Annals of Operations Research
https://doi.org/10.1007/s10479-023-05609-7

ORIGINAL RESEARCH

A biased random-key genetic algorithm for the minimum quasi-clique partitioning problem

Rafael A. Melo[1] · Celso C. Ribeiro[2] · Jose A. Riveaux[3]

Received: 23 May 2023 / Accepted: 8 September 2023

Abstract
Let $G = (V, E)$ be a graph with vertex set V and edge set E, and consider $\gamma \in [0, 1)$ to be a real constant. A γ-clique (or quasi-clique) is a subset $V' \subseteq V$ inducing a subgraph of G with edge density at least γ. In this paper, we tackle the minimum quasi-clique partitioning problem (MQCPP), which consists of obtaining a minimum-cardinality partition of V into quasi-cliques. We propose a biased random-key genetic algorithm (BRKGA) relying on an efficient partitioning decoder that allows merge operations to combine smaller quasi-cliques into larger ones. Furthermore, we show that MQCPP and the problem of covering the graph with a minimum number of quasi-cliques are not equivalent. Computational experiments indicate that the proposed BRKGA is very effective in obtaining high-quality solutions for MQCPP in low computational times. More specifically, it can at least match all the best solutions available in the literature, strictly improving over them for 20.3% of the benchmark instances. Besides, the approach is robust as it obtains small deviations from the best-achieved solutions when executing multiple independent runs. We also consider the performance of our BRKGA on a new set of challenging large instances with up to 2851 vertices.

Keywords Combinatorial optimization · Quasi-clique partitioning · Quasi-cliques · Biased random-key genetic algorithms · Network clustering · Metaheuristics

✉ Rafael A. Melo
rafael.melo@ufba.br

Celso C. Ribeiro
celso@ic.uff.br

Jose A. Riveaux
jangel.riveaux@usp.br

[1] Institute of Computing, Universidade Federal da Bahia, Salvador, BA 40170-115, Brazil

[2] Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil

[3] Department of Production Engineering, University of São Paulo, São Paulo, SP 05508-010, Brazil

Published online: 28 September 2023

1 Introduction

1.1 Basic definitions

Let $G = (V, E)$ be a simple undirected graph with a set $V = \{v_1, \ldots, v_n\}$ of vertices and a set E of edges. A graph G is complete if every pair of vertices in V is connected by an edge in E. Given $H \subseteq V$, denote by $G[H]$ the graph induced in G by H, i.e., that with vertex set H and edge set formed by every edge in E with both extremities in H. A clique is a subset $V' \subseteq V$ that induces a complete subgraph $G[V']$ of G. Let $N(v)$ represent the neighbors of $v \in V$ in G and $deg_G(v)$ its degree, determined by $|N(v)|$. Moreover, consider $deg_G(v, H) = |N(v) \cap H|$. A graph partition is a partition of its vertex set. A clique partition (or vertex clique cover) is a partition of V into cliques. The minimum clique partitioning problem (or minimum vertex clique covering problem) consists in obtaining a minimum cardinality clique partition.

The density of G is defined as $d(G) = |E|/(|V| \cdot (|V| - 1)/2)$. Given G and a threshold $\gamma \in (0, 1]$, a γ-clique (also called a γ-quasi-clique) is a subset $C \subseteq V$ such that the density of $G[C]$ is greater than or equal to γ. Given a specific value for γ, a γ-clique is also denoted by simply quasi-clique. The minimum quasi-clique partitioning problem (MQCPP) consists in obtaining a minimum cardinality partition of a graph into quasi-cliques. Figures 1 and 2 illustrate, respectively, an input graph and a minimum quasi-clique partition for $\gamma = 0.51$. A vertex quasi-clique cover is a set of quasi-cliques that cover all the vertices of a graph. The minimum vertex quasi-clique covering problem (MVQCCP) consists in obtaining a minimum cardinality quasi-clique cover.

Additionally, a γ-clique C is maximal if there is no other γ-clique C' such that $C \subset C'$. Let $\bar{C} = V \backslash C$ denote the complement of C. A vertex $v \in \bar{C}$ is a γ-vertex with respect to a γ-clique C if $C \cup \{v\}$ is a γ-clique. Denote by $N_\gamma(C)$ the set of γ-vertices with respect to a γ-clique C. A subset $H \subseteq \bar{C}$ is a γ-set if $C \cup H$ is a γ-clique.

1.2 Literature review

MQCPP is related to the problems of obtaining dense subgraphs (Bomze et al., 1999; Wu & Hao, 2015) and dense clusters in networks (Kriegel et al., 2011; Campello et al., 2020). Finding dense subgraphs has shown to be applicable in several domains, including telecommunications (Abello et al., 2002), biology (Spirin & Mirny, 2003), and social networks (Seo & Kim, 2021). On the other hand, applications of partitioning graphs into dense subgraphs appear in bioinformatics (Hu et al., 2005), quantum computing (Verteletskyi et al., 2020), data mining (Glaria et al., 2021), and community detection (Yang et al., 2016; Zhao et al., 2021), among others. MQCPP is also associated with problems of obtaining induced subgraphs with some required properties (Agra et al., 2017; Melo et al., 2021; Marzo et al., 2022; Melo & Ribeiro, 2022, 2023).

Several approaches have been proposed for obtaining maximum and maximal quasi-cliques. Adaptive construction heuristics and their iterated greedy extensions appeared in Oliveira et al. (2013) and Pinto et al. (2018). Metaheuristics include greedy randomized adaptive search procedures (GRASP) (Abello et al., 2002), local search (Tsourakakis et al., 2013), biased random-key genetic algorithms (BRKGA) (Pinto et al., 2018, 2021), memetic algorithm (Zhou et al., 2020), artificial bee colony (Peng et al., 2021), and kernel-based heuristic (Sanei-Mehri et al., 2021). Amongst the exact methods, we can highlight linear and

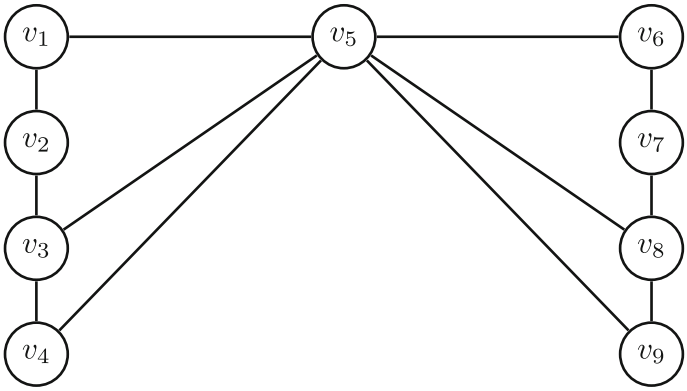

Fig. 1 Example of an input graph G with $|V| = 9$ and $|E| = 12$

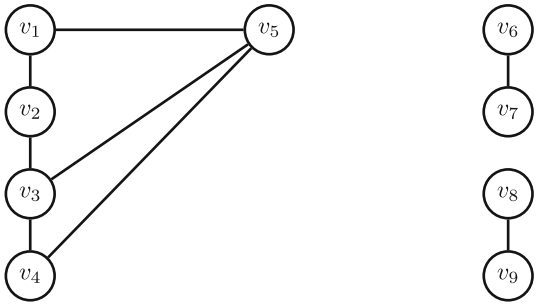

Fig. 2 Optimal quasi-clique partition for the graph in Fig. 1 with $\gamma = 0.51$: the partition $\{\{v_1, v_2, v_3, v_4, v_5\}, \{v_6, v_7\}, \{v_8, v_9\}\}$ is composed of three quasi-cliques with densities 0.6, 1.0, and 1.0, respectively

integer programming approaches (Pattillo et al., 2013; Veremyev et al., 2016; Marinelli et al., 2021) and backtracking (Ribeiro & Riveaux, 2019).

Despite its vast potential applicability, quasi-clique partitioning was only recently considered in the literature. To the best of our knowledge, the first work on the subject was Basu et al. (2014) in the context of community detection. However, the authors did not consider the problem from an optimization point of view. Instead, they proposed a game-theoretical approach for partitioning a graph into so-called (λ, γ)-cliques, characterized by both their densities and their vertices' degrees. Melo et al. (2022) formalized MQCPP and showed that its decision version is NP-complete even for the case of bipartition (differently from what happens with the minimum clique partitioning problem). They also proposed four compact integer programming formulations and a multi-start greedy randomized heuristic for the problem. MQCPP was recently considered in a tutorial about MIP formulations for optimization problems involving induced graphs (Melo & Ribeiro, 2023).

1.3 Contributions and organization

The contributions of our work are twofold. First, we show that although the minimum clique partitioning and the minimum vertex cover partitioning problems are equivalent, MQCPP and MVQCCP are not. After that, as our main contribution, we provide a BRKGA metaheuristic for MQCPP. The BRKGA algorithm relies on an effective partitioning decoder that employs merging operations allowing the union of smaller quasi-cliques into larger ones. We also propose a new benchmark set composed of challenging large instances.

Table 1 Number of edges in a complete graph with $|V|$ vertices and the required minimum number of edges for a γ-clique with $\gamma = 0.51$

$\lvert V\rvert$	$\frac{\lvert V\rvert\cdot(\lvert V\rvert-1)}{2}$	$\left\lceil 0.51 \cdot \frac{\lvert V\rvert\cdot(\lvert V\rvert-1)}{2} \right\rceil$
2	1	1
3	3	2
4	6	4
5	10	6
6	15	8
7	21	11
8	28	15
9	36	19

The remainder of this paper is organized as follows. Section 2 proves that MQCPP and MVQCCP are not equivalent. Section 3 details the proposed biased random-key genetic algorithm for MQCPP. Section 4 summarizes the computational results. Section 5 discusses concluding remarks.

2 Nonequivalence of MQCPP and MVQCCP

As it was already mentioned in Sect. 1.1, finding a minimum clique partition in a graph is equivalent to obtaining a minimum vertex clique cover (Garey & Johnson, 1979). Theorem 1 shows via an example that, differently from what happens with the minimum clique partitioning and the minimum vertex clique covering problems, MQCPP and MVQCCP are not equivalent.

Theorem 1 *MQCPP and MVQCCP are not equivalent.*

Proof Let G be the graph illustrated in Fig. 1 and consider $\gamma = 0.51$. The required minimum number of edges for γ-cliques with $|V| \in \{2, 3, 4, 5, 6, 7, 8, 9\}$ and $\gamma = 0.51$ are provided in Table 1. G has a minimum quasi-clique partition with three quasi-cliques (see Fig. 2) and a minimum quasi-clique cover with two quasi-cliques (see Fig. 3). To see why these are minimum, firstly, notice that the graph only has γ-cliques ($\gamma = 0.51$) for subgraphs with up to five vertices (see Table 1). This ensures a lower bound of two quasi-cliques for both MQCPP and MVQCCP. Thus, the quasi-clique cover in Fig. 3 is minimum. Secondly, observe that all the possible quasi-cliques with four or five vertices must contain v_5. This implies that, in a minimum cardinality quasi-clique partition, v_5 has to be in a quasi-clique with five vertices. Consequently, at least two disjoint quasi-cliques are needed for partitioning the remaining four vertices. This ensures a lower bound of three quasi-cliques for MQCPP. Thus, the quasi-clique partition in Fig. 2 is minimum. □

3 Biased random-key genetic algorithm

Random-key genetic algorithms (RKGA) were introduced by Bean (1994). Solutions are associated with vectors of real numbers (denoted as random keys) in the interval $[0, 1)$. A deterministic algorithm, which in this context is also called a decoder, takes a vector of random keys to produce a feasible solution to the optimization problem at hand and computes

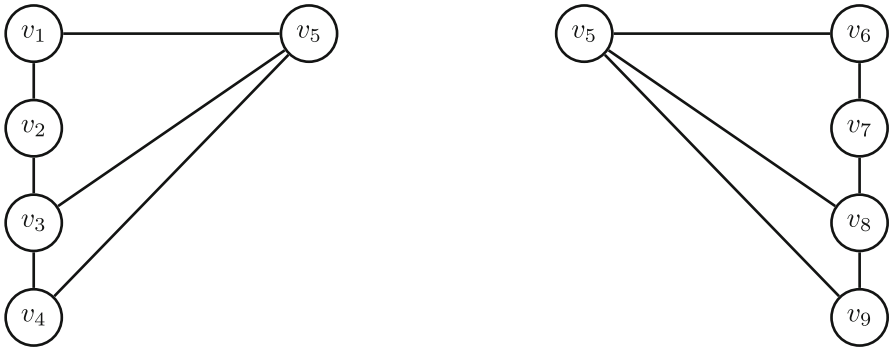

Fig. 3 Optimal vertex quasi-clique cover for the graph in Fig. 1 with $\gamma = 0.51$: the cover $\{\{v_1, v_2, v_3, v_4, v_5\}, \{v_5, v_6, v_7, v_8, v_9\}\}$ is composed of two quasi-cliques with density 0.6 each

its fitness or objective value. Parents are randomly selected from the entire population for mating and crossover, with repetitions allowed.

A biased random-key genetic algorithm (BRKGA) (Gonçalves & Resende, 2011) differs from an RKGA by the strategy used to select parents for mating, see Resende and Ribeiro (2016) for an advanced tutorial of methods and applications. One of the main characteristics of a BRKGA is that each new solution is generated by the combination of one solution selected at random from the subset of elite solutions of the current population with another that is always a non-elite solution. The crossover strategy is biased not only because one parent is always an elite solution but also because it has a higher probability of passing its characteristics to the offspring.

The algorithm uses the parametric uniform crossover strategy originally proposed in Spears and De Jong (1991) for combining two parent solutions and producing a new one. The solution generated by crossover inherits with a higher probability each of its keys from the best parent. The algorithm does not use the standard mutation operator. Instead, the following concept of mutants is used: new solutions (i.e., mutants) are introduced in the population at each generation, randomly generated with the same strategy as in the initial population. They play the same role as the mutation operator in more standard genetic algorithm frameworks, that is, diversifying the search and assisting the procedure in escaping from local optima (Gonçalves & Resende, 2015; Brandão et al., 2015, 2016; Pinto et al., 2020; Andrade et al., 2021; Carrabs, 2021).

