CP-Lib: Benchmark Instances of the Clique Partitioning Problem

Michael M. Sørensen[*] Adam N. Letchford[†]

To appear in *Mathematical Programming Computation*

Abstract

The *Clique Partitioning Problem* is a fundamental and much-studied $\mathcal{NP}$-hard combinatorial optimisation problem, with many applications. Several families of benchmark instances have been created in the past, but they are scattered across the literature and hard to find. To remedy this situation, we present CP-Lib, an online resource that contains most of the known instances, plus some challenging new ones.

Key Words: clique partitioning problem; combinatorial optimisation

1 Introduction

Given a complete undirected graph, with rational weights on the edges, the *clique partitioning problem* (CPP) calls for a partition of the vertex set into subsets, such that the sum of the weights of the edges that have both end-vertices in the same subset is maximised. The CPP was originally formulated with statistical clustering in mind [27, 38], but has since been applied to several other problems, including task distribution in distributed systems [1], clustering in group technology [43, 54], microarray data analysis [34], flight-to-gate assignment in airports [22], detecting communities in social networks [26], cluster editing in computational biology [4], and detecting bad links in Wikipedia [8].

The CPP is NP-hard in the strong sense [53]. A variety of solution approaches have been devised for it, including both exact algorithms (e.g., [23, 27, 28, 43, 44, 47, 49, 51]) and heuristics (e.g., [7, 10, 16, 19, 23, 37, 45, 55]). In order to compare the performance of exact and/or heuristic approaches, it is desirable to have a collection of benchmark instances. A large number of benchmark instances have been created in the past [4, 7, 10, 16, 19, 23, 25,

[*]Department of Economics and Business Economics, Aarhus University, Fuglesangs Allé 4, DK-8210 Aarhus V, Denmark. E-mail: mim@econ.au.dk.

[†]Department of Management Science, Lancaster University, Lancaster LA1 4YX, United Kingdom. E-mail: A.N.Letchford@lancaster.ac.uk.

27, 36, 37, 43, 45, 54, 55]. Unfortunately, most of them have not been made available online. To remedy this situation, we present `CP-Lib`, an online resource that contains most of the known instances, plus some challenging new ones.

The paper has the following structure. Section 2 gives an introduction to the library, and in Section 3, we give an overview of the benchmark CPP instances that have already been proposed in the literature. Finally, in Section 4, we present some new instances.

Throughout the paper, the number of vertices will be denoted by n and the weight of an edge $\{i, j\}$ will be denoted by w_{ij}.

2 The Web Repository

This section introduces `CP-Lib`. Subsection 2.1 describes the file format, and Subsection 2.2 presents our choice of which instances to include. Subsection 2.3 explains how we were able to compute proven optimal values for many of the instances, and Subsection 2.4 discusses how to classify the instances according to their difficulty.

`CP-Lib` can be accessed at GitHub: github.com/MMSorensen/CP-Lib. It contains a folder with the files for each family of instances mentioned in Sections 3 and 4 below.

2.1 File format

The data is available in plain text files containing only integers. The first line contains the number of vertices n, and the following lines contain all edge weights, which are separated by space or new-line characters. The data files have the format shown in Fig. 1.

$$\begin{array}{llll}
n & & & \\
w_{1,2} & w_{1,3} & \cdots & w_{1,n} \\
w_{2,3} & \cdots & w_{2,n} & \\
 & \vdots & & \\
w_{n-1,n} & & &
\end{array}$$

Figure 1: The file format.

2.2 Choice of instances

We have included most of the instances that we were able to track down in the literature. However, there are some instances in the literature that are associated with practical applications, such as those mentioned in [22], for which the data sets are not available due to confidentiality and other reasons.

We have also excluded some instances on the basis of their size. Instances with $n < 30$ were not included, since they were all trivial to solve. Some random instances with $n > 2500$, due to Lu *et al.* [37], were also excluded (see Subsection 3.4). These latter instances occupy an excessive amount of storage space, and we were not able to obtain useful lower or upper bounds for them in reasonable computing times.

