uff.br).

^oWork of Celso C. Ribeiro was partially supported by CNPq research grant 303958/2015-4 and by FAPERJ research grant E-26/201.198/2014.

The density of graph G is given by $dens(G) = |E|/(|V| \times (|V| - 1)/2)$. For any $v \in V$, the degree $deg_G(v)$ denotes the number of vertices in G that are adjacent to v.

A subset $C \subseteq V$ is a clique of G if the graph G(C) induced in G by C is complete. Given a graph G = (V, E), the *maximum clique problem* consists in finding a maximum cardinality clique of G. It was proved to be NP-hard by Karp (1972).

Given a graph G = (V, E) and a threshold $\gamma \in (0, 1]$, a γ -clique is any subset $C \subseteq V$ such that the density of the subgraph G(C) is greater than or equal to γ . A γ -clique C is maximal if there is no other γ -clique C' that strictly contains C. The maximum quasi-clique problem (MQCP) amounts to finding a maximum cardinality subset C^* of the vertices in V such that the density of the graph induced in G by C^* is greater than or equal to the threshold γ . This problem is also NP-hard, since it admits the maximum clique problem as a special case in which $\gamma = 1$, see (Pattillo et al., 2013). The problem has many applications and related clustering approaches include classifying molecular sequences in genome projects by using a linkage graph of their pairwise similarities (Brunato et al., 2008) and the analysis of massive telecommunication data sets obtained from social networks or phone call graphs (Abello et al., 2002), as well as various data mining and graph mining applications.

A few heuristics for MQCP exist in the literature, based on well known approaches such as greedy randomized algorithms and their iterated extensions (Oliveira, 2013; Oliveira et al., 2013), stochastic local search (Brunato et al., 2008), and GRASP (Abello et al., 2002). Pinto et al. (2017) proposed a biased random-key genetic algorithm for finding approximate solutions to the maximum cardinality quasi-clique problem, using two different decoders. They showed that the decoder based on an optimized iterated greedy constructive heuristic led to the best numerical results. They also showed that the use of a restart strategy significantly contributed to improve the robustness and the efficiency of the algorithm. The resulting BRKGA-IG* heuristic with restart(100) strategy achieved the best performance and outperformed the restarted optimized iterated greedy (RIG*) construction/destruction heuristic that originally reported the best results in the literature for dense graphs. BRKGA-IG* with restart(100) approach was also compared with the exact algorithms AlgF3 and AlgF4 of Veremyev et al. (2016) used as a heuristics with time limits on their running times. BRKGA-IG* with restart(100) applied to sparse graphs also outperformed these mixed integer programming approaches, finding target solution values in much smaller running times.

Ribeiro and Riveaux (2017) proposed an exact enumeration algorithm to solve the maximum quasiclique problem, based on a quasi-hereditary property. They also proposed a new upper bound that is used for pruning the search tree. Numerical results showed that their approach is competitive with the best integer programming formulations in (Pattillo et al., 2013; Veremyev et al., 2016) solved by CPLEX and with the branch-and-bound algorithm proposed by Pajouh et al. (2014), in terms of both solution quality and running time.

In this work, we show that the exact enumeration algorithm QClique proposed by Ribeiro and Riveaux (2017) can be hybridized with the biased random-key genetic algorithm BRKGA-IG* developed by (Pinto et al., 2017) as a local search algorithm to improve the quality of the solutions created by the decoder. This paper is organized as follows. Section 2 presents the problem formulation. Section 3 introduces biased random-key genetic algorithms and describes their customization to the maximum quasi-clique problem. Section 4 describes in detail the decoder DECODER-IG* previously used used in the implementation of the biased random-key genetic algorithm for the maximum quasi-clique problem. The new decoder DECODER-ExactQClique based on an exact local search procedure is presented in

© 2017 International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

2

Section 5. Numerical results are reported in Section 6. Concluding remarks are drawn in the last section.

2. Problem formulation and related work

The maximum quasi-clique problem can be formulated by associating a binary variable x_i to each vertex of the graph (Pattillo et al., 2013) :

$$x_i = \begin{cases} 1, & \text{if vertex } v_i \in V \text{ belongs to the solution,} \\ 0, & \text{otherwise.} \end{cases}$$

This formulation also considers a variable $y_{ij} = x_i \cdot x_j$ associated to each pair of vertices $i, j \in V$, with i < j, that is linearized as follows:

$$\max\sum_{i\in V} x_i \tag{1}$$

subject to:

$$\sum_{(i,j)\in E:i$$

$$y_{ij} \le x_i, \qquad \forall i, j \in V, \quad i < j,$$
(3)

$$y_{ij} \le x_j, \qquad \forall i, j \in V, \quad i < j,$$
(4)

$$y_{ij} \ge x_i + x_j - 1, \qquad i, j = 1, \dots, n, \quad i < j,$$
(5)

$$x_i \in \{0, 1\}, \qquad \forall i \in V, \tag{6}$$

$$y_{ij} \ge 0, \qquad \forall i, j \in V, \quad i < j. \tag{7}$$

The objective function (1) maximizes the number of vertices in the solution. If two vertices i, j belong to a solution, then $x_i = x_j = 1$ and $y_{ij} = x_i \cdot x_j = 1$. If edge $(i, j) \in E$, then it contributes to the density of the quasi-clique. Constraint (2) ensures that the density of the solution is greater than or equal to γ . Constraints (3) and (4) ensure that any edge may contribute to the density of a solution only if both of its ends are chosen to belong to this solution. Constraints (5) ensure that any existing edge $(i, j) \in E$ will contribute to the solution if both of its ends are chosen. Constraints (6) and (7) impose the binary and non-negativity requirements on the problem variables, respectively.