A BRKGA evolves a population formed by vectors of real numbers. Its evolutionary dynamics can be summarized as follows. The initial population P is entirely formed by vectors formed by randomly generated elements in the interval [0, 1). At each generation, the current population is partitioned into two subsets: TOP and $REST$. The subset TOP always contains the best (or elite) solutions, while $REST$ contains the non-elite solutions. The population size is thus $|P| = |TOP| + |REST|$. The non-elite set $REST$ is further partitioned into two subsets, MID and BOT, with BOT containing the worst elements of the population. The algorithm is elitist: the solutions in TOP are copied from the population of one generation to the next, i.e., TOP remains the same. This is shown in Fig. 4. The solutions in MID are replaced by new solutions generated by a biased crossover operation between an elite solution from TOP and a non-elite solution from $REST$. The solutions in BOT are replaced by randomly generated mutant solutions.

The implementation of biased random-key genetic algorithms can be supported by the C++ library developed by Toso and Resende (2015). The instantiation of the framework

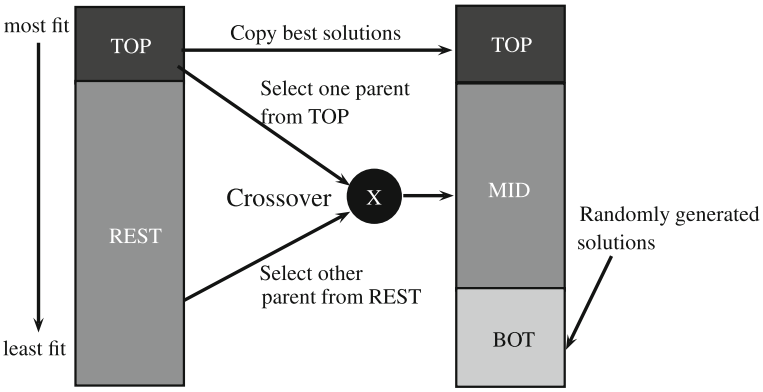

Fig. 4 Population evolution between consecutive generations of a BRKGA

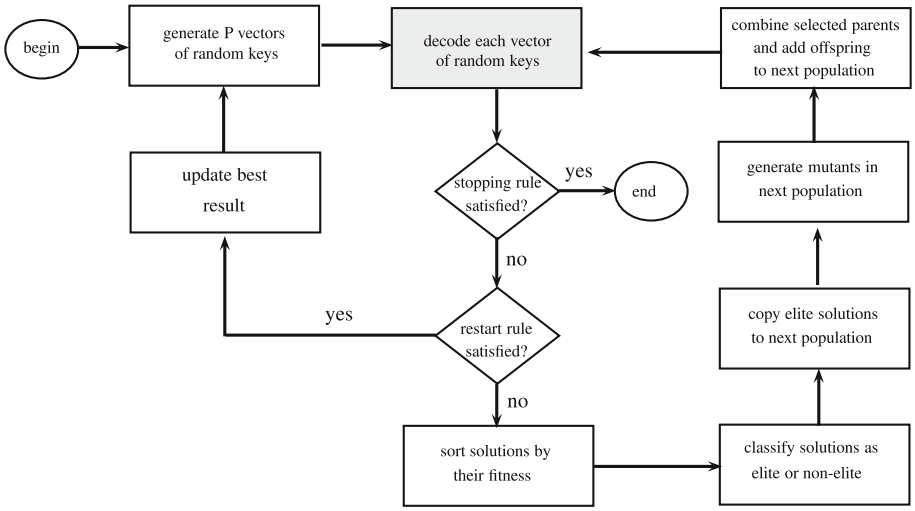

Fig. 5 BRKGA framework

shown in Fig. 5 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. Other applications of this framework in the implementation of BRKGAs appeared, e.g., in Noronha et al. (2011), Brandão et al. (2016), Pinto et al. (2020).

3.1 Solution encoding

Define a solution to MQCPP as a partition $S = \{C_1, \ldots, C_k\}$ of V into quasi-cliques. Moreover, consider a partial solution S' as a partition of $V' \subset V$ into quasi-cliques. Besides, let $V(S')$ be the set of vertices covered in S' and $\mathcal{C}'(v)$ be the set of quasi-cliques in S' for which $v \in \bar{V}(S') = V \backslash V(S')$ is a γ-vertex. Additionally, let $\bar{G}' = G[\bar{V}(S')]$.

In our approach, we encode a solution as a $|V|$-dimensional vector R of random keys $R_v \in [0, 1), v = 1, \ldots, |V|$. Each element R_v indicates the priority of inserting a vertex $v \in V$ into the solution, i.e., adding it to a quasi-clique.

3.2 Partitioning decoder

We propose a decoder that inserts each vertex $v \in \bar{V}(S')$ into the partial solution under construction S' (initially empty) in the order implied by their increasing random keys R_v, $v = 1, \ldots, |V|$. Each of them is added to the lowest-index (i.e., the first created) quasi-clique in $\mathcal{C}'(v)$. The partitioning decoder is detailed in Algorithm 1. It takes as inputs a graph G, a density γ, and a solution encoded by the random-key vector R. Firstly, the solution is defined as empty (line 1). The loop of lines 2–21 is executed while there are vertices from V that are not in the solution. Line 3 selects the vertex v with the lowest key R_v among those that are not part of the solution. If v is not a γ-vertex for any of the existing quasi-cliques, a new single-element quasi-clique formed exclusively by vertex v is built (lines 4–5). Otherwise, the loop of lines 7–13 verifies whether inserting v into the solution allows the merge of two quasi-cliques into a larger one. All pairs of quasi-cliques $\{C', C''\}$ belonging to $\mathcal{C}'(v)$ are considered as candidates, in the order they were created (line 8). If $\{v\}$ can be merged together with C' and C'' into a single, larger quasi-clique (line 9), then they are merged (lines 10–11), C is set to the merged quasi-clique (line 12) and the for loop is halted (line 13). If C is nonempty (line 14), i.e., smaller quasi-cliques were merged into a larger one, the algorithm verifies whether C can be further merged with other quasi-cliques that are γ-sets to it (lines 15–18). In case quasi-cliques could not be merged with the addition of v, then the vertex is inserted into the lowest-index, i.e., the first created quasi-clique in $\mathcal{C}'(v)$ (lines 19–21). The algorithm returns in line 22 the obtained solution S' and its associated fitness value $|S'|$.

Remark 1 The decoder proposed in Algorithm 1 can work properly with or without the merging operations (lines 8–18).

Proposition 1 *Algorithm 1 can be implemented to run in time* $O(|V|^3)$. *If the merging operations are disabled, the running time drops to* $O(|V| \log |V| + |E|)$.

Proof To analyze the running time of Algorithm 1, assume G is represented by an adjacency list. Let each quasi-clique be characterized by a list of vertices and a solution (partial or complete) by a vector of lists. Additionally, consider the following auxiliary variables and structures: the number of vertices already in the solution, a $|V|$-dimensional vector in which each element indicates the quasi-clique vertex $v \in V$ belongs, and a $|V|$-dimensional vector in which each element specifies the degree of $v \in V$ in $\bar{G}'$. Besides, consider the following variables and structures for each quasi-clique in the solution: cardinality, number of edges, number of vertices adjacent to each vertex in $\bar{V}(S')$, and number of vertices adjacent to the vertices of every other quasi-clique.

The first analysis regards the computational costs associated with adding a vertex $v \in \bar{V}(S')$ to the solution, i.e., inserting it into a quasi-clique. Notice that when adding a new vertex to a quasi-clique, we update the vector indicating its quasi-clique and the list of vertices in the corresponding quasi-clique together with its cardinality, all of which take $O(1)$. Then, by going through the list of vertices adjacent to v, we can update in time $O(|V|)$ the degree of the changed quasi-clique for every vertex in $\bar{V}(S')$, its number of edges, and the sets $C'(u)$ for the appropriate vertices $u \in \bar{V}(S')$. Notice, however, that all these updates throughout the algorithm's execution correspond to going once through the graph's adjacency list while performing constant-time operations. Thus, the total running time for performing all of them is $O(|V| + |E|)$.

Line 1 takes time $O(1)$. The while loop of lines 2–21 is executed $O(|V|)$ times. In each iteration, the selection of a vertex with a minimum key in line 3 can be performed in $O(1)$

Algorithm 1: Partitioning-Decoder(G, γ, R)

```
1  S' ← ∅;
2  while V̄(S') ≠ ∅ do
3      v ← arg min{R_j : j ∈ V̄(S')};
4      if C'(v) = ∅ then
5          S' ← S' ∪ {{v}};
6      else
7          C ← ∅;
8          for every pair C', C'' ∈ C'(v) taken in the order they were created do
9              if d(G[C' ∪ C'' ∪ {v}]) ≥ γ then
10                 C' ← C' ∪ C'' ∪ {v};
11                 S' ← S' \ C'';
12                 C ← C';
13                 break;
14         if C ≠ ∅ then
15             for every C' ∈ S' \ {C} taken in the order they were created do
16                 if d(G[C ∪ C']) ≥ γ then
17                     C ← C ∪ C';
18                     S' ← S' \ C';
19         else
20             C ← the lowest-index quasi-clique in C'(v);
21             C ← C ∪ {v};
22 return S', |S'|;
```

assuming the vertices are already sorted, what can be done only once at the beginning of the algorithm's execution in $O(|V|\log|V|)$. The cost of line 5 is already considered in the costs of inserting the vertices into the quasi-cliques. The condition in line 9 can be verified in $O(1)$. The updates related to lines 10–11 for the new larger quasi-clique: cardinality, number of vertices adjacent to each vertex in $\bar{V}(S')$, number of edges, and updated number of vertices adjacent to the vertices of every other quasi-clique can all be done in $O(|V|)$. To update the vector indicating the quasi-cliques of the different vertices, one has to go through the list of vertices in the largest index clique and change the values accordingly, which can be done in $O(|V|)$. Thus, the cost of lines 8–13 is $O(|\mathcal{C}'(v)|^2 + |V|)$. The for loop of lines 15–18 is executed $O(|S'|)$ times. In each iteration, evaluating the condition in line 16 takes $O(1)$ and the updates in lines 17–18 take $O(|V|)$. Thus, the whole loop takes $O(|S'||V|)$. Line 20 can be executed in $O(1)$. The cost of line 21 was already accounted for in the costs of inserting vertices into quasi-cliques. Let S'_{max} be the maximum number of quasi-cliques in S' during the execution of the algorithm. Thus, Algorithm 1 can be implemented to run in $O(|V|\log|V|+|E|+|V|^2 S'_{max})$, what, in terms of the input values, leads to $O(|V|^3)$. Notice that without the merging operations (lines 17–18) the algorithm can be implemented to run in $O(|V|\log|V|+|E|)$. □

Remark 2 Although Algorithm 1 has a worst-case running time in $O(|V|^3)$, it is sensitive to the number of quasi-cliques in the generated partial solutions. Therefore, practical settings can imply lower-order running times depending on the graph density and the value of γ.

4 Computational experiments

In this section, we summarize the experiments performed to assess the effectiveness of the proposed approach. All experiments were conducted on a machine running under Ubuntu GNU/Linux, with an Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz processor and 16Gb of RAM. The algorithms were implemented in C++ with the BRKGA implementations using the API of Toso and Resende (2015). All the tests were executed using a single thread.

Section 4.1 describes the benchmark instances used in the literature. Section 4.2 enumerates the tested approaches. Section 4.3 summarizes the computational experiments using the instances available in the literature. Section 4.4 proposes a new set of large benchmark instances and outlines the results for these new instances. Section 4.5 provides a comparative summary of the performance of the two BRKGA variants.

4.1 Benchmark instances

Each of the original benchmark instances (Melo et al., 2022) corresponds to an input graph and a density value γ. The used input graphs correspond to real networks that are commonly used in the literature and were recently considered in Matsypura et al. (2019), graphs from the DIMACS Implementation Challenges (DIMACS, 2021), and networks from the Moviegalaxies data set (Kaminski et al., 2018). Table 2 displays the input graphs, the numbers of vertices and edges, and the densities, ordered by $|V|$. For each of these 23 graphs, instances are considered for $\gamma \in \{0.999, 0.950, 0.900, 0.800, 0.700, 0.600, 0.500, 0.400, 0.300\}$ (207 instances in total).

4.2 Tested approaches and parameter settings

We considered the following approaches in our experiments: the multi-start heuristic (MSH) proposed in Melo et al. (2022), the BRKGA without the merging operations (BRKGA), and the complete BRKGA (BRKGA$_m$). We also report the best results obtained in Melo et al. (2022) using the IP formulations proposed therein (executed with a time limit of 3600 s).

The parameters used for MSH were set as in Melo et al. (2022). The settings for BRKGA and BRKGA$_m$ were established based on preliminary experiments considering the recommendations described in Gonçalves and Resende (2011). The settings are: the number of chromosomes in the population is $p = |P| = 100$, the size of the elite set in the population is $p_e = 0.2p$, the number of mutants to be introduced in the population at each generation is $p_m = 0.1p$, and the probability that a key is inherited from the elite parent is $\rho_e = 0.7$. Besides, BRKGA restarts with a newly generated population whenever 100 generations are performed without improving the incumbent solution.

4.3 Computational results

In the experiments reported in this section, ten independent runs were performed using MSH, BRKGA, and BRKGA$_m$ for each benchmark instance. A time limit of 300 s (five minutes) was imposed on each run of each algorithm.

Table 3 summarizes the results assembled by the input graphs. All the reported values are averaged over the nine instances (with different values of γ) for the specified input graph. The results for MSH, BRKGA, and BRKGA$_m$ correspond to the ten independent executions.