2.3 Verification of optimal values

Optimal values for many of the benchmark instances are reported in [23, 27, 28, 43, 44, 47, 49, 51]. To verify these values, we used our own CPP solver. Our solver is of branch-and-cut type, and it is based on the standard 0-1 LP formulation of the CPP [27, 38]. This formulation takes the form:

$$\begin{array}{rll} \max & \sum_{1\le i<j\le n} w_{ij}x_{ij} & \\ \text{s.t.} & x_{ik}+x_{jk}-x_{ij}\le 1 & \big(1\le i<j\le n,\ k\ne i,j\big) \\ & x_{ij}\in\{0,1\} & \big(1\le i<j\le n\big). \end{array} \tag{1}$$

Here, the binary variable x_{ij} takes the value 1 if and only if vertices i and j lie in the same clique.

The inequalities (1) are called *transitivity* inequalities. We employ these inequalities in our branch-and-cut algorithm, together with two other families of (facet-defining) inequalities: the *odd wheel* inequalities [17] and the *2-partition* inequalities [27]. The odd wheel inequalities are separated exactly using the algorithm described in [21]. For the 2-partition inequalities, we use an improved version of the separation heuristics described in [27, 43]. A detailed description of the improved version can be found in [50].

We also used a metaheuristic to obtain *lower bounds* on the optimal values. This metaheuristic is similar to the one by Zhou *et al.* [55]. For each instance, the lower bound from the metaheuristic was fed into the branch-and-cut solver. In most cases, this enabled us to solve the instance more quickly.

2.4 Classification of instances

In order to give an indication of the computational difficulty of each instance, we classify instances into three groups: *easy*, *non-trivial*, and *challenging*. This is done in the following way.

We say that an instance is *easy* if the standard LP relaxation (containing only transitivity and non-negativity inequalities) has at least one integer optimal solution. Such instances are often solved at the root node by our algorithm, either because the LP solution is integral, or because the lower bound from our heuristic is equal to the upper bound from the LP. (Note that our definition of "easy" makes no mention of computing time. In practice, some of the larger "easy" instances can actually take up to an hour to solve, due to the sheer size of the LPs involved.)

Non-trivial instances are those that cannot be classified as easy, but can be solved by our algorithm within one hour of computation. This involves the use of the other kinds of inequalities as cutting planes, and branching may also be performed.

Finally, *challenging* instances are those that cannot be solved by our branch-and-cut algorithm within one hour of computation.

3 Instances in the Literature

This section gives an overview of the existing benchmark instances. Subsections 3.1, 3.2 and 3.3 cover instances coming from aggregation of binary relations, machine cell formation and cluster editing, respectively. Subsection 3.4 concerns instances with random weights, and Subsection 3.5 concerns some instances that were designed to be challenging for local search heuristics.

In all of the tables in this section, the column headed "Value" gives the best known lower bound at the time of writing for the instance in question. We mark the value with an asterisk (*) when our branch-and-cut algorithm was able to prove optimality of the given value (within a reasonable amount of time). In the column "Source", we give a reference to the source of the instance data. Sometimes, we also give a reference to the paper that was the first to state the best known value. Finally, the column "Difficulty" classifies instances, as explained in Subsection 2.4.

3.1 Aggregation of binary relations

Grötschel & Wakabayashi [27,53] consider the CPP in the context of a problem in qualitative data analysis known as "aggregation of binary relations into an equivalence relation" (ABR for short). Other researchers have subsequently used the CPP to model and solve instances of the same problem, e.g. [22,23,36].

An instance of this problem is given by n "objects", each with m qualitative "attributes". The objects and the values of their attributes can be represented in a matrix $D = (d_{ik})$, where each entry d_{ik} is the value of attribute k for object i. For each pair $\{i, j\}$ of objects and each attribute k, we define the binary constant

$$r_{ij}^k = \begin{cases} 1, & \text{if attribute } k \text{ has the same value for objects } i \text{ and } j \\ 0, & \text{otherwise.} \end{cases}$$

Edge weights w_{ij} are then defined as the "similarities" of objects i and j with respect to the m attributes: $w_{ij} = 2\sum_{k=1}^{m} r_{ij}^k - m$ (see [27] for details).

Sometimes, the edge weights must be adjusted to take missing values or quantitative attributes into consideration. In the case of missing values, we follow the approach in [36], which is equivalent to the one in [27]. For any missing value d_{ik} or d_{jk}, let $r_{ij}^k = 0$, and let I_{ij} be the number of missing values for objects i or j for all attributes (missing values for both objects are counted only once). Edge weights are then modified as

$$w_{ij} = 2\sum_{k=1}^{m} r_{ij}^k - (m - I_{ij}). \tag{2}$$

In the case of no missing values, all weights in (2) equal the standard edge weights as mentioned above. Otherwise, only attributes with no missing values are considered in the calculation.