Veremyev et al. (2016) reported and compared four mixed integer programming formulations for the maximum quasi-clique problem in sparse graphs. Two algorithms based on the best formulations led to better results than the mixed integer programming formulation proposed in (Pattillo et al., 2013), with all mixed integer programs solved using FICO Xpress-Optimizer (FICO, 2017) with the time limit of 3600 seconds. Ribeiro and Riveaux (2017) developed an exact algorithm based on a quasi-hereditary property and proposed a new upper bound that is used for pruning the search tree. Numerical results showed that their approach is competitive with the best integer programming approaches in the literature and that their new upper bound is consistently tighter than previously existing bounds.

^{© 2017} International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

B.Q. Pinto, C.C. Ribeiro, J.A. Riveaux, and I. Rosseti / Intl. Trans. in Op. Res. 25 (2018) 1-20

3. Biased random-key genetic algorithms for maximum quasi-clique

Genetic algorithms with random keys, or random-key genetic algorithms (RKGA), were first introduced by Bean (1994) for combinatorial optimization problems whose solutions may be represented by permutation vectors. Solutions are represented as vectors of randomly generated real numbers called keys. A deterministic algorithm, called a decoder, takes as input a solution vector and associates with it a feasible solution of the combinatorial optimization problem, for which an objective value or fitness can be computed. Two parents are selected at random from the entire population to implement the crossover operation in the implementation of an RKGA. Parents are allowed to be selected for mating more than once in the same generation.

A biased random-key genetic algorithm (BRKGA) differs from an RKGA in the way parents are selected for crossover, see (Goncalves and Resende, 2011) for a review. In a BRKGA, each element is generated combining one element selected at random from the elite solutions in the current population, while the other is a non-elite solution. The selection is said to be biased because one parent is always an elite solution and has a higher probability of passing its genes to the new generation.

In the following, we summarize two variants of a biased random-key genetic algorithm for MQCP, each of them using a different decoder. Both of them evolve a population of chromosomes that consists of vectors of real numbers. Each chromosome is represented by a vector of |V| components, in which each key is a real number in the range [0, 1) associated with one of the vertices of the graph G. Each chromosome is decoded by an algorithm that receives the vector of keys and builds a feasible solution for MQCP, i.e., the decoder returns a γ -clique as its output. The two decoders DECODER-HCB and DECODER-IG^{*} are described in the next section and have been originally presented in (Pinto et al., 2017).

The parametric uniform crossover scheme proposed by Spears and de Jong (1991) is used to combine two parent solutions and to produce an offspring. In this scheme, the offspring inherits each of its keys from the best fit of the two parents with a higher probability. The biased random-key genetic algorithm developed in this work does not make use of the standard mutation operator, where parts of the chromosomes are changed with small probability. Instead, the concept of mutants is used: mutant solutions are introduced in the population in each generation, randomly generated in the same way as in the initial population. Mutants play the same role of the mutation operator in traditional genetic algorithms, diversifying the search and helping the procedure to escape from locally optimal solutions (Brandão et al., 2015, 2017; Noronha et al., 2011).

The |V| keys in the chromosome are randomly generated in the initial population. At each generation, the population is partitioned into two sets: TOP and REST. The size of the population is |TOP| + |REST|. Subset TOP contains the best solutions in the population. Subset REST is formed by two disjoint subsets: MID and BOT, with subset BOT being formed by the worst elements in the current population. As illustrated in Figure 1, the chromosomes in TOP are simply copied to the population of the next generation. The elements in BOT are replaced by newly created mutants that are placed in the new set BOT. The remaining elements of the new population are obtained by crossover, with one parent randomly chosen from TOP and the other from REST. This distinguishes a biased random-key genetic algorithm from the random-key genetic algorithm of Bean (1994), where both parents are selected at random from the entire population. Since a parent solution can be chosen for crossover more than once in any given generation, elite solutions have a higher probability of passing their random keys to the next

© 2017 International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

4



Fig. 1: Population evolution between consecutive generations of a BRKGA.

generation. In this way, |MID| = |REST| - |BOT| offspring solutions are created.

The implementations of the biased random-key genetic algorithms for the maximum quasi-clique problem make use of the C++ library brkgaAPI developed by Toso and Resende (2015), which is a framework for the development of biased random-key genetic algorithms. It can also be used in parallel architectures running OpenMP.

The instantiation of the framework shown in Figure 2 to some specific optimization problem requires exclusively the development of a class implementing the decoder for this problem. This is the only problem-dependent part of the tool.



Fig. 2: BRKGA framework.

According to Goncalves et al. (2013), the BRKGA framework requires the following parameters: (a) the population size (p = |TOP| + |REST|); (b) the fraction pe of the population corresponding to the elite set TOP; (c) the fraction pm of the population corresponding to the mutant set BOT; (d) the probability *rhoe* that the offspring inherits each of its keys from the best fit of the two parents; and (e) the number k of generations without improvement in the best solution until a restart is performed.

In the remainder of this work, we consider and compare two variants of a biased random-key genetic algorithm for solving MQCP, each of them based on a different decoder. Decoder DECODER-IG* was originally proposed by Pinto et al. (2017) and will be summarized in the next section. The new decoder DECODER-ExactQClique proposed in this work is based on the exact enumeration algorithm proposed by Ribeiro and Riveaux (2017) and will be presented in Section 5.