Table 2 Benchmark graphs used in Melo et al. (2022)

Input graph	$\|V\|$	$\|E\|$	$d(G)$
Memento	14	19	0.2088
The_X_Files	24	41	0.1486
Alien_3	25	77	0.2567
high-tech	33	91	0.1723
karate	34	78	0.1390
mexican	35	117	0.1966
sawmill	36	62	0.0984
tailorS1	39	158	0.2132
chesapeake	39	170	0.2294
Batman_Returns	51	124	0.0973
attiro	59	128	0.0748
krebs	62	153	0.0809
dolphins	62	159	0.0841
prison	67	142	0.0642
sanjuansur	75	144	0.0519
jean	77	254	0.0868
3-FullIns_3	80	346	0.1095
david	87	406	0.1085
myciel6	95	755	0.1691
4-FullIns_3	114	541	0.0840
ieeebus	118	179	0.0259
sfi	118	200	0.0290
anna	138	493	0.0522

Table 3 Summary of the average results for the original instances, assembled by the input graphs

Input graph	IP	MSH				BRKGA				BRKGA$_m$			
	Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
Memento	**7.78**	**7.78**	7.78	7.78	0.0	**7.78**	7.78	7.78	0.0	**7.78**	7.78	7.78	0.0
The_X_Files	**9.78**	**9.78**	9.81	9.89	14.6	**9.78**	9.78	9.78	0.0	**9.78**	9.78	9.78	0.1
Alien_3	**6.56**	6.89	6.89	6.89	0.3	**6.56**	6.56	6.56	0.0	**6.56**	6.56	6.56	0.0
high-tech	**10.44**	**10.44**	10.44	10.44	20.4	**10.44**	10.44	10.44	0.1	**10.44**	10.44	10.44	0.1
karate	**12.89**	13.22	13.29	13.44	25.8	**12.89**	12.89	12.89	0.6	**12.89**	12.89	12.89	2.7
mexican	**8.00**	**8.00**	8.04	8.22	18.6	**8.00**	8.00	8.00	0.1	**8.00**	8.00	8.00	0.1
sawmill	**12.78**	**12.78**	12.78	12.78	12.6	**12.78**	12.78	12.78	0.1	**12.78**	12.78	12.78	0.3
tailorS1	**9.78**	9.89	10.13	10.22	34.2	**9.78**	9.78	9.78	0.1	**9.78**	9.78	9.78	0.1
chesapeake	10.22	10.33	10.67	10.78	32.4	**10.11**	10.11	10.11	2.1	**10.11**	10.11	10.11	2.3
Batman_Returns	**13.33**	13.67	13.99	14.11	28.7	**13.33**	13.33	13.33	0.5	**13.33**	13.33	13.33	0.9
attiro	18.56	18.78	18.91	19.00	36.5	**18.33**	18.33	18.33	13.9	**18.33**	18.33	18.33	20.2
krebs	21.22	22.33	22.63	22.89	44.5	**21.11**	21.21	21.22	2.9	**21.11**	21.21	21.22	1.5
dolphins	19.11	19.89	20.11	20.44	54.2	**18.89**	18.96	19.00	13.7	**18.89**	18.97	19.00	9.7
prison	18.56	18.78	18.96	19.11	44.1	**18.33**	18.38	18.44	13.2	**18.33**	18.39	18.44	11.5

Table 3 continued

Input graph	IP	MSH				BRKGA				BRKGA$_m$			
	Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
sanjuansur	24.22	24.56	24.83	25.00	47.3	**24.00**	24.06	24.11	12.0	**24.00**	24.09	24.11	9.0
jean	22.89	24.67	25.37	26.00	56.2	**22.67**	22.67	22.67	6.1	**22.67**	22.67	22.67	3.7
3-FullIns_3	25.33	26.33	26.51	26.78	59.3	**25.00**	25.10	25.33	49.1	**25.00**	25.13	25.33	52.5
david	22.78	25.78	26.40	27.00	81.2	**21.89**	22.09	22.33	31.0	**21.89**	22.09	22.22	33.3
myciel6	33.67	38.56	39.03	39.78	97.4	**32.11**	32.31	32.56	26.6	32.33	32.52	32.67	14.6
4-FullIns_3	37.78	38.44	39.02	39.56	69.9	**36.44**	36.79	37.00	54.1	**36.44**	36.83	37.00	53.1
ieeebus	44.33	45.89	46.47	47.11	109.7	**43.00**	43.07	43.22	32.2	**43.00**	43.12	43.22	35.2
sfi	47.78	51.44	52.19	52.78	57.7	**47.56**	47.78	47.89	25.8	47.67	47.83	48.00	37.4
anna	61.56	65.44	66.58	67.44	88.2	**56.33**	56.72	57.22	70.5	56.67	56.94	57.33	69.3

The first column displays the input graph. The second column (best) presents the best results obtained in Melo et al. (2022) using the IP formulations proposed therein (with a time limit of 3600 s). The following 12 columns provide, for MSH, BRKGA, and BRKGA$_m$, the minimum (min), average (avg), and maximum (max) objective values over the ten independent runs, as well as the average time in seconds to reach the best-obtained solution (ttb). The values in boldface indicate the cases in which the corresponding approach achieved the lowest average minimum in each line. The results show that both BRKGA and BRKGA$_m$ obtain high-quality solutions within low computational times. Overall, BRKGA performs the best as it reaches the lowest minimum values for all the input graphs. Besides, BRKGA and BRKGA$_m$ have shown to be robust for the benchmark set as they obtain reasonably low deviations between the minimum and maximum values. Note that the minimum and maximum values are the same for both BRKGA and BRKGA$_m$ for all the graphs with up to 59 vertices. The minimum values obtained by BRKGA and BRKGA$_m$ at least match the best values obtained by the integer programming formulations (within the time limit) for all the input graphs. Moreover, strictly improved results were reached for 14 out of the 23 graphs (60.8%) for which the integer programming approach did not find the optimal solution for all the values of γ within 3600 s. Detailed results are available in Appendix A.

Table 4 summarizes the results gathered by the values of γ. All the reported values are averaged over the 23 instances with different input graphs for each specific value of γ. The results show that BRKGA obtained the lowest minima for all the values of γ but 0.800. On the other hand, BRKGA$_m$ reaches the smallest average minima for all the values of γ larger than or equal to 0.700. We also notice that the average minima obtained by BRKGA improve or match those obtained by BRKGA$_m$ for all the values of γ except 0.800.

Figure 6 graphically summarizes the quality of the best solutions obtained by BRKGA and BRKGA$_m$. It shows the fraction in percent of the 207 instances for which the two approaches obtained: (a) the current best-known solution, (b) a solution that at least matches the best one found by the IP formulations in Melo et al. (2022), and (c) a solution that strictly improves over the best one achieved by the IP formulations in Melo et al. (2022). It shows that BRKGA performs slightly better than BRKGA$_m$, finding the best-known solutions up-to-date for 99.5% of the instances. Besides, the plots show that BRKGA and BRKGA$_m$ improved over the best solutions achieved by the IP formulations in Melo et al. (2022) for 20.3% of the instances, even considering that the formulations were run with a much larger time limit of 3600 s, compared with the 300 s given to the two BRKGA variants.

Table 4 Summary of the average results for the original instances, gathered by the values of γ

γ	IP	MSH				BRKGA				BRKGA$_m$			
	Best	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
0.999	**31.22**	31.57	31.66	31.74	17.4	**31.22**	31.23	31.26	5.6	**31.22**	31.23	31.26	5.5
0.950	**31.09**	31.48	31.58	31.70	21.6	**31.09**	31.10	31.13	5.6	**31.09**	31.10	31.13	5.5
0.900	30.35	30.78	30.92	31.13	25.3	**30.26**	30.29	30.35	10.8	**30.26**	30.29	30.35	11.3
0.800	28.17	28.65	28.77	28.91	39.1	27.96	27.96	27.96	2.6	**27.91**	27.95	27.96	4.5
0.700	26.52	26.91	27.16	27.48	51.8	**25.83**	25.87	25.91	10.1	**25.83**	25.85	25.87	12.4
0.600	17.91	19.00	19.44	19.83	61.8	**16.74**	16.82	16.87	17.7	16.78	16.84	16.87	14.3
0.500	13.65	15.39	16.01	16.39	57.3	**12.26**	12.45	12.61	25.3	12.39	12.52	12.65	28.1
0.400	10.04	12.26	12.86	13.43	75.1	**9.04**	9.16	9.39	30.8	9.09	9.27	9.43	31.3
0.300	6.43	8.87	9.33	9.70	55.2	**6.22**	6.43	6.57	30.4	6.30	6.52	6.61	26.9

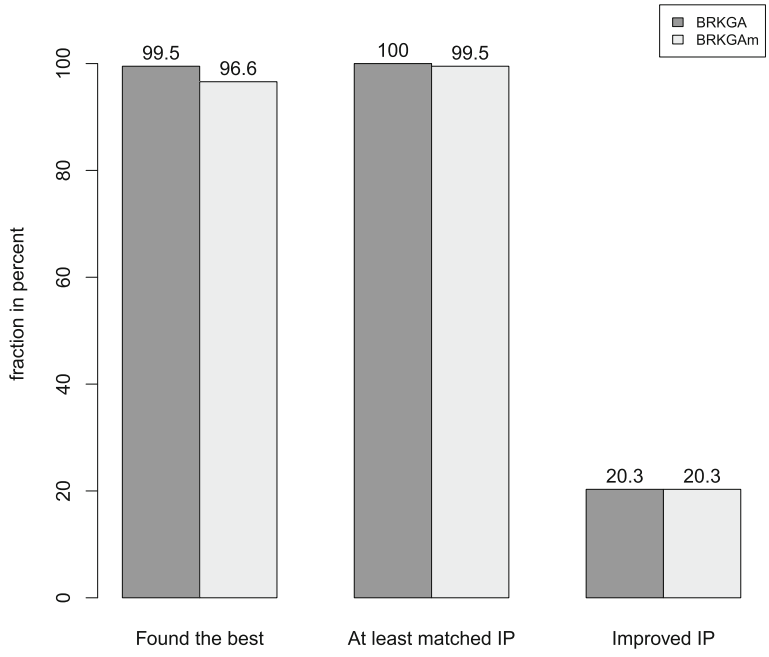

Fig. 6 Fraction in percent of the 207 original benchmark instances for which BRKGA and BRKGA$_m$ obtained: **a** the best-known solution, **b** a solution that at least matches the best found by the IP formulations in Melo et al. (2022), and **c** a solution that strictly improves over the best one achieved by the IP formulations in Melo et al. (2022)

The boxplot with jittered points depicted in Fig. 7 summarizes the deviations in percent from the best-known solutions for all the 2070 runs (ten runs for each of the 207 instances). The deviation for a given run is defined as $100 \cdot \frac{z_{run} - z_{best}}{z_{best}}$, where z_{run} represents the objective value achieved in that run and z_{best} is the best-known solution obtained using any of the approaches for the specific instance. The boxplot shows evidence that both BRKGA and BRKGA$_m$ are very robust, as all the nonzero deviations are classified as outliers, i.e., both

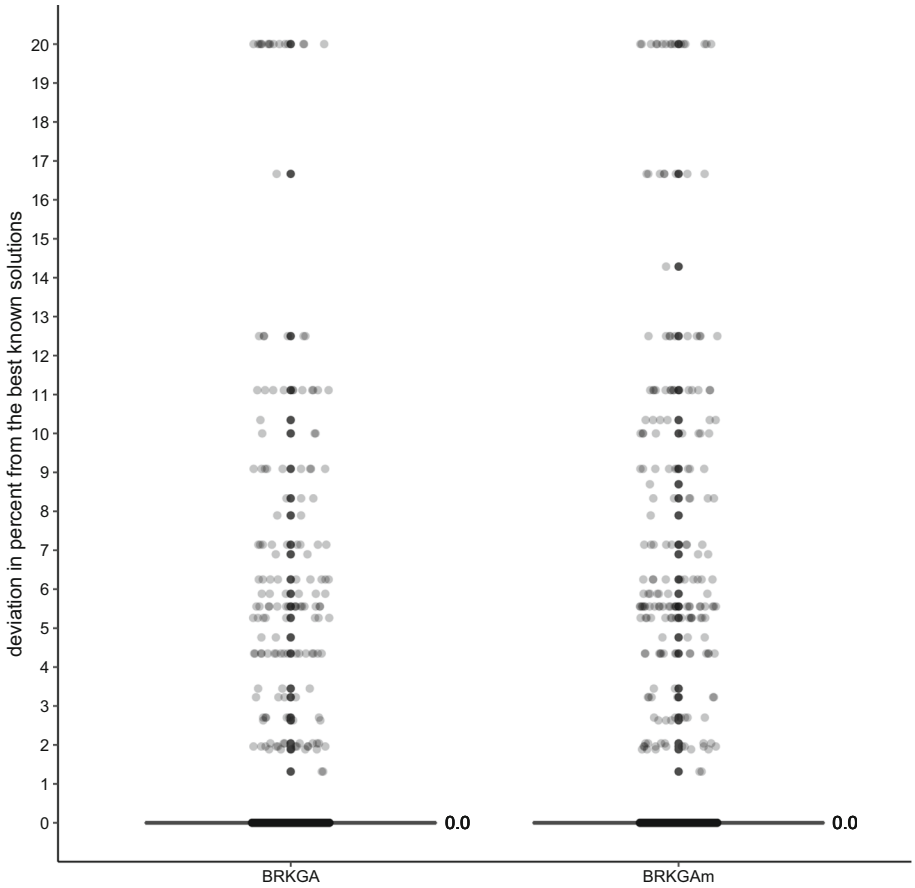

Fig. 7 Boxplot summarizing the deviation in percent from the best-known solutions for the 2070 runs (ten independent runs for each of the 207 original instances) performed using BRKGA and BRKGA$_m$

the upper and lower whiskers are null and, consequently, the median and the second and third quartiles are also null.

4.4 Results using a new benchmark set with more challenging large instances

This section assesses how the proposed BRKGA implementations perform on even more challenging large instances. To accomplish that, we introduce a new benchmark set composed of eight miscellaneous sparse graphs with $|V| \in [494, 2851]$ and $d(G) \in [0.238, 2.951]$, and 13 dense graphs from the DIMACS Implementation Challenges with $|V| \in [500, 2000]$ and $d(G) \in [24.475, 77.131]$. The sparse and dense graphs are summarized in Tables 5 and 6, respectively. Their columns are the same as in Table 2. These graphs are available in Rossi and Ahmed (2015a), Rossi and Ahmed (2015b). Instances are considered for the values of $\gamma \in \{0.999, 0.950, 0.900, 0.800, 0.700, 0.600, 0.500, 0.400, 0.300\}$ that are at least 0.01 larger than the corresponding graphs' densities. In total, there are 72 sparse and 78 dense instances.