When the data contains quantitative attributes, the binary relations can be modified as suggested in [27]:

$$r_{ij}^k = 1 \quad \Leftrightarrow \quad \frac{|d_{ik} - d_{jk}|}{\max(d_{ik}, d_{jk}, 1)} \leq \alpha, \tag{3}$$

where α is a threshold to be specified.

3.1.1 Instances from Grötschel and Wakabayashi

Grötschel & Wakabayashi [27, 53] presented six new instances from real-life applications and seven other instances previously considered in the literature. Table 1 lists these instances. Although [27] and [53] are listed as the only sources, some of the instances have different origins, such as [38]. We have used the data tables provided in the sources mentioned to obtain the input to these instances.

We note that the value of the instance "UNO_3a" does not correspond to the one stated in [27]. We have double-checked that we have obtained the input correctly as it appears in [53]. Furthermore, Martin Grötschel has informed one of us that their original data got lost and may not have been restored properly. We therefore ascribe the difference to this fact.

A few other inconsistencies are mentioned by Dorndorf and Pesch [23]. While they do not consider the instance "UNO_3a", they state that they obtain different objective values for three other instances ("companies", "UNO", and "UNO_1b"), which may be attributed to typos in the data matrices. They also had to adjust the parameter α used in (3) for calculating the edge weights for the instance "micro" (using $\alpha = 0.262$ instead of 0.3 as stated in [27]) to obtain the same objective value as in [27]. We also had to make this adjustment of the parameter, but otherwise our results are consistent with those obtained in [27]. We therefore suspect that there are errors in the data used in [23].

Table 1: Instances considered by Grötschel and Wakabayashi.

Instance	n	Value	Source	Difficulty
cars	33	1501*	[27]	easy
cetacea	36	967*	[27]	easy
companies	137	81802*	[27]	easy
micro	40	966*	[27]	easy
UNO	54	798*	[27]	easy
UNO_1a	158	12197*	[27]	easy
UNO_1b	139	11775*	[27]	easy
UNO_2a	158	72820*	[27]	easy
UNO_2b	145	71818*	[27]	easy
UNO_3a	158	73068*	[53]	easy
UNO_3b	147	72629*	[53]	easy
wildcats	30	1304*	[27]	easy
workers	34	964*	[27]	easy

3.1.2 Further ABR instances

Further ABR instances of the CPP have been considered in the literature (e.g., [10, 36]). Most of these instances are obtained from the UCI machine learning repository [25]. Table 2 presents some of these and a couple of other instances that we found.

The instance "lecturers" is obtained from the data given in [23], but using a different weight function than the one mentioned in the reference. Erwin Pesch has confirmed (via e-mail correspondence) that, according to his memory, we use the same calculation as they did in [23].

We note that the instance "primary-tumor" has an optimal partition with three clusters, one consisting of 337 objects and the other two being singletons. Therefore, this partition is likely of little use in the analysis of the data.

The "soup" instance has not been considered previously in the literature. It originates from a consumer survey with data provided by a colleague of one of the authors of this paper.

The three "soybean" instances have 35 attributes, but the small instances on 47 vertices have 14 attributes that are identical for all the objects. Using all the attributes, we obtain the result shown as "soybean-35", with an optimal partition consisting of a single cluster, which carries no useful information (this instance is considered in [36]). We prefer the instance "soybean-21", which has also been considered in [10]: it uses only the 21 non-identical attributes and has an optimal partition with 3 clusters. These small "soybean" instances consist of a subset of the objects in the "soybean-large" instance, for which we use all 35 attributes in the weight calculations.

Table 2: More ABR instances.

Instance	n	Value	Source	Difficulty
bridges	108	3867*	[25, 36]	non-trivial
Hayes-Roth	160	2800*	[25, 36]	non-trivial
lecturers	797	14317*	[23]	challenging
lung-cancer	32	3472*	[25, 36]	easy
lymphography	148	19174*	[25, 36]	non-trivial
primary-tumor	339	323614*	[25, 36]	easy
soup	209	4625*		non-trivial
soybean-21	47	3041*	[10, 25]	easy
soybean-35	47	14613*	[25, 36]	easy
soybean-large	307	316469*	[25]	easy
sponge	76	25677*	[25, 36]	easy
ta-evaluation	151	1108*	[25, 36]	easy
zoo	101	16948*	[10, 25]	easy

The instance "ta-evaluation" is used in [36] under the name "Teaching Evaluation". In [25], the same data set is named "tae".