4. Decoder DECODER-IG*

We first review decoder DECODER-HCB, as presented in (Pinto et al., 2017). Each solution is associated with a set of |V| random keys. The decoder receives as parameters the random keys $r_j \in [0, 1), j = 1, \ldots, |V|$. Each random key is a real number in the range [0, 1) and corresponds to a vertex of the graph. Each chromosome represented by a set of random keys is decoded by an algorithm that receives the keys and builds a feasible solution to MQCP. In other words, the decoder returns a γ -clique associated with the set of random keys. Its pseudo-code is described in Algorithm 1. The roles of parameters *minsize* and α are the same explained in (Pinto et al., 2017).

It may be used in two situations. First, to build a solution from scratch. Second, to complete (i.e., to reconstruct) a partially destroyed solution. In the second case, the decoder receives as an additional parameter a partial solution formed by a non-empty list of vertices.

Decoder DECODER-IG^{*} is an extension of the previous decoder that receives as parameters two sets of random keys $r_j^1, r_j^2 \in [0, 1), j = 1, ..., |V|$, i.e., there are two random keys r_j^1 and r_j^2 associated with each vertex $j \in V$. The first set r^1 of random keys is used in the construction of the initial solution and in the reconstruction phase, while the second set r^2 is used in the destruction phase. The roles of the other parameters *minsize*, α , δ , and β are explained in (Pinto et al., 2017).

The pseudo-code of Algorithm 2 starts by creating an initial solution S' in line 1, using the decoder DECODER-HCB and the random keys $r_j^1, j \in V$. The loop in lines 2 to 10 repeats the partial destruction (vertex eliminations) followed by the reconstruction (vertex insertions) of the current solution, until no further improvements can be obtained. The current solution S' is copied to S in line 2. The current solution S' is copied to S in line 3. The loop in lines 4 to 8 removes one by one the $\delta \cdot |S'|$ vertices that should be eliminated from the current solution. A restricted candidate RCL of size max{minsize, $\beta \cdot |CL|$ } is created in line 5, containing the vertices with the smallest degrees in G(S'). The vertex with the smallest random key $r_j^2, j \in RCL$, is selected from the restricted candidate list in line 6 and eliminated from the current solution in line 7. The reconstruction phase is performed in line 9, where the current, partial solution S' is rebuilt by decoder DECODER-HCB, once again using the first set r^1 of random keys. The loop is interrupted in line 10 when the new solution S' obtained by destruction-reconstruction does not improve the incumbent S or the graph G(S') is not connected; otherwise a new iteration resumes. The best solution S is returned in line 11.

6

^{© 2017} International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

Algorithm 1 DECODER-HCB($G, \gamma, \alpha, minsize, S, r$)

1: $CL \leftarrow V \setminus S$ 2: if $S = \emptyset$ then $RCL \leftarrow \{v \in CL : |\{v' \in CL : deg_G(v') \ge deg_G(v)\}| \le \max\{li, \alpha \cdot |CL|\}\}$ 3: 4: $x \leftarrow argmin\{r_j : j \in RCL\}$ $S \leftarrow \{x\}$ 5: 6: **end if** 7: while $CL \neq \emptyset$ do $CL \leftarrow \emptyset$ 8: for all $v \in V \setminus S$ do 9: if $\frac{|E(S)| + deg_{G(S)}(v)}{|S| \cdot (|S|+1)/2} \ge \gamma$ then 10: 11: $CL \leftarrow CL \cup \{v\}$ 12: end if end for 13: 14: if $CL \neq \emptyset$ then for all $v \in CL$ do 15: $dif_{v} \leftarrow deg_{G(CL)}(v) + |CL| \cdot (deg_{G(S)}(v) - \gamma \cdot (|S| + 1))$ 16: end for 17: $RCL \leftarrow \{v \in CL : |\{v' \in CL : dif(v') \ge dif(v)\}| \le \max\{minsize, \alpha \cdot |CL|\}\}$ 18: $x \leftarrow argmin\{r_j : j \in RCL\}$ 19: $S \leftarrow S \cup \{x\}$ 20: end if 21: 22: end while 23: **return** *S*

Algorithm 2 DECODER-IG* $(G, \gamma, \alpha, \delta, \beta, minsize, S, r^1, r^2)$

1: $S' \leftarrow \text{DECODER-HCB}(G, \gamma, \alpha, minsize, \emptyset, r^1)$ 2: repeat $S \leftarrow S'$ 3: for k = 1 to $\delta \cdot |S'|$ do 4: $RCL \leftarrow \{v \in S' : |\{v' \in S' : deg_{G(S')}(v') \le deg_{G(S')}(v)\}| \le \max\{minsize, \beta \cdot |S'|\}\}$ 5: $x \leftarrow argmin\{r_j^2 : j \in RCL\}$ 6: $S' \leftarrow \check{S'} \setminus \{x\}$ 7: end for 8: $S' \leftarrow \mathsf{DECODER}\text{-}\mathsf{HCB}(G, \gamma, \alpha, \textit{minsize}, S', r^1)$ 9: 10: **until** $|S'| \leq |S|$ or graph G(S') is not connected 11: return S

5. New decoder based on exact local search: DECODER-ExactQClique

The main idea of the new decoder consists in replacing the reconstruction phase of decoder DECODER-IG^{*} by an exact algorithm that optimally completes the partial solution S' obtained at the exit of the loop

in lines 4 to 8 of Algorithm 2, obtaining a maximal γ -clique S'' that contains S'.

In order to reduce the number of times the exact search algorithm is applied, an additional, (2|V|+1)th random key $r^3 \in [0, 1)$ will be associated to each solution. Given a parameter $\rho \in [0, 1]$, local search will be applied to a solution whenever $r^3 \ge \rho$, otherwise the decoder DECODER-HCB will be used to reconstruct the solution. We note that, if $\rho = 0$ then the exact search algorithm will always be executed; if $\rho = 1$, then it will never be executed.