Table 5 New benchmark set with challenging large sparse instances

Input graph	$\|V\|$	$\|E\|$	$d(G)$
494bus	494	586	0.481
662bus	662	906	0.414
email-dnc-corecipient	906	12,100	2.951
email	1133	5, 451	0.850
polblogs	1490	16,715	1.507
bcsstk13	2003	40,940	2.042
hamsterster	2400	16,600	0.577
data	2851	15,093	0.371

Table 6 New benchmark set with challenging large dense instances

Input graph	$\|V\|$	$\|E\|$	$d(G)$
p-hat500-1	500	31,569	25.306
p-hat500-2	500	62,946	50.458
p-hat500-3	500	93,800	75.190
keller5	766	225,990	77.131
brock800-3	800	207,333	64.873
p-hat1000-1	1000	122,253	24.475
p-hat1000-2	1000	244,799	49.009
san1000	1000	250,500	50.150
p-hat1000-3	1000	371,746	74.424
p-hat1500-1	1500	284,923	25.343
p-hat1500-2	1500	568,960	50.608
p-hat1500-3	1500	847,244	75.361
C2000-5	2000	999,836	50.017

For the experiments reported in this section, we only compare the results obtained by MSH, BRKGA, and BRKGA$_m$. We do not provide those using the IP formulations given their limitations to tackle such large instances, considering their $O(|V|^3)$ variables; see Melo et al. (2022). The parameter settings for MSH, BRKGA, and BRKGA$_m$ were the same as those described in Sect. 4.2. Ten independent runs were executed using each of the approaches (MSH, BRKGA, and BRKGA$_m$) for each instance. A time limit of $|V|$ seconds was imposed on each run.

4.4.1 Results for the sparse graphs

Table 7 summarizes the results for the large sparse instances assembled by the input graphs. The columns are the same as in Table 3. All the reported values are averaged over the nine instances (given by the different values of γ) for the specified input graph. The results for MSH and BRKGA correspond to the ten independent executions. Detailed results for each instance are available in Appendix B.

The results show that BRKGA$_m$ reaches the lowest minimum values for all the input graphs except 662_bus. Furthermore, BRKGA$_m$ reaches maximum values that are always lower than the minimum values achieved by MSH. Besides, for all the input graphs, the differences

Table 7 Summary of the average results for the new large sparse instances, assembled by the input graphs

Input graph	MSH				BRKGA				BRKGA$_m$			
	Min	Avg	max	ttb (s)	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
494_bus	238.56	241.49	243.44	205.7	218.67	221.18	223.78	268.3	**215.67**	217.43	219.00	453.8
662_bus	307.33	311.48	315.11	273.0	**280.33**	283.54	288.33	492.2	282.67	286.39	290.33	630.9
email-dnc-corecipient	446.33	450.19	453.22	464.4	419.67	424.47	430.89	700.4	**393.89**	397.02	400.11	788.0
email	498.33	504.40	508.89	534.4	453.00	462.90	472.11	1100.8	**438.78**	443.84	448.33	1063.9
polblogs	638.89	644.26	649.44	687.3	591.44	599.52	608.67	1172.6	**548.67**	552.71	556.22	1112.9
bcsstk13	200.33	203.64	206.78	1080.0	205.11	209.07	212.67	1425.3	**175.00**	177.28	179.22	1321.8
soc-hamsterster	719.44	728.84	734.33	1554.5	711.11	719.98	727.22	2149.8	**624.44**	629.78	633.67	2242.6
data	547.11	554.12	560.44	1874.4	566.33	572.84	577.56	1914.6	**531.67**	535.16	538.33	2037.1

Table 8 Summary of the average results for the new large sparse instances, gathered by the values of γ

γ	MSH				BRKGA				BRKGA$_m$			
	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
0.999	572.25	576.46	580.12	853.9	**540.88**	544.26	547.25	1207.4	556.62	558.79	560.75	1225.9
0.950	567.50	572.21	576.50	818.8	**538.38**	542.27	545.88	1238.7	552.38	555.08	557.25	1224.1
0.900	553.75	558.35	562.25	814.7	**527.50**	531.61	535.12	1266.4	535.38	538.12	540.62	1285.8
0.800	512.00	516.69	519.88	757.2	494.12	498.53	504.00	1216.4	**485.75**	488.75	491.50	1185.8
0.700	487.38	493.21	496.88	973.7	474.12	480.75	486.75	1136.7	**451.00**	454.25	456.75	1089.1
0.600	406.75	412.48	416.62	842.8	388.75	395.27	401.62	1215.2	**343.50**	347.59	351.38	1213.1
0.500	363.25	368.77	373.88	879.4	350.25	355.59	363.25	1129.9	**285.25**	290.65	295.38	1273.7
0.400	322.12	328.06	333.12	796.1	311.50	320.43	330.12	1023.3	**238.75**	243.04	247.50	1174.2
0.300	260.88	266.99	271.38	771.3	250.88	261.48	269.88	943.1	**163.50**	168.30	172.25	1185.7

between the values obtained by BRKGA in the columns min, avg, and max are relatively small, indicating the robustness of the approach. It is noteworthy that, differently from what happened with the original instances, that have up to 138 vertices and are thus considerably smaller, BRKGA$_m$ reaches better results than those achieved by BRKGA, indicating the benefits of the proposed merging operations.

Table 8 summarizes the results gathered by the values of γ. The columns are the same as in Table 4. The results show that BRKGA reaches the lowest minimum values for the largest settings of γ, while BRKGA$_m$ is the best performing approach for the values of γ below 0.800.

The boxplot with jittered points illustrated in Fig. 8 summarizes the deviations in percent from the best-known solutions for all the 720 runs for the instances corresponding to the large sparse graphs. The boxplot shows that the solutions obtained in multiple runs by BRKGA$_m$ do not have significant deviations from the best-known solutions, as they are always below 10%. BRKGA does not have a similar behavior, since some runs can reach deviations above 100%.

4.4.2 Results for the dense graphs

Table 9 summarizes the results for the large dense instances assembled by the input graphs. The columns are the same as in Table 3, except for the second column (#inst) that provides the number of instances for the corresponding input graphs, given by the different values of γ that are at least 0.01 larger than the graphs' densities. All the reported values are averaged over the number of instances for the specified input graph. The results for MSH, BRKGA, and BRKGA$_m$ correspond to the ten independent executions.

The results show that BRKGA$_m$ reaches the lowest minimum values for all the input graphs but the largest one (C2000-5). Furthermore, BRKGA$_m$ reaches the lowest average values for all the input graphs. As observed for the large sparse graphs, the differences between the values obtained by BRKGA in the columns min, avg, and max are often small.

Table 10 summarizes the results gathered by the values of γ. The columns are the same as in Table 4, except for the second column (#inst) that gives the number of graphs with density at least 0.01 below the corresponding value of γ. The results show that BRKGA$_m$ obtains the lowest minimum values for all the values of γ, except $\gamma = 0.999$.

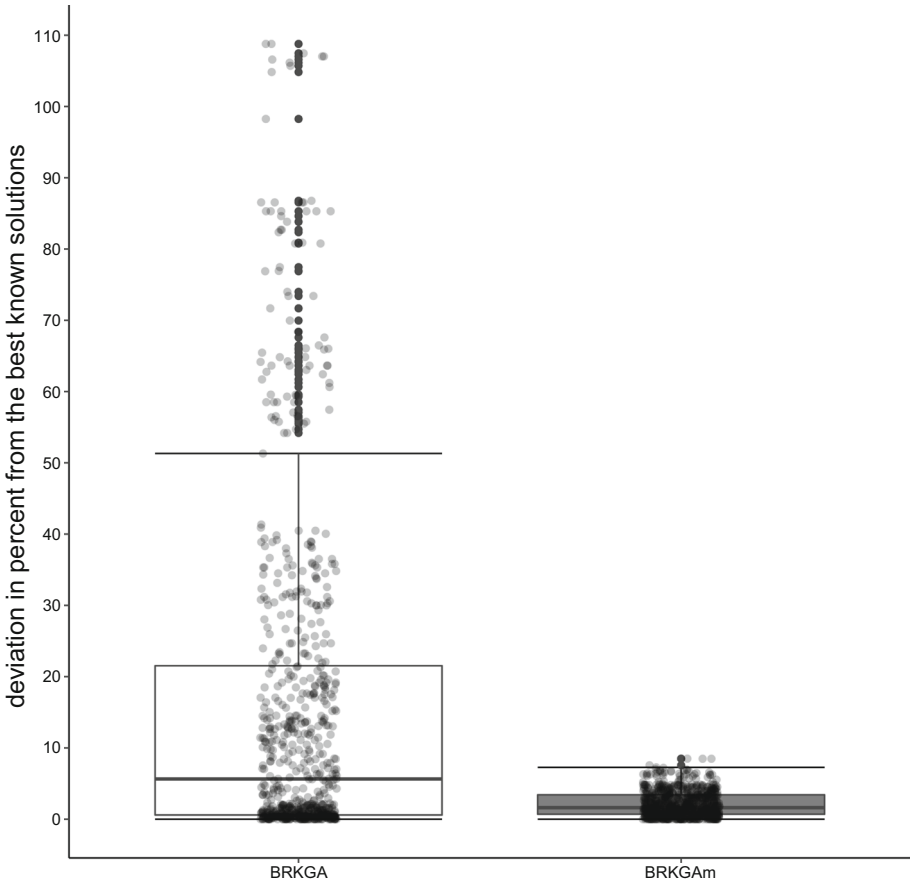

Fig. 8 Boxplot summarizing the deviation in percent from the best-known solutions for the 720 runs (ten independent runs for each of the 72 instances) performed using BRKGA and BRKGA$_m$ considering the large sparse instances

The boxplot with jittered points illustrated in Fig. 9 summarizes the deviations in percent from the best-known solutions for all the 780 runs for the instances corresponding to the large dense graphs. The boxplot shows that, as observed for the large sparse instances, the obtained solutions in multiple runs by BRKGA$_m$ do not have large deviations from the best-known solutions, differently from what happens for BRKGA. Once again, the results show the benefits of the proposed merging operations in the decoder.

4.5 Summary of the behaviors of BRKGA and BRKGA$_m$

We remark that, given the caracteristics of the two variants, BRKGA potentially explores a larger variety of solutions while BRKGA$_m$ tends to converge faster to locally optimal solutions. For the original smaller instances, that have up to 138 vertices, the ability of exploring more solutions allows BRKGA to perform slightly better. However, when it comes to the larger instances, with 500 or more vertices, such an advantage does not hold. In that case, the faster convergence of BRKGA$_m$ provides competitiveness as very low-quality solutions

Table 9 Summary of the average results for the new large dense instances assembled by the input graphs

Input graph	#inst	MSH				BRKGA				BRKGA$_m$			
		Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
p-hat500-1	9	75.33	76.87	78.22	224.8	69.22	70.63	71.89	276.0	**64.67**	65.69	66.56	246.1
p-hat500-2	6	59.83	61.35	62.67	206.0	53.50	55.03	55.83	248.1	**48.33**	49.43	50.00	220.4
p-hat500-3	4	25.25	25.97	26.50	156.5	24.50	24.80	25.25	95.9	**23.25**	23.60	24.00	93.6
keller5	4	30.50	31.30	32.00	316.1	30.00	30.78	31.25	347.6	**29.50**	29.65	30.00	225.7
brock800-3	5	40.20	40.78	41.60	210.4	40.20	40.52	41.20	212.9	**39.60**	39.94	40.60	206.6
p-hat1000-1	9	141.11	143.34	145.33	534.5	129.00	130.68	132.00	596.7	**121.56**	122.94	124.44	600.7
p-hat1000-2	6	112.67	115.10	117.33	467.6	101.00	102.67	104.33	604.8	**92.17**	92.83	94.00	585.1
san1000	6	48.83	49.23	49.67	347.3	48.83	49.10	49.33	199.1	**48.17**	48.50	48.67	173.8
p-hat1000-3	4	44.50	45.62	46.75	404.7	43.00	43.75	44.00	255.2	**41.50**	42.17	42.50	277.6
p-hat1500-1	9	192.56	196.00	198.56	862.7	177.00	178.69	180.56	994.3	**168.11**	169.18	170.44	939.8
p-hat1500-2	6	151.33	154.37	156.67	972.6	136.00	137.63	139.33	915.5	**124.17**	125.57	126.50	844.6
p-hat1500-3	4	58.25	59.55	60.50	980.5	56.50	57.23	57.75	412.5	**54.50**	55.30	55.75	521.4
C2000-5	6	**124.50**	125.63	126.67	1015.9	125.17	125.95	126.67	623.1	125.00	125.47	126.00	587.1

Table 10 Summary of the average results for the new large dense instances, gathered by the values of γ

γ	*#inst*	MSH				BRKGA				BRKGA$_m$			
		Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
0.999	13	142.15	143.86	145.15	519.2	**131.77**	133.00	134.23	483.3	132.46	133.42	134.46	511.3
0.950	13	129.69	131.16	132.69	485.3	121.54	122.83	123.85	496.1	**121.23**	122.28	123.23	493.9
0.900	13	106.54	108.60	110.00	544.2	100.15	101.37	102.54	491.5	**97.38**	98.52	99.46	494.6
0.800	13	74.00	75.12	76.46	481.9	68.15	69.26	70.15	459.8	**63.46**	64.03	64.69	439.6
0.700	9	77.89	80.41	82.00	497.8	73.89	74.92	76.00	505.3	**62.00**	62.74	63.78	455.4
0.600	8	57.38	58.71	60.00	630.7	51.25	52.26	53.12	477.7	**44.00**	44.56	44.88	370.4
0.500	3	72.67	75.10	76.67	444.6	69.33	70.27	71.00	500.3	**57.67**	58.67	59.33	534.8
0.400	3	52.33	54.33	56.00	645.4	49.00	50.20	51.33	505.9	**33.33**	33.80	34.00	396.8
0.300	3	27.67	29.43	32.00	648.3	23.33	24.43	25.33	456.2	**12.33**	12.60	13.00	328.1

are potentially avoided, obtaining significantly better solutions than BRKGA in the same running times for most instances.