3.2 Machine cell formation

The "machine cell formation problem" occurs in the context of Group Technology in production system layout. Several instances of this problem are considered by Oosten *et al.* [43] and Wang *et al.* [54]. All these instances come from the literature, and they are obtained as follows.

An instance of the problem consists of a set of machines and and a set of parts to be processed by the machines together with a 0-1 matrix containing a column for each part and a row for each machine, or vice versa. 1-entries in this matrix indicate that the corresponding part must be processed by the associated machine, and 0-entries indicate the opposite.

An instance of the CPP is obtained by defining a vertex for each part and a vertex for each machine. Edge weights are then obtained in the following way. Edges connecting two part-vertices or two machine-vertices are given weight 0. Each edge connecting a part-vertex and a machine-vertex is given weight 1 if the part must be processed by the machine; otherwise the weight is -1.

Table 3 lists many of the instances obtained from 0-1 matrices in the literature, and the specific sources of the inputs are listed in the table. We note that a few of the instances come from different applications, e.g., the instance "ROG_05" originates from a survey of industrial purchasing behaviour.

The "Wang" instances are introduced in [54] (under the names GT_50_200, GT_100_700, and GT_150_1000) and have subsequently been considered in [24] and [29]. We have obtained the data for these instances from the first author of the last-mentioned reference.

3.3 Cluster editing

"Cluster editing", also known as "correlation clustering", is a technique in data mining [2, 20]. Given a graph $G = (V, E)$ on n vertices, the problem is to determine a set of edge modifications (insertions and deletions), of minimum cardinality, such that the modified graph is a clique partition.

To explain this in detail, we need some more notation. Let E_n denote the set of edges of the complete graph on n vertices, i.e., $E_n = \big\{\{i, j\} : 1 \leq i < j \leq n\big\}$. Also let $P \subseteq E_n$ be the edge set of a clique partition. Then, to convert E into P, we need to insert $|P \setminus E|$ edges from P and delete $|E| - |P \cap E|$ edges from E. The goal is to find the set P that minimises the total number of edge modifications.

In the literature on cluster editing, the weights of the edges in E_n are $+1$ for the edges in $E_n \setminus E$ and -1 for the edges in E. However, since we consider the maximisation version of the CPP here, the edge weights in our instances are reversed such that the weights of the edges are -1 for the edges in $E_n \setminus E$ and $+1$ for the edges in E.

Böcker *et al.* [4] consider several thousand semi-random instances of this problem. These instances turn out to be quite easy, and we have decided not to include them here. They are all available at: `bio.informatik.uni-jena.de/software/peace/`.

Simanchev *et al.* [49] consider several instances where the edges in E are drawn randomly from E_n and with different densities. We follow their approach and provide similar instances in Table 4. The names of these instances are "ce$n$-d", where n is the number of vertices of the graph and d is the edge density measured as $|E|/|E_n| \cdot 100$ %. We include only instances with density between 20% and 60%, because we found that other instances are extremely easy for both exact and heuristic solution methods.

3.4 Random instances

We now turn our attention to random instances. Most of the random instances considered in the literature are obtained by using edge weights that are random uniformly distributed integers in an interval from l to u. For ease of notation, we will denote these weights as RUI $[\,l, u\,]$.

Charon & Hudry [16] presented seven random instances. Brusco & Köhn [10] added six "missing" random instances to those of Charon & Hudry. The web links to data instances given in these papers appear to be insecure or not valid any longer. Instead, we have obtained the data

Table 3: Instances from machine cell formation in Group Technology.