A slightly modified version of the QClique algorithm of Ribeiro and Riveaux (2017) will be used to complete the partial solution S' obtaining a maximal γ -clique.

Algorithm 3 describes the pseudo-code of decoder DECODER-ExactQClique. It requires the same parameters as DECODER-IG^{*}, except for for two new parameters: the probability ρ that the exact search algorithm is applied to each partial solution and the random key r^3 . The algorithm starts by creating an initial solution S' in line 1, using the decoder DECODER-HCB and the random keys $r_i^1, j \in V$. The loop in lines 2 to 14 repeats the partial destruction (vertex eliminations) followed by the reconstruction (vertex insertions) of the current solution, until no further improvements can be obtained. The current solution S' is copied to S in line 3. The loop in lines 4 to 8 removes one by one the $\delta \cdot |S'|$ vertices that should be eliminated from the current solution. A restricted candidate RCL of size max{minsize, $\beta \cdot |CL|$ } is created in line 5, containing the vertices with the smallest degrees in G(S'). The vertex with the smallest random key $r_i^2, j \in RCL$, is selected in line 6 and eliminated from the current solution in line 7. The reconstruction phase starts in line 9. If the random key r^3 is greater than or equal to parameter ρ and G(S') is a γ -clique, then the partial solution S' is completed in line 10 by an exact search algorithm to become a maximal γ -clique S" that contains S'. Solution S" is copied to S' in line 11. Solution S' is rebuilt by decoder DECODER-HCB in line 13, once again using the first set r^1 of random keys. The loop is interrupted in line 14 when the new solution S' obtained by destruction-reconstruction does not improve the incumbent S or the graph G(S') is not connected; otherwise a new iteration resumes. The best solution S is returned in line 15.

Algorithm 3 DECODER-ExactQClique($G, \gamma, \alpha, \delta, \beta, minsize, r^1, r^2, r^3, \rho$)

1: $S' \leftarrow \text{DECODER-HCB}(G, \gamma, \alpha, minsize, \emptyset, r^1)$ 2: repeat $S \leftarrow S'$ 3: for k = 1 to $\delta \cdot |S'|$ do 4: $RCL \leftarrow \{v \in S' : |\{v' \in S' : deg_{G(S')}(v') \le deg_{G(S')}(v)\}| \le \max\{minsize, \beta \cdot |S'|\}\}$ 5: $x \leftarrow argmin\{r_j^2 : j \in RCL\}$ $S' \leftarrow S' \setminus \{x\}$ 6: 7: end for 8: if $r^3 > \rho$ and $dens(G(S')) > \gamma$ then 9: Find a maximum cardinality quasi-clique S'' containing all vertices in S'. 10: $S' \leftarrow S''$ 11: end if 12: $S' \leftarrow \mathsf{DECODER}\text{-HCB}(G, \gamma, \alpha, minsize, S', r^1)$ 13: 14: **until** $|S'| \leq |S|$ or graph G(S') is not connected 15: return S

© 2017 International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

8

We now proceed to describe how the exact algorithm QClique (Ribeiro and Riveaux, 2017) can be adapted to be used in line 10 of Algorithm 3 to optimally complete the partial solution S', generating a maximal γ -clique S''.

Algorithm 4 QClique($G, \Pi, \gamma, LB, \Pi^*$) (Ribeiro and Riveaux, 2017)

1: $i \leftarrow |\Pi|$ 2: if $|E(\Pi)| + \sum_{v \in F} deg_{G(\Pi)}(v) + \min\{\frac{\sum_{v \in D} deg_G(v) - \sum_{v \in F} deg_{G(\Pi)}(v)}{2}, \binom{|V'|}{2}\} < \gamma \cdot \binom{k}{2}$ for every $k = i + 1, \dots, LB + 1$ then 3: return 4: end if 5: $CL \leftarrow qetCandidates(G, \Pi, \gamma)$ 6: if $CL = \emptyset$ then return 7: 8: end if 9: if $|\Pi| + 1 > LB$ then $LB \leftarrow |\Pi| + 1$ 10: Randomly select $v \in CL$ 11: $\Pi^* \leftarrow \Pi \oplus \{v\}$ 12. 13: end if 14: for all $j \in CL$ do $\Pi' \leftarrow \Pi \oplus \{j\}$ 15: $QClique(G, \Pi', \gamma, LB, \Pi^*)$ 16: 17: end for

The partial solution S' in line 10 of Algorithm 3 is passed to the pseudocode of Algorithm 4 as the set of vertices that will be used as the initial prefix Π , i.e. $\Pi = S'$. The best known solution Π^* is the current solution S, and the initial lower bound LB = |S|.

We recall that two restrictions were used to guarantee the uniqueness of a permutation in the enumeration tree generated by the original QClique algorithm, allowing to speedup the search by pruning. However, these restrictions might exclude some high quality solutions when the search starts from a set of fixed vertices.

In order to avoid the exclusion of such solutions, the uniqueness conditions are replaced by the requirement that $dens(G(\Pi)) \ge dens(G(\Pi'))$ in the *getCandidates* method that defines the vertices that may be used to extend the current solution Π , ensuring that the conditions of Theorem 1 in (Ribeiro and Riveaux, 2017) are met to form a new solution Π' . In addition, as an strategy to improve the search process, the candidate vertices are taken in the non increasing order of their number of neighbors in the current solution. In case of ties, a vertex with maximum degree goes first.