Finally, we performed some tests for statistical significance regarding the relative behaviors of BRKGA and BRKGA$_m$. They took into consideration the deviations from the best-known solutions already defined earlier in this section. Namely, we considered three populations: (A) the 2070 observations for the 207 original instances, (B) the 720 observations for the 72 large sparse instances, and (C) the 780 observations for the 78 large dense instances. First, as normality cannot be assumed, we applied the Shapiro-Wilk test (Shapiro & Wilk, 1965) to each of the three populations to check for normality. The null hypothesis that the data has normal distribution was rejected with a p-value smaller than 2.2×10^{-16} for all of them. After that, we applied the nonparametric Wilcoxon-Mann–Whitney test (Mann & Whitney, 1947) to verify whether there is statistical evidence that the proposed merging operations allow BRKGA$_m$ to outperform BRKGA. The null hypothesis was that the devia-

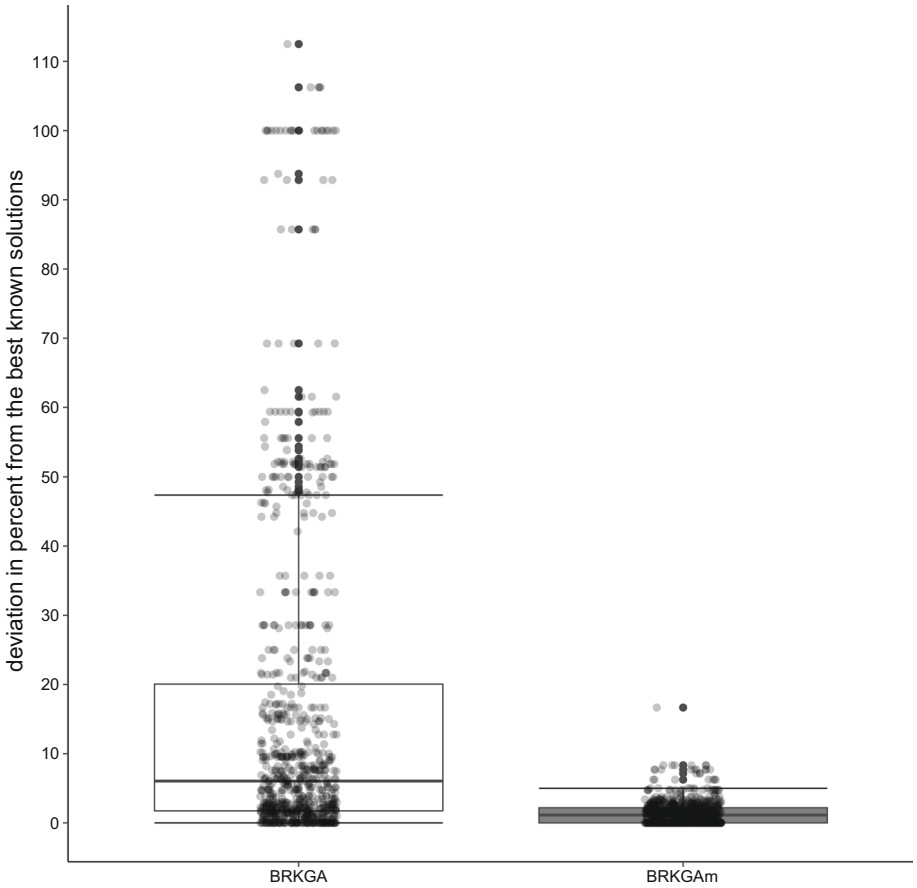

Fig. 9 Boxplot summarizing the deviation in percent from the best-known solutions for the 780 runs (ten independent runs for each of the 78 instances) performed using BRKGA and BRKGA$_m$, considering the large dense instances

tions achieved by BRKGA are less than or equal to those obtained by BRKGA$_m$. There was not enough evidence to reject the null hypothesis for the original instances, with a p-value larger than 0.992, implying that BRKGA performs at least as good as BRKGA$_m$. Contrarily, the null hypothesis was rejected for each of the two sets of large instances with a p-value smaller than 2.2×10^{-16}. Thus, there is statistical evidence that BRKGA$_m$ performs strictly better than BRKGA for the larger instances.

5 Concluding remarks

This paper considered the minimum quasi-clique partitioning problem (MQCPP). First, we showed that MQCPP and the minimum vertex quasi-clique covering problem do not share an equivalence relationship, differently from what happens between the minimum clique partitioning problem and the minimum vertex clique covering problem. Second, we proposed a biased random-key genetic algorithm for the problem that relies on an efficient partitioning

decoder. Furthermore, a new benchmark set of larger and more challenging instances is proposed for MQCPP, for which feasible solutions are provided for the first time.

The proposed BRKGA's decoder can merge quasi-cliques during its execution and runs in $O(|V|^3)$. Whenever one chooses not to perform merge operations, the decoder can be implemented to run in $O(|V|\log|V| + |E|)$. The computational experiments show that the two BRKGA variants are able to generate high-quality solutions in low computational times. Solutions at least as good as the ones available in the literature were obtained for all the available benchmark instances, with new best results achieved for 20.3% of them, i.e., for 42 out of the 207 benchmark instances. Furthermore, the two BRKGA variants are very robust as they obtain low deviations from the best-obtained solutions, with all the nonzero deviations being statistically characterized as outliers. The results for the new large benchmark instances indicate that the complete BRKGA with merging operations obtains solutions that outperform the existing multi-start heuristic. The results also evidence the benefits of the proposed merging operations. Besides, robustness can also be observed in its behavior for such instances as low deviations from the best-known solutions are observed when multiple executions are performed.

Acknowledgements The work of Rafael A. Melo was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) research grant 314662/2020-0. The work of Celso C. Ribeiro was partially supported by CNPq research grant 309869/2020-0 and by FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro) research grant E-26/200.926/2021. The work of Jose A. Riveaux was sponsored by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) research grant 2022/06747-0.

Data Availability The data that support the findings of this study are available from the corresponding author upon request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A: Detailed results for the benchmark instances of Melo et al. (2022)

Table 11 details the results obtained for the original instances. The first two columns indicate the input graph and the value of γ. The third column provides the best result obtained in Melo et al. (2022) using any formulations. The following columns show, for MSH and BRKGA, the minimum (min), average (avg), and maximum (max) objective values over the ten independent runs, as well as the average time in seconds to reach the best-obtained solution (ttb).

Table 11 Detailed results obtained by the approaches for the original instances

Input graph	γ	IP	MSH				BRKGA				BRKGA$_m$			
		Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
Memento	0.999	10	10	10.0	10	0.0	10	10.0	10	0.0	10	10.0	10	0.0
Memento	0.950	10	10	10.0	10	0.0	10	10.0	10	0.0	10	10.0	10	0.0
Memento	0.900	10	10	10.0	10	0.0	10	10.0	10	0.0	10	10.0	10	0.0
Memento	0.800	9	9	9.0	9	0.0	9	9.0	9	0.0	9	9.0	9	0.0
Memento	0.700	9	9	9.0	9	0.0	9	9.0	9	0.0	9	9.0	9	0.0
Memento	0.600	8	8	8.0	8	0.0	8	8.0	8	0.0	8	8.0	8	0.0
Memento	0.500	6	6	6.0	6	0.0	6	6.0	6	0.0	6	6.0	6	0.0
Memento	0.400	5	5	5.0	5	0.0	5	5.0	5	0.0	5	5.0	5	0.0
Memento	0.300	3	3	3.0	3	0.0	3	3.0	3	0.0	3	3.0	3	0.0
The_X_Files	0.999	14	14	14.0	14	0.0	14	14.0	14	0.0	14	14.0	14	0.0
The_X_Files	0.950	14	14	14.0	14	0.0	14	14.0	14	0.0	14	14.0	14	0.0
The_X_Files	0.900	14	14	14.0	14	0.0	14	14.0	14	0.0	14	14.0	14	0.0
The_X_Files	0.800	13	13	13.0	13	0.0	13	13.0	13	0.0	13	13.0	13	0.0
The_X_Files	0.700	11	11	11.0	11	0.0	11	11.0	11	0.0	11	11.0	11	0.0
The_X_Files	0.600	9	9	9.0	9	0.0	9	9.0	9	0.0	9	9.0	9	0.0
The_X_Files	0.500	6	6	6.0	6	0.8	6	6.0	6	0.0	6	6.0	6	0.0
The_X_Files	0.400	4	4	4.3	5	129.7	4	4.0	4	0.2	4	4.0	4	0.3
The_X_Files	0.300	3	3	3.0	3	0.7	3	3.0	3	0.0	3	3.0	3	0.0
Alien_3	0.999	11	11	11.0	11	0.0	11	11.0	11	0.0	11	11.0	11	0.0
Alien_3	0.950	11	11	11.0	11	0.0	11	11.0	11	0.0	11	11.0	11	0.0
Alien_3	0.900	10	10	10.0	10	0.0	10	10.0	10	0.0	10	10.0	10	0.0
Alien_3	0.800	9	9	9.0	9	0.0	9	9.0	9	0.0	9	9.0	9	0.0
Alien_3	0.700	6	7	7.0	7	2.6	6	6.0	6	0.0	6	6.0	6	0.0
Alien_3	0.600	5	5	5.0	5	0.1	5	5.0	5	0.0	5	5.0	5	0.0
Alien_3	0.500	3	4	4.0	4	0.2	3	3.0	3	0.1	3	3.0	3	0.1
Alien_3	0.400	2	3	3.0	3	0.1	2	2.0	2	0.0	2	2.0	2	0.0
Alien_3	0.300	2	2	2.0	2	0.0	2	2.0	2	0.0	2	2.0	2	0.0
high-tech	0.999	16	16	16.0	16	0.1	16	16.0	16	0.0	16	16.0	16	0.0
high-tech	0.950	16	16	16.0	16	0.1	16	16.0	16	0.0	16	16.0	16	0.0
high-tech	0.900	15	15	15.0	15	0.2	15	15.0	15	0.0	15	15.0	15	0.0
high-tech	0.800	14	14	14.0	14	0.4	14	14.0	14	0.0	14	14.0	14	0.0
high-tech	0.700	12	12	12.0	12	23.6	12	12.0	12	0.2	12	12.0	12	0.2
high-tech	0.600	8	8	8.0	8	42.4	8	8.0	8	0.0	8	8.0	8	0.0
high-tech	0.500	6	6	6.0	6	49.7	6	6.0	6	0.1	6	6.0	6	0.1
high-tech	0.400	4	4	4.0	4	62.4	4	4.0	4	0.1	4	4.0	4	0.2
high-tech	0.300	3	3	3.0	3	4.9	3	3.0	3	0.0	3	3.0	3	0.0

Table 11 continued

Input graph	γ	IP	MSH				BRKGA				BRKGA$_m$			
		Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
karate	0.999	20	20	20.0	20	0.0	20	20.0	20	0.1	20	20.0	20	0.1
karate	0.950	20	20	20.0	20	0.0	20	20.0	20	0.1	20	20.0	20	0.1
karate	0.900	19	19	19.0	19	0.0	19	19.0	19	0.1	19	19.0	19	0.1
karate	0.800	17	17	17.0	17	0.6	17	17.0	17	0.1	17	17.0	17	0.1
karate	0.700	15	15	15.0	15	13.4	15	15.0	15	0.4	15	15.0	15	0.5
karate	0.600	10	11	11.0	11	44.9	10	10.0	10	0.7	10	10.0	10	0.7
karate	0.500	7	8	8.5	9	75.2	7	7.0	7	0.3	7	7.0	7	0.6
karate	0.400	5	5	5.1	6	83.4	5	5.0	5	0.1	5	5.0	5	0.1
karate	0.300	3	4	4.0	4	15.1	3	3.0	3	3.9	3	3.0	3	21.6
mexican	0.999	13	13	13.0	13	0.7	13	13.0	13	0.1	13	13.0	13	0.1
mexican	0.950	13	13	13.0	13	0.7	13	13.0	13	0.1	13	13.0	13	0.1
mexican	0.900	12	12	12.0	12	2.7	12	12.0	12	0.1	12	12.0	12	0.1
mexican	0.800	10	10	10.0	10	8.7	10	10.0	10	0.1	10	10.0	10	0.1
mexican	0.700	8	8	8.1	9	65.7	8	8.0	8	0.1	8	8.0	8	0.1
mexican	0.600	6	6	6.3	7	59.9	6	6.0	6	0.1	6	6.0	6	0.1
mexican	0.500	5	5	5.0	5	3.6	5	5.0	5	0.0	5	5.0	5	0.1
mexican	0.400	3	3	3.0	3	24.0	3	3.0	3	0.1	3	3.0	3	0.1
mexican	0.300	2	2	2.0	2	1.1	2	2.0	2	0.0	2	2.0	2	0.0
sawmill	0.999	18	18	18.0	18	0.0	18	18.0	18	0.1	18	18.0	18	0.1
sawmill	0.950	18	18	18.0	18	0.0	18	18.0	18	0.1	18	18.0	18	0.1
sawmill	0.900	18	18	18.0	18	0.0	18	18.0	18	0.1	18	18.0	18	0.1
sawmill	0.800	16	16	16.0	16	0.0	16	16.0	16	0.1	16	16.0	16	0.1
sawmill	0.700	16	16	16.0	16	0.0	16	16.0	16	0.1	16	16.0	16	0.1
sawmill	0.600	11	11	11.0	11	0.2	11	11.0	11	0.0	11	11.0	11	0.1
sawmill	0.500	8	8	8.0	8	5.0	8	8.0	8	0.1	8	8.0	8	0.1
sawmill	0.400	6	6	6.0	6	14.2	6	6.0	6	0.0	6	6.0	6	0.1
sawmill	0.300	4	4	4.0	4	94.3	4	4.0	4	0.7	4	4.0	4	2.3
tailorS1	0.999	17	17	17.0	17	0.0	17	17.0	17	0.1	17	17.0	17	0.1
tailorS1	0.950	17	17	17.0	17	0.0	17	17.0	17	0.1	17	17.0	17	0.1
tailorS1	0.900	15	15	15.0	15	2.1	15	15.0	15	0.1	15	15.0	15	0.1
tailorS1	0.800	12	12	12.0	12	90.5	12	12.0	12	0.1	12	12.0	12	0.1
tailorS1	0.700	10	11	11.0	11	50.3	10	10.0	10	0.2	10	10.0	10	0.2
tailorS1	0.600	7	7	7.9	8	24.3	7	7.0	7	0.2	7	7.0	7	0.2
tailorS1	0.500	5	5	5.5	6	82.7	5	5.0	5	0.1	5	5.0	5	0.1
tailorS1	0.400	3	3	3.8	4	48.6	3	3.0	3	0.1	3	3.0	3	0.1
tailorS1	0.300	2	2	2.0	2	9.1	2	2.0	2	0.1	2	2.0	2	0.1