Instance	n	Value	Source	Difficulty
BOC_1	46	58*	[5]	non-trivial
BOC_2	46	61*	[5]	non-trivial
BOC_3	46	60*	[5]	non-trivial
BOC_4	46	50*	[5]	non-trivial
BOC_5	46	72*	[5]	non-trivial
BOC_6	46	76*	[5]	easy
BOC_7	46	78*	[5]	easy
BOC_8	46	61*	[5]	non-trivial
BOC_9	46	89*	[5]	easy
BOC_10	46	70*	[5]	non-trivial
BOE_91	55	80*	[6]	non-trivial
BUR_69	55	98*	[46]	easy
BUR_73	126	126	[33]	challenging
BUR_75	59	67*	[32]	non-trivial
BUR_91	59	72*	[11]	non-trivial
CAN_97	68	157*	[12]	non-trivial
CHA_86	55	102*	[14]	easy
CHA_87	140	347*	[15]	easy
GRO_80	43	53*	[35]	non-trivial
IRA_95	31	38*	[46]	non-trivial
KAT_97	108	175	[30]	challenging
KIN_80	38	41*	[31]	easy
LEE_97	70	115*	[46]	easy
MAS_97	35	41*	[39]	non-trivial
MCC_72	40	43*	[40]	non-trivial
MIL_91	60	46*	[41]	non-trivial
NAI_96a	64	117*	[42]	easy
NAI_96b	64	93*	[42]	easy
NAI_96c	64	91*	[42]	non-trivial
NAI_96d	64	74*	[42]	non-trivial
ROG_05	65	60*	[46]	non-trivial
SEI_88	33	54*	[48]	non-trivial
SUL_91	31	46*	[52]	non-trivial
Wang250	250	419	[29, 54]	challenging
Wang800	800	1177	[29, 54]	challenging
Wang1150	1150	3236	[29, 54]	challenging

Table 4: Random cluster editing instances.

Instance	n	Value	Difficulty
ce50-20	50	58*	non-trivial
ce50-30	50	79*	non-trivial
ce50-40	50	105*	challenging
ce50-50	50	163*	non-trivial
ce50-60	50	257*	non-trivial
ce60-20	60	73*	non-trivial
ce60-30	60	100	challenging
ce60-40	60	151*	challenging
ce60-50	60	200	challenging
ce60-60	60	373*	non-trivial
ce70-20	70	93*	non-trivial
ce70-30	70	128	challenging
ce70-40	70	177	challenging
ce70-50	70	266	challenging
ce70-60	70	491*	non-trivial
ce80-20	80	107*	challenging
ce80-30	80	157	challenging
ce80-40	80	227	challenging
ce80-50	80	325	challenging
ce80-60	80	657*	non-trivial

sets from Zhou *et al.* [55] and have transformed them into maximisation instances of the CPP.

These instances are shown in Table 5. The instances "rand$n$-i" on n vertices have RUI $[-i, i]$ weights. The "regnier300-50" instance has been obtained by considering 50 (random) bipartitions and setting edge weights by counting the number of clusters in which each pair of vertices is or is not in the same cluster. The instance "sym300-50" has been obtained by creating 50 (random) symmetric relations among the vertices and computing edge weights by counting the numbers of related and unrelated vertex pairs. Finally, the instance "zahn300" has random edge weights that are -1 or 1.

Brimberg *et al.* [7] also use the instances in Table 5 and some random instances of their own. The latter instances are not available, but they are generated in the same way as the "p1000", "p1500" and "p2000" instances mentioned below.

Palubeckis *et al.* [45] consider 35 random instances on 500 to 2000 vertices. Table 6 gives an overview. The instances "p500-i-c" on 500 vertices have RUI $[-i, i]$ weights, and "c" is just an identifier. The larger instances "pn-c" on 1000 and more vertices have RUI $[-100, 100]$ edge weights. These

Table 5: Charon & Hudry and Brusco & Köhn random instances.

Instance	n	Value	Source	Difficulty
rand100-5	100	1407	[10]	challenging
rand100-100	100	24296	[16]	challenging
rand200-5	200	4079	[10]	challenging
rand200-100	200	74924	[10]	challenging
rand300-5	300	7732	[16]	challenging
rand300-100	300	152709	[16]	challenging
rand400-5	400	12133	[10]	challenging
rand400-100	400	222757	[10]	challenging
rand500-5	500	17127	[10]	challenging
rand500-100	500	309125	[16]	challenging
regnier300-50	300	32164	[16]	challenging
sym300-50	300	17592	[16]	challenging
zahn300	300	2504	[16]	challenging

instances have subsequently been considered by Jovanovic *et al.* [29], Lu *et al.* [37], and Zhou *et al.* [55]. Frequently some of these references obtain better partition values, in which cases we state their best values and include a reference to them in the "Source" column. As above, we have obtained these instances from Zhou *et al.* [55].