Algorithm 5 displays the pseudocode of the *getCandidates* function that builds the candidate list CL formed by the vertices that can be added at the end of a prefix solution. Line 1 initializes CL as empty and line 2 initializes i as the size of the current permutation Π . The loop in lines 3 to 8 considers the addition to the current permutation Π of all vertices in $V \setminus \Pi$. We denote by $deg_{G(\Pi)}(j)$ the number of vertices of the current permutation Π that are adjacent to vertex j. Line 4 checks if, for a candidate vertex j, the subgraph induced by $\Pi' = \Pi \cup \{j\}$ is a γ -clique and the density of the induced subgraph $G(\Pi')$

is not greater than that of $G(\Pi)$. If this is the case, then vertex j is definitely added to the candidate list CL in line 6. The vertices in the candidate list CL are sorted in the non increasing order of their number of neighbors in the current solution in line 9 and returned in line 10.

Algorithm 5 getCandidates(G, Π, γ)

1: $CL \leftarrow \emptyset$ 2: $i \leftarrow |\Pi|$ 3: for all $j \in V \setminus \Pi$ do 4: $\Pi' \leftarrow \Pi \oplus \{j\}$ 5: if $deg_{G(\Pi)}(j) + |E(\Pi)| \ge \gamma(i+1)i/2$ and $dens(G(\Pi)) \ge dens(G(\Pi'))$ then 6: $CL \leftarrow CL \cup \{j\}$ 7: end if 8: end for 9: sort vertices in the candidate list CL by the non-decreasing order of their number of neighbors 10: return CL

6. Computational results

All algorithms were implemented using version 19.00.23504 of the Microsoft C/C++ Optimizing compiler. The computational experiments have been performed on an Intel Core i5-5200 processor with 2.20 GHz and 8 GB of RAM running under Windows 10.

We have used 96 instances derived from maximum clique problems of the Second DIMACS Implementation Challenge (Johnson, 1996).

The newly proposed algorithm BRKGA-ExactQClique was compared with the original BRKGA-IG^{*} heuristic of Pinto et al. (2017), which was the best heuristic for the maximum quasi-clique problem at the time of writing. The best parameters for algorithm BRKGA-IG^{*} were determined by Pinto et al. (2017) using the automatic tuning tool IRACE (López-Ibánez et al., 2011; Pérez Cáceres et al., 2014). The same settings were used for algorithm BRKGA-ExactQClique. Parameters settings for both algorithm are shown in Table 1. A parameter *maxnodes* is used as a maximum limit to the number of nodes of the search tree generated by the exact algorithm QClique. In case the search tree generated by the QClique algorithm reaches the maximum limit *maxnodes* of nodes, then it returns the best solution found until this point. The additional application of the DECODER-HCB reconstruction in line 13 of Algorithm 3 is used to further improve the current solution.

The number of vertices deleted in the destruction phase of DECODER-IG^{*} of Algorithm 2 depends on the value of a parameter $\delta \in [0, 1]$. In order to to enforce that the fraction of nodes eliminated in the destruction phase is smaller for dense graphs, we empirically set $\delta = 1 - dens(G)$ if dens(G) < 0.8; $\delta = 2(1 - dens(G))$ otherwise. We note that no vertices will be removed if dens(G) = 1, i.e G is a clique (or a complete graph). However, in this case, the condition in line 10 of Algorithm 1 is true for every $S \subset V$ and every $v \in V \setminus S$. Therefore, in this case DECODER-HCB always finds the optimal solution S = V, which is the optimum for every complete graph G.

Tables 2 to 4 display average results over ten runs of each algorithm for each instance. A target value

10

Algorithm	p	p_e	p_m	rhoe	α	δ	β	maxnodes	ρ
BRKGA-IG*	89	0.16	0.11	0.77	0.01	0.34	0.10	-	-
BRKGA-ExactQClique	89	0.16	0.11	0.77	0.01	-	0.10	131	0.12

Table 1: Parameters settings.

is given to each instance. Each run stops when a solution at least as good as the target is found or a time limit of ten minutes is reached. For each instance, the table presents the threshold γ , the target for the size of the γ -clique, the average clique size found by algorithm BRKGA-IG^{*} and the number of runs it matched or improved the target over the ten runs, the average clique size found by algorithm BRKGA-ExactQClique and the number of runs it matched or improved the target over the ten runs. Cells highlighted in boldface indicate the algorithms that attained the best values for each instance.

We observe in these tables that BRKGA-ExactQClique found strictly larger average γ -cliques than BRKGA-IG^{*} for 28 instances, while BRKGA-IG^{*} did better than BRKGA-ExactQClique in only ten instances. Regarding the number of runs for which each algorithm matched or improved the target, BRKGA-ExactQClique did better than BRKGA-IG^{*} in 12 instances, while BRKGA-IG^{*} performed better than BRKGA-ExactQClique in only two instances. Although BRKGA-ExactQClique clearly outperformed BRKGA-IG^{*} in terms of solution quality, the latter was faster in 55 out of the 96 test instances.

In the next experiment, we evaluate and compare the run time distributions (or time-to-target plots – or ttt-plots, for short) of algorithms BRKGA-IG^{*} and BRKGA-ExactQClique. Time-to-target plots display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis. Run time distributions have also been advocated by Hoos and Stützle (1998) as a way to characterize the running times of stochastic local search algorithms for combinatorial optimization. In this experiment, the two algorithms were made to stop whenever a solution with cost greater than or equal to a given target value was found. The targets are the same used in the previous experiment and reported in Tables 2 to 4. The heuristics were run 200 times each, with different initial seeds for the pseudo-random number generator. Next, the empirical probability distributions of the time taken by each heuristic to find a target solution value are plotted. To plot the empirical distribution for each heuristic, we followed the methodology proposed by Aiex et al. Aiex et al. (2002, 2007). We associate a probability $p_i = (i - \frac{1}{2})/200$ with the *i*-th smallest running time t_i and plot the points (t_i, p_i) , for $i = 1, \ldots, 200$. The more to the left is a plot, the better is the algorithm corresponding to it.