Table 11 continued

Input graph	γ	IP	MSH				BRKGA				BRKGA$_m$			
		Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
chesapeake	0.999	17	17	17.0	17	0.8	17	17.0	17	0.1	17	17.0	17	0.1
chesapeake	0.950	17	17	17.0	17	0.8	17	17.0	17	0.1	17	17.0	17	0.1
chesapeake	0.900	16	16	16.8	17	17.8	16	16.0	16	15.1	16	16.0	16	14.7
chesapeake	0.800	13	14	14.0	14	43.5	13	13.0	13	2.3	13	13.0	13	4.7
chesapeake	0.700	12	12	12.0	12	19.6	11	11.0	11	0.9	11	11.0	11	0.6
chesapeake	0.600	7	7	7.8	8	56.9	7	7.0	7	0.1	7	7.0	7	0.2
chesapeake	0.500	5	5	5.9	6	28.4	5	5.0	5	0.1	5	5.0	5	0.1
chesapeake	0.400	3	3	3.5	4	117.4	3	3.0	3	0.1	3	3.0	3	0.5
chesapeake	0.300	2	2	2.0	2	6.7	2	2.0	2	0.1	2	2.0	2	0.1
Batman_Returns	0.999	20	20	20.0	20	0.0	20	20.0	20	0.1	20	20.0	20	0.1
Batman_Returns	0.950	20	20	20.0	20	0.0	20	20.0	20	0.1	20	20.0	20	0.1
Batman_Returns	0.900	20	20	20.0	20	0.0	20	20.0	20	0.1	20	20.0	20	0.1
Batman_Returns	0.800	17	17	17.0	17	27.6	17	17.0	17	0.1	17	17.0	17	0.1
Batman_Returns	0.700	15	16	16.0	16	1.2	15	15.0	15	0.2	15	15.0	15	0.1
Batman_Returns	0.600	10	10	10.7	11	95.7	10	10.0	10	0.1	10	10.0	10	0.1
Batman_Returns	0.500	8	9	9.9	10	42.2	8	8.0	8	1.1	8	8.0	8	0.7
Batman_Returns	0.400	6	7	7.5	8	49.5	6	6.0	6	0.3	6	6.0	6	0.3
Batman_Returns	0.300	4	4	4.8	5	42.4	4	4.0	4	2.6	4	4.0	4	6.5
attiro	0.999	27	27	27.0	27	9.6	27	27.0	27	0.2	27	27.0	27	0.2
attiro	0.950	27	27	27.0	27	9.7	27	27.0	27	0.2	27	27.0	27	0.2
attiro	0.900	27	27	27.0	27	9.7	27	27.0	27	0.2	27	27.0	27	0.2
attiro	0.800	24	24	24.0	24	3.7	24	24.0	24	0.2	24	24.0	24	0.2
attiro	0.700	21	21	21.0	21	30.0	21	21.0	21	0.3	21	21.0	21	0.3
attiro	0.600	14	14	14.5	15	58.1	13	13.0	13	4.1	13	13.0	13	4.5
attiro	0.500	12	12	12.7	13	57.9	11	11.0	11	36.1	11	11.0	11	73.3
attiro	0.400	9	10	10.0	10	97.7	9	9.0	9	1.5	9	9.0	9	2.5
attiro	0.300	6	7	7.0	7	52.6	6	6.0	6	81.9	6	6.0	6	100.4
krebs	0.999	33	33	33.0	33	0.2	33	33.0	33	0.2	33	33.0	33	0.2
krebs	0.950	33	33	33.0	33	0.1	33	33.0	33	0.2	33	33.0	33	0.2
krebs	0.900	31	31	31.1	32	54.4	31	31.0	31	0.5	31	31.0	31	0.5
krebs	0.800	26	27	27.0	27	12.1	26	26.0	26	4.0	26	26.0	26	2.9
krebs	0.700	25	25	25.0	25	71.8	25	25.0	25	0.2	25	25.0	25	0.2
krebs	0.600	18	18	18.9	19	67.0	17	17.0	17	1.1	17	17.0	17	0.8
krebs	0.500	12	15	15.4	16	45.2	12	12.0	12	1.2	12	12.0	12	1.2
krebs	0.400	8	11	11.9	12	115.3	8	8.0	8	1.8	8	8.0	8	5.4
krebs	0.300	5	8	8.4	9	34.9	5	5.9	6	16.6	5	5.9	6	2.2

Table 11 continued

Input graph	γ	IP	MSH				BRKGA				BRKGA$_m$			
		Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
dolphins	0.999	28	28	28.0	28	24.0	28	28.0	28	0.3	28	28.0	28	0.3
dolphins	0.950	28	28	28.0	28	24.3	28	28.0	28	0.3	28	28.0	28	0.3
dolphins	0.900	27	27	27.0	27	40.9	27	27.0	27	0.3	27	27.0	27	0.3
dolphins	0.800	25	25	25.0	25	24.2	25	25.0	25	0.3	25	25.0	25	0.3
dolphins	0.700	23	23	23.2	24	118.2	23	23.0	23	0.3	23	23.0	23	0.4
dolphins	0.600	16	17	17.5	18	64.7	15	15.0	15	22.5	15	15.0	15	33.6
dolphins	0.500	12	14	14.0	14	68.1	11	11.0	11	23.2	11	11.0	11	18.8
dolphins	0.400	8	10	10.8	12	60.9	8	8.0	8	1.6	8	8.0	8	2.3
dolphins	0.300	5	7	7.5	8	63.0	5	5.6	6	74.6	5	5.7	6	31.4
prison	0.999	26	26	26.0	26	20.5	26	26.0	26	0.4	26	26.0	26	0.4
prison	0.950	26	26	26.0	26	20.5	26	26.0	26	0.4	26	26.0	26	0.4
prison	0.900	25	25	25.0	25	2.4	25	25.0	25	0.2	25	25.0	25	0.2
prison	0.800	24	24	24.0	24	51.7	24	24.0	24	0.4	24	24.0	24	0.4
prison	0.700	22	22	22.0	22	71.3	21	21.0	21	1.2	21	21.0	21	2.1
prison	0.600	15	16	16.5	17	47.0	15	15.0	15	1.2	15	15.0	15	1.6
prison	0.500	12	12	12.9	13	60.2	12	12.0	12	1.4	12	12.0	12	1.0
prison	0.400	10	10	10.2	11	95.2	9	9.4	10	85.0	9	9.5	10	73.7
prison	0.300	7	8	8.0	8	27.9	7	7.0	7	28.7	7	7.0	7	24.1
sanjuansur	0.999	35	35	35.0	35	2.3	35	35.0	35	0.2	35	35.0	35	0.2
sanjuansur	0.950	35	35	35.0	35	2.4	35	35.0	35	0.2	35	35.0	35	0.2
sanjuansur	0.900	35	35	35.0	35	2.4	35	35.0	35	0.2	35	35.0	35	0.2
sanjuansur	0.800	31	31	31.0	31	18.3	31	31.0	31	0.5	31	31.0	31	0.5
sanjuansur	0.700	29	29	29.0	29	85.6	29	29.0	29	0.6	29	29.0	29	0.6
sanjuansur	0.600	18	20	20.0	20	42.5	18	18.0	18	1.4	18	18.0	18	2.0
sanjuansur	0.500	15	15	16.2	17	78.2	14	14.0	14	11.3	14	14.0	14	19.5
sanjuansur	0.400	12	12	12.6	13	124.9	11	11.0	11	26.1	11	11.0	11	30.4
sanjuansur	0.300	8	9	9.7	10	68.6	8	8.5	9	67.2	8	8.8	9	27.0
jean	0.999	35	35	35.0	35	0.2	35	35.0	35	0.3	35	35.0	35	0.3
jean	0.950	35	35	35.0	35	0.1	35	35.0	35	0.3	35	35.0	35	0.3
jean	0.900	33	33	33.0	33	1.4	33	33.0	33	0.3	33	33.0	33	0.3
jean	0.800	30	30	30.2	31	92.0	30	30.0	30	0.3	30	30.0	30	0.3
jean	0.700	27	27	28.0	29	79.7	27	27.0	27	0.3	27	27.0	27	0.3
jean	0.600	20	22	23.2	24	32.7	19	19.0	19	4.8	19	19.0	19	2.5
jean	0.500	13	18	19.0	20	88.4	12	12.0	12	30.6	12	12.0	12	15.3
jean	0.400	8	13	15.1	16	68.9	8	8.0	8	2.1	8	8.0	8	2.8
jean	0.300	5	9	9.8	11	142.5	5	5.0	5	16.3	5	5.0	5	11.5

Table 11 continued

Input graph	γ	IP	MSH				BRKGA				BRKGA$_m$			
		Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
3-FullIns_3	0.999	37	39	39.0	39	86.5	37	37.3	38	103.3	37	37.3	38	102.1
3-FullIns_3	0.950	37	39	39.0	39	85.8	37	37.3	38	103.4	37	37.3	38	102.0
3-FullIns_3	0.900	37	38	38.7	39	47.0	37	37.0	37	17.5	37	37.0	37	17.3
3-FullIns_3	0.800	36	37	37.0	37	89.6	36	36.0	36	7.9	36	36.0	36	7.8
3-FullIns_3	0.700	35	36	36.1	37	46.6	35	35.0	35	4.5	35	35.0	35	4.6
3-FullIns_3	0.600	17	18	18.1	19	60.9	16	16.0	16	67.8	16	16.0	16	51.3
3-FullIns_3	0.500	13	13	13.7	14	26.3	12	12.3	13	46.8	12	12.6	13	76.9
3-FullIns_3	0.400	10	10	10.0	10	55.1	9	9.0	9	54.5	9	9.0	9	58.0
3-FullIns_3	0.300	6	7	7.0	7	36.2	6	6.0	6	36.4	6	6.0	6	52.6
david	0.999	36	36	36.0	36	0.2	36	36.0	36	0.3	36	36.0	36	0.3
david	0.950	34	34	34.9	35	30.4	34	34.0	34	1.3	34	34.0	34	1.3
david	0.900	33	33	33.2	34	108.4	33	33.0	33	0.4	33	33.0	33	0.4
david	0.800	29	30	30.4	31	79.8	29	29.0	29	0.6	29	29.0	29	0.6
david	0.700	26	28	28.5	29	107.7	23	23.2	24	102.8	23	23.0	23	162.3
david	0.600	18	24	24.3	25	98.9	17	17.6	18	73.1	17	17.7	18	29.8
david	0.500	14	19	20.5	21	106.7	11	11.9	12	10.3	11	11.9	12	19.3
david	0.400	9	16	17.5	19	87.9	8	8.1	9	88.7	8	8.2	9	83.3
david	0.300	6	12	12.3	13	110.9	6	6.0	6	1.3	6	6.0	6	2.2
myciel6	0.999	48	52	52.2	53	88.1	48	48.0	48	2.6	48	48.0	48	2.6
myciel6	0.950	48	52	52.2	53	88.6	48	48.0	48	2.6	48	48.0	48	2.6
myciel6	0.900	48	52	52.2	53	89.4	48	48.0	48	2.6	48	48.0	48	2.6
myciel6	0.800	48	52	52.2	53	87.9	48	48.0	48	2.6	48	48.0	48	2.6
myciel6	0.700	52	52	52.2	53	87.0	48	48.0	48	2.6	48	48.0	48	2.6
myciel6	0.600	23	30	30.3	31	116.0	19	19.5	20	47.2	20	20.0	20	16.9
myciel6	0.500	18	24	24.7	25	91.7	14	14.9	15	42.7	15	15.1	16	68.9
myciel6	0.400	12	19	20.0	21	86.8	10	10.3	11	61.8	10	10.8	11	12.2
myciel6	0.300	6	14	15.3	16	141.7	6	6.1	7	74.5	6	6.8	7	20.3
4-FullIns_3	0.999	55	55	56.7	57	48.7	55	55.0	55	12.3	55	55.0	55	12.2
4-FullIns_3	0.950	55	56	56.5	57	66.0	55	55.0	55	10.6	55	55.0	55	10.5
4-FullIns_3	0.900	54	56	56.1	57	65.5	53	53.5	54	76.3	53	53.5	54	75.6
4-FullIns_3	0.800	53	54	54.1	55	69.8	52	52.0	52	20.9	51	51.8	52	48.4
4-FullIns_3	0.700	52	52	52.7	53	75.7	49	49.7	50	59.6	49	49.6	50	75.6
4-FullIns_3	0.600	26	27	27.1	28	151.9	23	23.8	24	82.8	23	23.7	24	68.0
4-FullIns_3	0.500	20	20	20.8	21	40.8	18	18.4	19	110.3	19	19.0	19	33.0
4-FullIns_3	0.400	15	15	15.6	16	56.7	14	14.0	14	68.2	14	14.0	14	61.2
4-FullIns_3	0.300	10	11	11.6	12	54.0	9	9.7	10	46.3	9	9.9	10	93.0

Table 11 continued

Input graph	γ	IP	MSH				BRKGA				BRKGA$_m$			
		Best	Min	Avg	Max	ttb (s)	Min	avg	Max	ttb (s)	Min	Avg	Max	ttb (s)
ieeebus	0.999	57	59	59.2	60	113.8	57	57.0	57	5.2	57	57.0	57	5.1
ieeebus	0.950	57	59	59.2	60	109.8	57	57.0	57	5.2	57	57.0	57	5.2
ieeebus	0.900	57	59	59.2	60	110.4	57	57.0	57	5.2	57	57.0	57	5.2
ieeebus	0.800	57	58	58.9	59	115.3	57	57.0	57	7.6	57	57.0	57	7.6
ieeebus	0.700	57	58	58.6	59	78.6	57	57.0	57	4.1	57	57.0	57	4.1
ieeebus	0.600	40	40	40.7	41	109.1	36	36.0	36	30.4	36	36.0	36	32.3
ieeebus	0.500	31	32	33.0	34	98.0	27	27.0	27	67.0	27	27.0	27	80.4
ieeebus	0.400	25	26	27.1	28	150.6	21	21.2	22	53.3	21	21.3	22	106.0
ieeebus	0.300	18	22	22.3	23	101.9	18	18.4	19	112.0	18	18.8	19	71.3
sfi	0.999	65	65	65.0	65	0.2	65	65.0	65	0.8	65	65.0	65	0.8
sfi	0.950	65	65	65.0	65	0.2	65	65.0	65	0.8	65	65.0	65	0.8
sfi	0.900	65	65	65.0	65	0.1	65	65.0	65	0.8	65	65.0	65	0.8
sfi	0.800	61	61	61.0	61	4.1	61	61.0	61	0.7	61	61.0	61	0.7
sfi	0.700	57	57	58.2	59	77.3	57	57.0	57	0.7	57	57.0	57	0.7
sfi	0.600	45	48	48.4	49	89.0	45	45.0	45	1.1	45	45.0	45	1.5
sfi	0.500	32	42	43.0	44	112.4	31	31.4	32	80.0	31	31.6	32	101.8
sfi	0.400	24	35	37.1	39	102.2	23	23.7	24	100.4	23	23.9	25	161.5
sfi	0.300	16	25	27.0	28	133.5	16	16.9	17	46.6	17	17.0	17	68.2
anna	0.999	80	80	80.0	80	3.9	80	80.0	80	1.2	80	80.0	80	1.2
anna	0.950	79	79	79.5	80	57.5	79	79.0	79	2.1	79	79.0	79	2.1
anna	0.900	77	78	78.8	79	26.7	76	76.2	77	128.7	76	76.2	77	141.7
anna	0.800	74	75	75.8	76	80.4	70	70.0	70	11.7	70	70.0	70	26.5
anna	0.700	70	72	73.1	74	86.1	66	66.0	66	52.3	66	66.0	66	29.2
anna	0.600	61	61	62.9	64	160.2	51	51.0	51	68.5	51	51.0	51	83.5
anna	0.500	51	56	57.5	59	156.2	38	39.5	41	119.5	39	39.8	41	136.2
anna	0.400	40	52	52.7	54	96.2	29	30.0	32	161.2	30	31.4	32	119.7
anna	0.300	22	36	38.9	41	126.9	18	18.8	19	89.5	19	19.1	20	84.0

Appendix B: Detailed results for the new large sparse benchmark instances

Table 12 details the results obtained for each large sparse instance. The columns are the same as in Table 11 in Appendix A. The only difference is that the results using the formulations are not available for these large instances.