Zhou *et al.* [55] provide 15 additional instances on 500 to 800 vertices as shown in Table 7. The "gauss500-100-c" instances have random edge weights following a Gaussian distribution $\mathcal{N}(0, 5^2)$, and the "unif$n$-100-c" instances have RUI$\,[-100, 100\,]$ weights. As above, these instances have also been considered by Lu *et al.* [37], and when they obtain better partition values, we state their best values and include a reference to them in the "Source" column.

Lu *et al.* [37] consider 31 large random instances on 2500 to 7000 vertices. We have chosen to include only the instances on 2500 vertices in the library, as shown in Table 8. These instances differ from other random instances in the sense that they are obtained by recasting ten instances of UBQP from the OR-Lib [3]. These instances have also been considered in [29] and we give a reference in the table to this source when it states the best partition value. The remaining larger instances, which we do not consider here, are generated with RUI$\,[-100, 100\,]$ weights; these instances are available from the web-link provided in [37].

Du *et al.* [24] and Jovanovic *et al.* [29] created another set of random instances, with 25 to 100 vertices. According to [24], 80 % of the edges are generated with RUI$\,[0, 100\,]$ weights, and the remaining 20 % of the weights are set to a large negative value (namely, -9999). In line with our policy

Table 6: Palubeckis *et al.* random instances.

Instance	n	Value	Source	Difficulty
p500-5-1	500	17691	[45]	challenging
p500-5-2	500	17169	[45]	challenging
p500-5-3	500	16816	[45, 55]	challenging
p500-5-4	500	16808	[45]	challenging
p500-5-5	500	16957	[45]	challenging
p500-5-6	500	16615	[45]	challenging
p500-5-7	500	16649	[45]	challenging
p500-5-8	500	16756	[45]	challenging
p500-5-9	500	16629	[45]	challenging
p500-5-10	500	17360	[45]	challenging
p500-100-1	500	308896	[45]	challenging
p500-100-2	500	310241	[45, 55]	challenging
p500-100-3	500	310477	[45]	challenging
p500-100-4	500	309567	[45]	challenging
p500-100-5	500	309135	[45]	challenging
p500-100-6	500	310280	[45]	challenging
p500-100-7	500	310063	[45]	challenging
p500-100-8	500	303148	[45]	challenging
p500-100-9	500	305305	[45]	challenging
p500-100-10	500	314864	[45]	challenging
p1000-1	1000	885281	[29, 45]	challenging
p1000-2	1000	881751	[45, 55]	challenging
p1000-3	1000	866488	[37, 45]	challenging
p1000-4	1000	869374	[45, 55]	challenging
p1000-5	1000	888960	[37, 45]	challenging
p1500-1	1500	1619470	[29, 45]	challenging
p1500-2	1500	1649778	[37, 45]	challenging
p1500-3	1500	1611197	[37, 45]	challenging
p1500-4	1500	1641933	[37, 45]	challenging
p1500-5	1500	1595627	[37, 45]	challenging
p2000-1	2000	2508005	[29, 45]	challenging
p2000-2	2000	2495730	[29, 45]	challenging
p2000-3	2000	2544728	[29, 45]	challenging
p2000-4	2000	2528721	[29, 45]	challenging
p2000-5	2000	2514009	[29, 45]	challenging

Table 7: Zhou *et al.* random instances.

Instance	n	Value	Source	Difficulty
gauss500-100-1	500	265070	[55]	challenging
gauss500-100-2	500	269076	[55]	challenging
gauss500-100-3	500	257700	[55]	challenging
gauss500-100-4	500	267683	[55]	challenging
gauss500-100-5	500	271567	[55]	challenging
unif700-100-1	700	515016	[55]	challenging
unif700-100-2	700	519441	[55]	challenging
unif700-100-3	700	512351	[55]	challenging
unif700-100-4	700	513582	[55]	challenging
unif700-100-5	700	510585	[37, 55]	challenging
unif800-100-1	800	639675	[55]	challenging
unif800-100-2	800	630704	[55]	challenging
unif800-100-3	800	629375	[37, 55]	challenging
unif800-100-4	800	624728	[55]	challenging
unif800-100-5	800	625905	[55]	challenging

described in Subsection 2.2, we include in our library only the instances with at least 30 vertices. These instances, described in Table 9, were kindly provided to us by Raka Jovanovic. For the smaller instances, the optimal values have been verified by our CPP solver. For the remaining instances, we give the best known lower bound, along with the reference where that lower bound was first reported.