Figures 3 to 6 illustrate the time-to-target plots for instances DSJC500.5, frb45-21-1, frb30-15-2, and frb30-15-5. These plots show that BRKGA-ExactQClique performed better for the two first instances, while BRKGA-IG* performed better for the two last ones. These conclusions are consistent with the results observed for each algorithm over the 200 runs that generated the time to target plots, as depicted in Table 5. This table shows that even though algorithm BRKGA-ExactQClique matches or improves BRKGA-IG* in terms of the average solution quality, the latter is faster than the former in terms of their average running times.

Figures 7 and 8 illustrate the evolution of the solution population along 100 generations of BRKGA-

Instances	γ	target	IG*	#opt.	QClique	#opt.	IG^*	QClique
		size					(seconds)	(seconds)
C125.9	0.999	34	34.00	10	34.00	10	0.052	0.207
C250.9	0.999	44	44.00	10	44.00	10	0.213	0.987
C500.9	0.999	57	57.00	10	57.00	10	7.717	41.596
C1000.9	0.999	67	67.00	10	66.80	8	149.145	181.719
C2000.9	0.999	74	73.60	6	74.10	7	456.268	369.326
C4000.5	0.8	46	43.20	0	47.50	10	648.695	220.310
DSJC500.5	0.8	34	34.00	10	34.00	10	22.445	16.735
DSJC1000.5	0.8	38	38.10	10	38.10	10	67.373	63.808
MANN_a9	0.999	16	16.00	10	16.00	10	0.013	0.023
MANN_a27	0.999	133	133.00	10	133.00	10	1.869	14.940
MANN_a45	0.999	428	427.20	6	427.60	8	360.498	324.388
brock200_1	0.8	114	114.00	10	114.00	10	0.511	1.211
brock200_2	0.8	24	24.00	10	24.00	10	1.027	1.003
brock200_3	0.8	41	41.00	10	41.00	10	0.452	0.964
brock400_1	0.8	189	189.00	10	189.00	10	5.488	12.592
brock400_2	0.8	186	186.00	10	186.00	10	6.822	12.548
brock400_3	0.8	187	187.00	10	187.00	10	40.283	31.571
brock800_1	0.8	94	93.70	8	94.40	10	163.726	38.846
brock800_2	0.8	93	93.00	10	93.10	10	99.053	41.638
brock800_3	0.8	92	92.00	10	92.00	10	118.929	63.718
c-fat200-1	0.5	30	30.00	10	30.00	10	0.032	0.310
c-fat200-2	0.5	58	58.00	10	58.00	10	0.052	0.544
c-fat200-5	0.5	148	148.00	10	148.00	10	0.126	0.898
c-fat500-1	0.5	35	35.00	10	35.00	10	0.069	0.548
c-fat500-2	0.5	66	66.00	10	66.00	10	0.135	1.246
c-fat500-5	0.5	164	164.00	10	164.00	10	0.311	2.625
c-fat500-10	0.5	324	324.00	10	324.00	10	0.725	3.960
frb59-26-5	0.95	216	216.60	7	216.80	10	422.615	224.426
frb59-26-4	0.95	222	222.60	9	223.40	10	273.719	149.048
frb59-26-2	0.95	229	229.00	8	229.40	10	372.323	127.098
frb59-26-1	0.95	232	232.10	8	233.00	10	466.144	152.021

Table 2: Numerical results over ten runs of each algorithm for each instance - Part A.

IG^{*} and BRKGA-ExactQClique for one execution of instances frb59-26-2 and frb50-23-5, respectively. In addition, Figures 9 and 10 display how the best solutions found by the two algorithms evolve along the first 1000 seconds of processing time, for the same instances frb59-26-2 and frb50-23-5, respectively. They show that BRKGA-ExactQClique systematically finds better solutions faster than BRKGA-IG^{*}. The best solution value obtained by BRKGA-ExactQClique is better than or equal to that found by

Instances	γ	target	IG*	#opt.	QClique	#opt.	IG*	QClique
		size					(seconds)	(seconds)
frb56-25-5	0.95	192	192.30	10	193.50	10	230.380	76.185
frb56-25-4	0.95	190	189.00	4	190.40	9	502.663	285.087
frb56-25-2	0.95	204	204.50	10	204.80	10	290.945	119.251
frb56-25-1	0.95	220	221.00	10	220.20	10	295.512	150.905
frb53-24-5	0.95	162	162.50	10	162.80	10	225.562	105.855
frb53-24-4	0.95	174	175.00	9	174.50	10	233.042	167.407
frb53-24-2	0.95	168	168.00	8	168.20	10	398.870	356.155
frb53-24-1	0.95	192	192.90	8	193.10	10	287.673	143.790
frb50-23-5	0.95	156	156.30	9	156.00	10	358.773	183.970
frb50-23-4	0.95	150	150.80	10	150.30	10	125.065	45.791
frb50-23-2	0.95	153	152.90	8	153.30	10	355.328	101.622
frb50-23-1	0.95	153	153.80	10	154.20	10	160.822	98.673
frb45-21-5	0.95	118	118.60	10	118.40	10	111.796	53.902
frb45-21-4	0.95	125	125.40	10	125.20	10	47.415	36.153
frb45-21-2	0.95	120	120.30	10	120.10	10	98.871	61.110
frb45-21-1	0.95	119	119.00	10	119.80	10	59.476	18.847
frb40-19-5	0.95	98	98.10	10	98.40	10	56.161	48.600
frb40-19-4	0.95	94	94.10	10	94.30	10	34.232	20.139
frb40-19-2	0.95	101	101.10	10	101.20	10	72.465	56.743
frb40-19-1	0.95	109	109.20	10	109.10	10	31.920	22.767
frb35-17-5	0.95	78	78.00	10	78.30	10	19.124	22.494
frb35-17-4	0.95	79	79.20	10	79.20	10	19.196	43.029
frb35-17-2	0.95	73	73.30	10	73.20	10	7.134	9.398
frb35-17-1	0.95	77	77.20	10	77.00	10	18.130	18.475
frb30-15-5	0.95	59	59.00	10	59.00	10	6.447	5.907
frb30-15-4	0.95	60	60.00	10	60.00	10	44.322	135.412
frb30-15-2	0.95	58	58.00	10	58.00	10	5.778	9.608
frb30-15-1	0.95	59	59.00	10	59.00	10	7.652	5.563
gen200_p0.9_44	0.999	40	40.00	10	40.00	10	0.114	0.617
gen400_p0.9_55	0.999	53	53.00	10	53.00	10	8.169	103.434
gen400_p0.9_65	0.999	58	58.50	10	57.50	6	45.234	379.328