Table 12 Detailed results obtained by the approaches for the new large sparse instances

Input graph	γ	MSH				BRKGA				BRKGA$_m$			
		min	avg	max	ttb (s)	min	avg	max	ttb (s)	min	avg	max	ttb (s)
494_bus	0.999	293	294.7	296	185.1	278	278.8	279	156.5	279	280.2	281	455.0
494_bus	0.950	293	294.7	296	186.2	278	278.8	279	155.3	279	280.2	281	455.0
494_bus	0.900	293	294.7	296	186.0	278	278.8	279	156.4	279	280.2	281	454.6
494_bus	0.800	293	294.7	296	186.2	278	278.8	279	155.0	279	280.2	281	454.7
494_bus	0.700	293	294.7	296	185.9	278	278.8	279	156.9	279	280.2	281	454.6
494_bus	0.600	214	218.0	221	266.2	187	191.7	197	368.5	185	186.9	189	477.0
494_bus	0.500	179	182.3	186	206.6	151	154.6	160	384.1	142	144.7	149	451.3
494_bus	0.400	155	159.5	161	207.1	129	133.8	140	433.2	117	119.9	122	443.3
494_bus	0.300	134	140.1	143	242.2	111	116.5	122	449.1	102	104.4	106	438.5
662_bus	0.999	379	382.5	386	162.2	354	355.6	357	426.9	366	368.6	371	632.9
662_bus	0.950	379	382.5	386	162.2	354	355.5	357	462.1	366	368.6	371	633.7
662_bus	0.900	379	382.5	386	162.2	354	355.5	357	461.4	366	368.6	371	633.5
662_bus	0.800	378	380.4	383	288.7	352	353.1	355	395.7	365	366.3	369	626.9
662_bus	0.700	371	378.7	382	367.4	350	352.7	356	407.1	362	364.6	367	622.8
662_bus	0.600	278	282.5	285	337.2	247	249.9	252	517.0	247	250.9	256	641.4
662_bus	0.500	230	232.9	238	331.3	198	200.7	205	602.4	185	193.7	199	636.8
662_bus	0.400	196	201.5	205	292.0	167	171.1	176	555.4	153	157.6	166	649.2
662_bus	0.300	176	179.8	185	354.0	147	157.8	180	602.1	134	138.6	143	601.2
email-dnc-corecipient	0.999	498	500.0	501	361.0	495	495.0	495	81.1	495	495.0	495	589.0
email-dnc-corecipient	0.950	497	499.5	501	406.0	492	492.7	494	396.5	490	492.1	494	878.5
email-dnc-corecipient	0.900	495	496.4	498	432.0	483	485.3	487	646.7	483	484.4	486	745.5
email-dnc-corecipient	0.800	482	487.3	489	500.2	469	471.6	476	801.8	462	463.8	466	796.3
email-dnc-corecipient	0.700	475	476.6	479	663.3	449	456.1	465	851.5	440	441.8	443	713.3
email-dnc-corecipient	0.600	435	441.1	445	435.3	410	414.5	423	858.0	380	382.1	387	866.9
email-dnc-corecipient	0.500	414	418.8	426	599.2	381	386.9	397	889.0	338	343.6	348	786.9
email-dnc-corecipient	0.400	391	396.8	401	405.7	341	349.8	367	895.2	292	297.3	303	844.3
email-dnc-corecipient	0.300	330	335.2	339	377.4	257	268.3	274	883.6	165	173.1	179	871.5
email	0.999	613	618.4	622	513.1	559	561.9	566	1091.1	589	592.0	595	1032.8
email	0.950	610	614.7	619	651.2	555	560.7	566	1077.0	590	591.5	595	1074.1
email	0.900	599	606.8	613	657.9	548	551.8	557	1100.6	581	583.4	586	1012.2
email	0.800	569	574.4	578	596.2	527	530.2	535	1117.0	544	548.2	553	1007.1
email	0.700	550	559.6	564	637.9	510	518.6	524	1137.2	515	518.8	522	1136.9
email	0.600	458	462.8	467	325.6	408	419.5	428	1108.4	376	384.7	388	1052.1
email	0.500	412	416.7	420	395.9	369	379.6	394	1116.6	317	323.8	331	1121.5
email	0.400	369	374.6	380	379.1	325	349.2	372	1109.0	264	272.7	280	1052.0
email	0.300	305	311.6	317	652.4	276	294.6	307	1049.8	173	179.5	185	1086.7
polblogs	0.999	746	750.5	755	762.6	684	685.6	687	1161.3	712	714.3	716	1033.6
polblogs	0.950	738	746.3	752	795.3	681	685.3	688	1210.2	711	712.6	714	1052.2
polblogs	0.900	738	741.1	746	562.1	681	682.4	686	1193.0	702	703.6	706	1067.8

Table 12 continued

Input graph	γ	MSH				BRKGA				BRKGA$_m$			
		min	avg	max	ttb (s)	min	avg	max	ttb (s)	min	avg	max	ttb (s)
polblogs	0.800	707	712.5	718	610.0	661	668.3	677	1194.2	660	661.7	664	1154.6
polblogs	0.700	697	698.9	702	748.4	647	658.7	673	1182.7	633	636.5	639	993.7
polblogs	0.600	605	612.1	616	701.6	557	570.5	583	1168.0	480	485.9	491	1198.1
polblogs	0.500	558	564.9	572	763.6	522	530.6	542	1166.3	416	423.1	430	1178.3
polblogs	0.400	518	524.5	532	687.7	487	496.7	512	1184.3	371	375.2	379	1139.1
polblogs	0.300	443	447.5	452	554.6	403	417.6	430	1093.3	253	261.5	267	1199.0
bcsstk13	0.999	312	316.4	321	1317.7	292	297.6	304	1899.2	303	305.2	306	1540.1
bcsstk13	0.950	298	300.1	304	930.6	286	289.1	293	1906.0	286	289.3	291	1209.4
bcsstk13	0.900	285	288.8	291	873.3	278	282.8	288	1886.3	265	269.3	272	1581.2
bcsstk13	0.800	232	237.9	242	783.7	244	247.0	251	1289.7	211	214.4	218	1099.2
bcsstk13	0.700	203	206.1	208	1178.3	215	221.1	224	1451.5	170	172.7	176	1279.2
bcsstk13	0.600	160	162.1	164	1053.7	169	173.4	177	1628.3	126	127.5	129	1383.6
bcsstk13	0.500	132	134.3	137	1159.6	147	149.7	153	1013.9	94	95.7	98	1318.3
bcsstk13	0.400	104	107.7	112	1256.0	123	125.5	127	1030.2	68	68.9	70	1216.3
bcsstk13	0.300	77	79.4	82	1167.2	92	95.4	97	722.7	52	52.5	53	1269.0
soc-hamsterster	0.999	884	889.7	894	1736.4	835	839.7	843	2410.7	861	863.1	866	2208.8
soc-hamsterster	0.950	872	880.4	888	1627.6	830	835.9	843	2354.9	849	854.4	856	2177.3
soc-hamsterster	0.900	859	862.1	865	1658.7	818	827.9	834	2346.8	828	831.9	836	2169.1
soc-hamsterster	0.800	815	817.7	820	1436.8	791	797.7	811	2296.2	760	766.1	767	2252.6
soc-hamsterster	0.700	777	789.3	794	1672.5	770	779.2	787	2297.7	691	698.5	702	2104.4
soc-hamsterster	0.600	683	695.2	702	1615.0	687	692.6	698	2142.4	559	561.9	567	2288.0
soc-hamsterster	0.500	629	640.5	646	1347.6	639	644.2	653	2051.1	462	470.2	476	2359.4
soc-hamsterster	0.400	553	564.9	573	1583.7	578	592.8	600	1842.5	382	386.7	391	2415.6
soc-hamsterster	0.300	403	419.8	427	1312.4	452	469.8	476	1606.1	228	235.2	242	2208.1
data	0.999	853	859.5	866	1793.2	830	839.9	847	2432.5	848	851.9	856	2315.1
data	0.950	853	859.5	866	1791.2	831	840.2	847	2347.6	848	851.9	856	2312.3
data	0.900	782	794.4	803	1985.4	780	788.4	793	2339.8	779	783.6	787	2622.5
data	0.800	620	628.6	633	1655.7	631	641.5	648	2481.6	605	609.3	614	2095.3
data	0.700	533	541.8	550	2336.0	574	580.8	586	1608.7	518	520.9	524	1408.3
data	0.600	421	426.0	433	2008.0	445	450.1	455	1930.8	395	400.8	404	1797.6
data	0.500	352	359.8	366	2231.3	395	398.4	402	1815.9	328	330.4	332	2337.3
data	0.400	291	295.0	301	1557.8	342	344.5	347	1136.6	263	266.0	269	1633.9
data	0.300	219	222.5	226	1510.6	269	271.8	273	1138.2	201	201.6	203	1811.5

Appendix C: Detailed results for the large dense benchmark instances

Table 13 details the results obtained for each new large dense instance. The columns are the same as in Table 12 in Appendix B.

Table 13 Detailed results obtained by the approaches for the new large dense instances

Input graph	γ	MSH				BRKGA				BRKGA$_m$			
		min	avg	max	ttb (s)	min	avg	max	ttb (s)	min	avg	max	ttb (s)
p-hat500-1	0.999	126	128.2	130	258.5	115	117.1	119	334.2	117	117.8	119	360.7
p-hat500-1	0.950	125	126.4	128	247.3	114	116.4	118	355.6	114	116.4	118	337.1
p-hat500-1	0.900	115	116.8	118	211.5	105	106.6	108	318.7	103	104.8	107	280.3
p-hat500-1	0.800	93	94.3	95	154.4	84	85.6	87	354.8	79	80.2	81	267.4
p-hat500-1	0.700	78	79.4	81	219.4	74	74.7	76	239.2	64	65.3	67	306.1
p-hat500-1	0.600	56	57.4	58	237.3	52	53.1	54	217.5	47	47.9	48	230.8
p-hat500-1	0.500	41	42.3	44	180.9	39	40.1	41	268.7	32	32.8	33	237.3
p-hat500-1	0.400	29	31.0	32	265.2	27	28.3	30	280.2	19	19.0	19	115.1
p-hat500-1	0.300	15	16.0	18	248.5	13	13.8	14	114.8	7	7.0	7	80.3
p-hat500-2	0.999	85	86.9	88	246.0	75	76.6	77	265.8	76	76.8	77	313.4
p-hat500-2	0.950	81	82.3	83	142.0	72	73.4	75	272.7	71	72.8	74	290.4
p-hat500-2	0.900	72	73.0	74	143.3	65	66.7	68	289.3	61	62.4	63	296.3
p-hat500-2	0.800	55	56.3	58	195.2	50	51.4	52	250.0	42	43.4	44	155.0
p-hat500-2	0.700	42	44.7	47	256.9	40	40.9	41	244.3	27	27.3	28	205.0
p-hat500-2	0.600	24	24.9	26	252.8	19	21.2	22	166.6	13	13.9	14	62.6
p-hat500-3	0.999	39	39.8	40	93.7	37	37.4	38	101.0	37	37.4	38	109.7
p-hat500-3	0.950	31	31.4	32	176.9	30	30.4	31	154.5	29	29.4	30	150.5
p-hat500-3	0.900	22	23.0	24	146.4	22	22.4	23	73.3	20	20.6	21	93.5
p-hat500-3	0.800	9	9.7	10	208.9	9	9.0	9	54.7	7	7.0	7	20.8
keller5	0.999	58	58.9	60	318.3	56	56.2	57	384.4	56	56.2	57	413.1
keller5	0.950	41	41.6	42	224.2	40	40.7	41	283.2	39	39.4	40	203.2
keller5	0.900	21	22.6	23	444.7	22	22.9	23	304.5	21	21.0	21	165.3
keller5	0.800	2	2.1	3	277.2	2	3.3	4	418.1	2	2.0	2	121.0
brock800-3	0.999	72	72.1	73	168.1	70	70.2	71	180.3	70	70.2	71	200.8
brock800-3	0.950	58	59.5	61	178.1	59	59.5	60	180.0	58	58.9	59	254.0
brock800-3	0.900	43	43.7	45	333.6	43	43.8	45	279.7	43	43.4	44	203.1
brock800-3	0.800	21	21.6	22	156.4	22	22.0	22	207.8	21	21.1	22	245.2
brock800-3	0.700	7	7.0	7	215.8	7	7.1	8	216.6	6	6.1	7	129.9
p-hat1000-1	0.999	239	240.8	243	525.9	217	219.0	221	756.5	219	220.5	222	804.5
p-hat1000-1	0.950	233	235.1	238	587.3	214	216.5	218	572.9	215	217.7	220	604.6
p-hat1000-1	0.900	214	217.6	220	639.9	196	198.1	200	684.4	195	196.4	198	740.7
p-hat1000-1	0.800	173	174.1	176	495.3	156	157.5	159	724.1	149	150.0	152	661.3
p-hat1000-1	0.700	143	146.4	148	684.2	133	135.8	137	625.8	119	121.3	124	675.5
p-hat1000-1	0.600	106	107.1	108	460.1	95	96.3	98	539.5	88	89.5	91	430.6
p-hat1000-1	0.500	76	78.5	80	465.0	73	73.3	74	512.8	60	61.2	62	569.5
p-hat1000-1	0.400	55	57.1	59	450.3	51	52.6	53	507.3	35	35.8	36	448.9
p-hat1000-1	0.300	31	33.4	36	502.9	26	27.0	28	446.7	14	14.1	15	470.6
p-hat1000-2	0.999	160	162.7	164	514.5	142	142.8	145	616.0	142	143.3	145	674.4
p-hat1000-2	0.950	152	153.0	155	498.6	135	137.2	139	617.2	134	135.4	138	683.9
p-hat1000-2	0.900	134	136.9	138	288.4	121	123.5	125	725.4	116	116.9	118	617.2