3.5 Artificial instances

Finally, we mention that De Amorim *et al.* [19] describe a method for creating so-called "special instances" that are designed to trick traditional local search methods into arriving at a local optimum that is not a global optimum. In each instance, they consider a weighted complete graph on $n = 2n'$ vertices, where V^1, V^2 is a partition of the vertex set such that $V^1 = \{1, \ldots, n'\}$ and $V^2 = \{n'+1, \ldots, 2n'\}$. Edge weights are then assigned in the following way:

- Weights equal to $\alpha > 2$ are assigned to the cycle $\{1,2\}, \{2,3\}, \ldots, \{n'-1, n'\}, \{n', 1\}$ of length n' on vertices V^1.
- All other edges with end-vertices in V^1 have weight equal to 2.
- The edges $\{i, n'+i\}$ with $i \in V^1$ and $n'+i \in V^2$, for $i = 1, \ldots, n'$, have weight equal to $\alpha + 1$.
- All other edges have weight equal to $\beta < -\big((n')^2 + 2n'(\alpha - 1)\big)$.

Table 8: Lu *et al.* random instances.

Instance	n	Value	Source	Difficulty
b2500-1	2500	1063621	[29, 37]	challenging
b2500-2	2500	1064144	[29, 37]	challenging
b2500-3	2500	1082946	[29, 37]	challenging
b2500-4	2500	1066239	[29, 37]	challenging
b2500-5	2500	1066387	[37]	challenging
b2500-6	2500	1066978	[29, 37]	challenging
b2500-7	2500	1068377	[29, 37]	challenging
b2500-8	2500	1070060	[29, 37]	challenging
b2500-9	2500	1071272	[37]	challenging
b2500-10	2500	1066770	[29, 37]	challenging

Note that the above description of edge weights applies to the maximisation version of the CPP. The optimal partitions of these graphs consist of $n' + 1$ clusters: vertex set V^1 and singletons $\{n' + i\}$, for each $n' + i \in V^2$, and their values are $n'(\alpha + n' - 3)$.

De Amorim *et al.* consider instances with n up to 100 (i.e. $n' \leq 50$) with $\alpha = 10$ and $\beta = -\big((n')^2 + 2n'\alpha\big)$. Brusco & Köhn [10] consider instances with n between 100 and 300 vertices and the same values of α and β.

In Table 10 we present instances with n between 50 and 300 (in increments of 50) with $\alpha \in \{3, 10, 20\}$ and $\beta = -\big((n')^2 + 2n'\alpha\big)$. We name these instances using the formula "am-n'-α", so that the parameters are contained in the names.

4 New Instances

In this last section, we introduce two additional sets of CPP instances. Subsection 4.1 introduces some instances derived from a consideration of the "equicut" problem, and Subsection 4.2 presents a procedure for generating new instances based on correlations between random variables.

4.1 Instances from the equicut problem

The "equicut" or "equipartition" problem is similar to the CPP, but there is the added constraint that there must be exactly two clusters, each of cardinality $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$ (see, e.g., [9, 18]).

It is possible to convert any equicut instance into a CPP instance by simply removing the above-mentioned constraint. Of course, for the resulting CPP instance to be non-trivial, it is necessary that there be a mixture of positive and negative edge weights. It turns out that some of the equi-

Table 9: Du *et al.* and Jovanovic *et al.* random instances

Instance	n	Value	Source	Difficulty
CPn35-1	35	7837*	[24]	Non-trivial
CPn35-2	35	7215*	[24]	Non-trivial
CPn35-3	35	7633*	[24]	Non-trivial
CPn35-4	35	7652*	[24]	Non-trivial
CPn45-1	45	11545*	[24]	Challenging
CPn45-2	45	12345*	[29]	Non-trivial
CPn45-3	45	11880*	[29]	Challenging
CPn45-4	45	10506*	[24]	Challenging
CPn50-1	50	13562	[29]	Challenging
CPn50-2	50	14080	[24]	Challenging
CPn50-3	50	13172	[29]	Challenging
CPn50-4	50	13728	[24]	Challenging
CPn65-1	65	20028	[29]	Challenging
CPn65-2	65	20753	[29]	Challenging
CPn65-3	65	20463	[29]	Challenging
CPn65-4	65	20000	[29]	Challenging
CPn100-1	100	37188	[29]	Challenging
CPn100-2	100	37460	[29]	Challenging
CPn100-3	100	39766	[29]	Challenging
CPn100-4	100	38192	[29]	Challenging

cut instances mentioned in [9] are of this type: the so-called "negative" instances.