Table 3: Numerical results over ten runs of each algorithm for each instance - Part B.

BRKGA-IG* anytime along the runs displayed in these figures.

Instances	γ	target	IG^*	#opt.	QClique	#opt.	IG^*	QClique
		size					(seconds)	(seconds)
hamming6-2	0.95	37	37.00	10	37.00	10	0.033	0.073
hamming6-4	0.5	32	32.00	10	32.00	10	0.028	0.163
hamming8-2	0.999	129	129.00	10	129.00	10	0.233	0.214
hamming8-4	0.8	71	71.00	10	71.00	10	0.341	0.798
hamming10-2	0.999	525	525.00	10	525.00	10	12.910	7.591
hamming10-4	0.95	82	82.00	10	82.50	10	48.831	11.117
johnson8-4-4	0.8	43	43.00	10	43.00	10	0.056	0.310
johnson16-2-4	0.8	34	34.00	10	34.00	10	0.061	0.195
johnson32-2-4	0.95	21	21.00	10	21.00	10	0.704	0.909
keller4	0.8	54	54.00	10	54.00	10	0.178	0.578
keller5	0.8	486	486.00	10	486.00	10	4.467	15.628
keller6	0.95	271	271.10	8	272.00	9	416.216	388.637
p_hat300-1	0.5	64	64.00	10	64.00	10	2.604	36.551
p_hat300-2	0.8	114	114.00	10	114.00	10	0.927	2.000
p_hat500-1	0.5	96	96.00	10	96.00	10	1.916	6.745
p_hat500-2	0.8	211	211.00	10	211.00	10	1.865	4.451
p_hat700-1	0.5	119	119.00	10	119.00	10	13.095	55.259
p_hat700-2	0.8	288	288.00	10	288.00	10	3.695	8.318
p_hat1000-1	0.5	144	144.00	10	144.10	10	14.248	32.708
p_hat1000-2	0.8	385	385.00	10	385.00	10	9.384	20.052
p_hat1000-3	0.95	210	210.00	10	210.00	10	10.737	15.623
p_hat1500-2	0.8	642	642.00	10	642.00	10	31.317	53.467
san1000	0.8	562	562.00	10	562.00	10	10.101	16.393
san200_0.7_1	0.95	57	57.00	10	57.00	10	0.195	0.444
san200_0.7_2	0.95	34	34.00	10	34.00	10	0.206	0.520
san200_0.9_1	0.999	55	55.70	10	55.50	10	6.079	24.426
san200_0.9_2	0.999	60	60.00	10	60.00	10	0.410	1.323
san200_0.9_3	0.999	42	42.20	10	42.20	10	12.199	49.317
san400_0.5_1	0.6	285	285.00	10	285.00	10	1.236	3.239
san400_0.7_1	0.95	201	201.00	10	201.00	10	1.137	1.885
san400_0.7_2	0.95	62	62.00	10	62.00	10	0.566	0.903
san400_0.7_3	0.95	40	40.00	10	40.00	10	25.685	52.996
sanr400_0.7	0.95	32	32.00	10	32.00	10	2.130	2.486
sanr400_0.5	0.8	32	32.00	10	32.00	10	1.527	1.989

Table 4: Numerical results over ten runs of each algorithm for each instance - Part C.

7. Concluding remarks

We proposed an improvement in the biased random-key genetic algorithm BRKGA-IG^{*} of Pinto et al. (2017) for approximately solving the maximum quasi-clique problem, hybridizing a local search al-



Fig. 3: Time to target plot for instance DSJC500.5.

Fig. 4: Time to target plot for instance frb45-21-1.



© 2017 International Transactions in Operational Research © 2017 International Federation of Operational Research Societies



Fig. 5: Time to target plot for instance frb30-15-2.

Fig. 6: Time to target plot for instance frb30-15-5.



© 2017 International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

Instances	γ	target size	IG*	#opt.	QClique	#opt.	IG* (seconds)	QClique (seconds)
DSJC500.5	34	0.8	34.000	200	34.000	200	22.048	14.376
frb45-21-1	119	0.95	119.325	200	119.525	200	37.418	19.143
frb30-15-2	58	0.95	58.000	200	58.000	200	7.477	11.301
frb30-15-5	59	0.95	59.010	200	59.055	200	6.350	9.046

Table 5: Numerical results over 200 runs of each algorithm for each instance.