Table 13 continued

Input graph	γ	MSH				BRKGA				BRKGA_m			
		min	avg	max	ttb (s)	min	avg	max	ttb (s)	min	avg	max	ttb (s)
p-hat1000-2	0.800	103	105.2	108	440.5	92	94.3	96	555.2	82	82.1	83	580.2
p-hat1000-2	0.700	79	83.2	87	483.3	75	76.3	78	527.6	52	52.3	53	464.3
p-hat1000-2	0.600	48	49.6	52	580.4	41	41.9	43	587.5	27	27.0	27	490.8
san1000	0.999	133	133.8	135	603.5	132	132.5	133	356.1	132	132.7	133	225.4
san1000	0.950	114	115.1	116	424.1	113	113.9	114	207.3	112	112.4	113	352.6
san1000	0.900	40	40.5	41	459.4	42	42.2	43	478.2	39	39.9	40	313.4
san1000	0.800	2	2.0	2	171.2	2	2.0	2	50.8	2	2.0	2	50.6
san1000	0.700	2	2.0	2	194.0	2	2.0	2	51.2	2	2.0	2	50.5
san1000	0.600	2	2.0	2	231.9	2	2.0	2	50.8	2	2.0	2	50.5
p-hat1000-3	0.999	70	70.8	72	294.4	66	66.6	67	348.6	66	66.6	67	364.6
p-hat1000-3	0.950	54	55.2	56	449.1	52	53.6	54	360.9	52	52.7	53	230.0
p-hat1000-3	0.900	38	39.7	41	352.6	38	38.8	39	218.9	36	36.8	37	228.6
p-hat1000-3	0.800	16	16.8	18	522.8	16	16.0	16	92.6	12	12.6	13	287.4
p-hat1500-1	0.999	331	335.7	338	866.7	303	305.4	308	1142.1	306	307.6	310	1039.2
p-hat1500-1	0.950	322	325.7	330	851.7	298	299.6	303	1114.3	301	301.3	302	1123.5
p-hat1500-1	0.900	299	304.5	308	860.4	273	275.6	278	1092.3	270	272.6	275	1261.7
p-hat1500-1	0.800	235	237.8	240	628.4	214	215.4	217	1060.1	206	207.2	209	1128.7
p-hat1500-1	0.700	195	198.8	201	682.8	181	183.0	185	1194.1	166	167.2	169	1035.0
p-hat1500-1	0.600	140	143.2	145	772.0	128	129.6	131	1088.9	121	121.4	122	1012.7
p-hat1500-1	0.500	101	104.5	106	688.0	96	97.4	98	719.4	81	82.0	83	797.8
p-hat1500-1	0.400	73	74.9	77	1220.7	69	69.7	71	730.1	46	46.6	47	626.3
p-hat1500-1	0.300	37	38.9	42	1193.5	31	32.5	34	807.2	16	16.7	17	433.3
p-hat1500-2	0.999	220	223.3	225	806.0	194	196.9	199	770.5	195	197.1	199	930.5
p-hat1500-2	0.950	206	208.2	210	891.2	185	186.3	187	1015.1	184	185.0	186	967.4
p-hat1500-2	0.900	181	184.5	187	607.5	167	168.2	170	921.8	158	159.7	161	955.9
p-hat1500-2	0.800	137	139.0	141	828.3	122	125.2	128	1084.1	109	110.6	111	968.1
p-hat1500-2	0.700	102	107.5	110	1256.5	97	98.1	100	918.7	67	68.2	69	772.0
p-hat1500-2	0.600	62	63.7	67	1446.2	51	51.1	52	782.7	32	32.8	33	473.8
p-hat1500-3	0.999	94	95.2	96	840.5	88	89.5	90	522.5	88	89.5	90	553.9
p-hat1500-3	0.950	71	72.2	73	638.9	70	70.8	71	486.1	69	69.9	70	357.3
p-hat1500-3	0.900	50	51.4	52	1140.5	50	50.1	51	366.0	47	47.6	48	532.2
p-hat1500-3	0.800	18	19.4	21	1302.2	18	18.5	19	275.4	14	14.2	15	642.4
C2000-5	0.999	221	222.0	223	1213.0	218	218.8	220	505.2	218	218.8	220	656.3
C2000-5	0.950	198	199.4	201	999.0	198	198.5	199	829.1	198	198.4	199	865.7
C2000-5	0.900	156	157.6	159	1447.1	158	158.9	160	637.3	157	158.6	160	741.9
C2000-5	0.800	98	98.3	100	883.6	99	100.2	101	849.0	100	100.0	100	586.8
C2000-5	0.700	53	54.7	55	487.3	56	56.4	57	530.1	55	55.0	55	460.0
C2000-5	0.600	21	21.8	22	1065.3	22	22.9	23	387.8	22	22.0	22	211.6

References

Abello, J., Resende, M., & Sudarsky, S. (2002). Massive quasi-clique detection. In J. Abello & J. Vitter (Eds.), *Proceedings of the 5th Latin American Symposium on the Theory of Informatics. Lecture Notes in Computer Science* (Vol. 2286, pp. 598–612). Berlin: Springer.

Agra, A., Dahl, G., Haufmann, T. A., & Pinheiro, S. J. (2017). The *k*-regular induced subgraph problem. *Discrete Applied Mathematics, 222*, 14–30.

Andrade, C. E., Toso, R. F., Gonçalves, J. F., & Resende, M. G. C. (2021). The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research, 289*, 17–30.

Basu, S., Sengupta, D., Maulik, U., & Bandyopadhyay, S. (2014). A strong Nash stability based approach to minimum quasi clique partitioning. In *2014 Sixth International Conference on Communication Systems and Networks, Bangalore* (pp. 1–6). IEEE.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing, 6*, 154–160.

Bomze, I. M., Budinich, M., Pardalos, P. M., & Pelillo, M. (1999). The maximum clique problem. In P. M. Pardalos, D.-Z. Du, & R. L. Graham (Eds.), *Handbook of combinatorial optimization* (pp. 1–74). Boston: Springer.

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 in Operational Research, 22*, 823–839.

Brandão, J. S., Noronha, T. F., Resende, M. G. C., & Ribeiro, C. C. (2016). A biased random-key genetic algorithm for scheduling heterogeneous multi-round systems. *International Transactions in Operational Research, 24*, 1061–1077.

Brandão, J. S., Noronha, T. F., & Ribeiro, C. C. (2016). A biased random-key genetic algorithm to maximize the number of accepted lightpaths in WDM optical networks. *Journal of Global Optimization, 65*, 813–835.

Campello, R. J., Kröger, P., Sander, J., & Zimek, A. (2020). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10*(2), 1343.

Carrabs, F. (2021). A biased random-key genetic algorithm for the set orienteering problem. *European Journal of Operational Research, 292*, 830–854.

DIMACS. (2021). Implementation Challenges. Online reference at http://dimacs.rutgers.edu/Challenges/. Last visited on October 17, 2022.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: a Guide to the Theory of NP-completeness*. San Francisco: Freeman.

Glaria, F., Hernández, C., Ladra, S., Navarro, G., & Salinas, L. (2021). Compact structure for sparse undirected graphs based on a clique graph partition. *Information Sciences, 544*, 485–499.

Gonçalves, J. F., & Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics, 17*(5), 487–525.

Gonçalves, J. F., & Resende, M. G. C. (2015). A biased random-key genetic algorithm for the unequal area facility layout problem. *European Journal of Operational Research, 246*, 86–107.

Hu, H., Yan, X., Huang, Y., Han, J., & Zhou, X. J. (2005). Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics, 21*, 213–221.

Kaminski, J., Schober, M., Albaladejo, R., Zastupailo, O., & Hidalgo, C. (2018). *Moviegalaxies - Social networks in movies*. Harvard Dataverse. Online reference at, 2022. https://doi.org/10.7910/DVN/T4HBA3, last visited on October 17.

Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1*, 231–240.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 50–60.

Marinelli, F., Pizzuti, A., & Rossi, F. (2021). LP-based dual bounds for the maximum quasi-clique problem. *Discrete Applied Mathematics, 296*, 118–140.

Marzo, R. G., Melo, R. A., Ribeiro, C. C., & Santos, M. C. (2022). New formulations and branch-and-cut procedures for the longest induced path problem. *Computers & Operations Research, 139*, 105627.

Matsypura, D., Veremyev, A., Prokopyev, O. A., & Pasiliao, E. L. (2019). On exact solution approaches for the longest induced path problem. *European Journal of Operational Research, 278*, 546–562.

Melo, R. A., Queiroz, M. F., & Ribeiro, C. C. (2021). Compact formulations and an iterated local search-based matheuristic for the minimum weighted feedback vertex set problem. *European Journal of Operational Research, 289*, 75–92.

Melo, R. A., & Ribeiro, C. C. (2022). Maximum weighted induced forests and trees: new formulations and a computational comparative review. *International Transactions in Operational Research, 29*, 2263–2287.

Melo, R. A., & Ribeiro, C. C. (2023). MIP formulations for induced graph optimization problems: a tutorial. *International Transactions in Operational Research, 30*, 3159–3200.

Melo, R. A., Ribeiro, C. C., & Riveaux, J. A. (2022). The minimum quasi-clique partitioning problem: Complexity, formulations, and a computational study. *Information Sciences, 612*, 655–674.

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., Plastino, A., & Ribeiro, C. C. (2013). Construction heuristics for the maximum cardinality quasi-clique problem. In *Abstracts of the Tenth Metaheutistics International Conference*, Singapore (p. 84).

Pattillo, J., Veremyev, A., Butenko, S., & Boginski, V. (2013). On the maximum quasi-clique problem. *Discrete Applied Mathematics, 161*, 244–257.

Peng, B., Wu, L., Wang, Y., & Wu, Q. (2021). Solving maximum quasi-clique problem by a hybrid artificial bee colony approach. *Information Sciences, 578*, 214–235.

Pinto, B. Q., Ribeiro, C. C., Riveaux, J. A., & Rosseti, I. (2021). A BRKGA-based matheuristic for the maximum quasi-clique problem with an exact local search strategy. *RAIRO: Recherche Opérationnelle, 55*, 741–763.

Pinto, B. Q., Ribeiro, C. C., Rosseti, I., & Noronha, T. F. (2020). A biased random-key genetic algorithm for routing and wavelength assignment under a sliding scheduled traffic model. *Journal of Global Optimization, 77*, 949–973.

Pinto, B. Q., Ribeiro, C. C., Rosseti, I., & Plastino, A. (2018). A biased random-key genetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research, 271*, 849–865.

Resende, M. G. C., & Ribeiro, C. C. (2016). Biased-random key genetic algorithms: An advanced tutorial. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference - GECCO'16 Companion Volume* (pp. 483–514). Association for Computing Machinery.

Ribeiro, C. C., & Riveaux, J. A. (2019). An exact algorithm for the maximum quasi-clique problem. *International Transactions in Operational Research, 26*, 2199–2229.

Rossi, R. A., & Ahmed, N. K. (2015). The Network Data Repository with Interactive Graph Analytics and Visualization. Online reference at http://networkrepository.com. Last access on May 5, 2023.

Rossi, R., & Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. *Proceedings of the AAAI Conference on Artificial Intelligence29*(1).

Sanei-Mehri, S.-V., Das, A., Hashemi, H., & Tirthapura, S. (2021). Mining largest maximal quasi-cliques. *ACM Transactions on Knowledge Discovery from Data, 15*, 1–21.

Seo, J. H., & Kim, M. H. (2021). Finding influential communities in networks with multiple influence types. *Information Sciences, 548*, 254–274.

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika, 52*(3/4), 591–611.

Spears, W., & De Jong, K. A. (1991). On the virtues of parameterized uniform crossover. In R. Belew & L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 230–236). San Mateo: Morgan Kaufman.

Spirin, V., & Mirny, L. A. (2003). Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences, 100*, 12123–12128.

Toso, R. F., & Resende, M. G. (2015). A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software, 30*, 81–93.

Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., & Tsiarli, M. (2013). Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago* (pp. 104–112).

Veremyev, A., Prokopyev, O. A., Butenko, S., & Pasiliao, E. L. (2016). Exact MIP-based approaches for finding maximum quasi-clique and dense subgraphs. *Computational Optimization and Applications, 64*, 177–214.

Verteletskyi, V., Yen, T.-C., & Izmaylov, A. F. (2020). Measurement optimization in the variational quantum eigensolver using a minimum clique cover. *The Journal of Chemical Physics, 152*, 124114.

Wu, Q., & Hao, J.-K. (2015). A review on algorithms for maximum clique problems. *European Journal of Operational Research, 242*, 693–709.

Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports, 6*, 30750.

Zhao, X., Liang, J., & Wang, J. (2021). A community detection algorithm based on graph compression for large-scale social networks. *Information Sciences, 551*, 358–372.

Zhou, Q., Benlic, U., & Wu, Q. (2020). An opposition-based memetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research, 286*, 63–83.