Fig. 7: Population evolution for instance frb59-26-2 along 100 generations of BRKGA-IG* and BRKGA-ExactQClique.



gorithm that is based on an exact algorithm. The exact local search is a modification of the QClique algorithm of Ribeiro and Riveaux (2017). The numerical results showed that the new, hybrid approach outperforms BRKGA-IG^{*}, finding better solutions for most test problems.

Future work will focus on improvements of the exact local search method and better parameter settings. In particular, we are interested in investigating better values for the destruction parameter δ and for

Fig. 8: Population evolution for instance frb50-23-5 along 100 generations of BRKGA-IG* and BRKGA-ExactQClique.



the maximum number of nodes *maxnodes* in the search tree. The exact algorithm may also be further improved by cuts that prune duplicate solutions.

References

- Abello, J., Resende, M., Sudarsky, S., 2002. Massive quasi-clique detection. In Abello, J. and Vitter, J. (eds), *Proceedings of the 5th Latin American Symposium on the Theory of Informatics*, Springer, pp. 598–612.
- Aiex, R., Resende, M., Ribeiro, C.C., 2002. Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics* 8, 343–373.
- Aiex, R., Resende, M., Ribeiro, C.C., 2007. TTTPLOTS: A Perl program to create time-to-target plots. *Optimization Letters* 1, 355–366.

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 2, 154–160.

Brandão, J.S., Noronha, T.F., Resende, M.G.C., Ribeiro, C.C., 2015. A biased random-key genetic algorithm for single-round divisible load scheduling. *International Transactions Operational Research* 22, 823–839.

Brandão, J.S., Noronha, T.F., Resende, M.G.C., Ribeiro, C.C., 2017. A biased random-key genetic algorithm for scheduling heterogeneous multi-round systems. *International Transactions Operational Research* 27, 1061–1077.

Fig. 9: Evolution of the best value found by BRKGA-IG^{*} and BRKGA-ExactQClique along the 1000 first seconds of running time for instance frb59-26-2.



Fig. 10: Evolution of the best value found by BRKGA-IG^{*} and BRKGA-ExactQClique along the 1000 first seconds of running time for instance frb50-23-5.



© 2017 International Transactions in Operational Research © 2017 International Federation of Operational Research Societies

- Brunato, M., Hoos, H., Battiti, R., 2008. On effectively finding maximal quasi-cliques in graphs. In Maniezzo, V., Battiti, R. and Watson, J.P. (eds), *Learning and Intelligent Optimization, Lecture Notes in Computer Science*. Vol. 5313. Springer, Berlin, pp. 41–55.
- FICO, 2017. FICO Xpress Optimization Suite 7.6. http://www.fico.com/en/products/fico-xpress-optimization-suite. Online reference, last visited on May 25, 2017.
- Goncalves, J.F., Resende, M.G.C., 2011. Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics 17, 487–525.
- Goncalves, J.F., Resende, M.G.C., Toso, R.F., 2013. Biased and unbiased random key genetic algorithms: An experimental analysis. In Abstracts of the 10th Metaheuristics International Conference, Singapore.
- Hoos, H., Stützle, T., 1998. Evaluation of Las Vegas algorithms Pitfalls and remedies. In Cooper, G. and Moral, S. (eds), *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, pp. 238–245.
- Johnson, D.S., 1996. Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Dimacs Series in Discrete Mathematics and Theoretical Computer Science. Vol. 26. American Mathematical Society, Providence.
- Karp, R.M., 1972. Reducibility among combinatorial problems. In Miller, R.E. and Thatcher, J.W. (eds), Complexity of Computer Computations, Plenum, New York, pp. 85–103.
- López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M., 2011. The IRACE package: Iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.
- Noronha, T.F., Resende, M.G.C., Ribeiro, C.C., 2011. A biased random-key genetic algorithm for routing and wavelength assignment. *Journal of Global Optimization* 50, 503–518.
- Oliveira, A.B., 2013. Heurísticas para o Problema de Quasi-Clique de Cardinalidade Máxima. Master's thesis, Universidade Federal Fluminense, Niterói, Brazil.
- Oliveira, A.B., Plastino, A., Ribeiro, C.C., 2013. Construction heuristics for the maximum cardinality quasi-clique problem. In Abstracts of the 10th Metaheuristics International Conference, Singapore, p. 84.
- Pajouh, F.M., Miao, Z., Balasundaram, B., 2014. A branch-and-bound approach for maximum quasi-cliques. Annals of Operations Research 216, 1, 145–161.
- Pattillo, J., Veremyev, A., Butenko, S., Boginski, V., 2013. On the maximum quasi-clique problem. Discrete Applied Mathematics 161, 244–257.
- Pérez Cáceres, L., López-Ibáñez, M., Stützle, T., 2014. An analysis of parameters of IRACE. In Proceedings of the 14th European Conference on Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science. Vol. 8600. Springer, Berlin, pp. 37–48.
- Pinto, B.Q., Ribeiro, C.C., Riveaux, J.A., Rosseti, I., 2017. A biased random-key genetic algorithm for solving the maximum quasi-clique problem. *Journal of Global Optimization* Submitted.
- Ribeiro, C.C., Riveaux, J.A., 2017. An exact algorithm for the maximum quasi-clique problem. *Journal of Global Optimization* Submitted.
- Spears, W., de Jong, K., 1991. On the virtues of parameterized uniform crossover. In Belew, R. and Booker, L. (eds), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman, San Mateo, pp. 230–236.
- Toso, R.F., Resende, M.G.C., 2015. A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software* 30, 81–93.
- Veremyev, A., Prokopyev, O.A., Butenko, S., Pasiliao, E.L., 2016. Exact MIP-based approaches for finding maximum quasiclique and dense subgraph. *Computational Optimization and Applications* 64, 1, 177–214.