The "negative" instances are created using a "density" parameter. In detail, given a density of $k\%$, the edge weights are set to 0 with probability $(100-k)\%$, and to a non-zero integer with probability $k\%$. In the latter case, the weight is selected uniformly at random from $\{-9,-8,\ldots,-1\} \cup \{1,2,\ldots 9\}$.

Table 11 gives some information about the instances on 50 or more vertices. In this table, we include an extra column, "Density", showing the percentage of non-zero edge weights. Note that these instances have not been considered previously as instances of the CPP. For this reason, the values stated are those obtained by our own algorithms.

4.2 Correlation instances

Finally, we found that one can generate quite challenging CPP instances by computing correlations between n independent random variables, and then setting each edge weight to the corresponding correlation coefficient. In more detail, we do the following:

Table 10: Artificial instances.

Instance	n	Value	Difficulty
am-25-3	50	625*	easy
am-25-10	50	800*	easy
am-25-20	50	1050*	easy
am-50-3	100	2500*	easy
am-50-10	100	2850*	easy
am-50-20	100	3350*	easy
am-75-3	150	5625*	easy
am-75-10	150	6150*	easy
am-75-20	150	6900*	easy
am-100-3	200	10000*	easy
am-100-10	200	10700*	easy
am-100-20	200	11700*	easy
am-125-3	250	15625*	easy
am-125-10	250	16500*	easy
am-125-20	250	17750*	easy
am-150-3	300	22500*	easy
am-150-10	300	23550*	easy
am-150-20	300	25050*	easy

1. Let n be the desired number of vertices.

2. Create an n by n square matrix in which each entry is uniformly distributed between 0 and 1.

3. For $1 \leq i < j \leq n$, let c_{ij} be the correlation between the ith and jth columns of the matrix.

4. For $1 \leq i < j \leq n$, set w_{ij} to be $100c_{ij}$ rounded to the nearest integer.

We call instances of this kind "correlation instances".

Following this scheme, we created ten instances of each kind for $n \in \{40, 60, 80\}$. The resulting instances turned out to be "non-trivial" or "challenging". Table 12 presents some information about the instances.

References

[1] H.H. Ali & H. El-Rewini, Task allocation in distributed systems: a split graph model, *J. Combin. Math. Combin. Comput.*, 14, 15–32 (1993)

[2] N. Bansal, A. Blum & S. Chawla, Correlation clustering, *Machine Learning*, 56, 89–113 (2004)

Table 11: Equicut “negative” instances.

Instance	n	Density	Value	Source	Difficulty
neg-c-00	50	100	752*	[9]	challenging
neg-c-10	50	90	649*	[9]	challenging
neg-c-20	50	80	604*	[9]	challenging
neg-c-30	50	70	582*	[9]	non-trivial
neg-c-40	50	60	577*	[9]	non-trivial
neg-c-50	50	50	549*	[9]	non-trivial
neg-c-60	50	40	463*	[9]	non-trivial
neg-c-70	50	30	452*	[9]	non-trivial
neg-c-80	50	20	317*	[9]	non-trivial
neg-s-80	60	20	473*	[9]	non-trivial
neg-tt-80	70	20	592	[9]	challenging

[3] J.E. Beasley, OR-Library: distributing test problems by electronic mail, *J. Oper. Res. Soc.*, 41, 1069–1072 (1990)

[4] S. Böcker, S. Briesemeister & G.W. Klau, Exact algorithms for cluster editing: evaluation and experiments, *Algorithmica*, 60, 316–334 (2011)

[5] F. Boctor, A linear formulation of the machine-part cell formation problem, *Int. J. Prod. Res.*, 29, 343–356 (1991)

[6] W.J. Boe & C.H. Cheng, A close neighbour algorithm for designing cellular manufacturing systems, *Int. J. Prod. Res.*, 29, 2097–2116 (1991)

[7] J. Brimberg, S. Janićijević, N. Mladenović & D. Urošević, Solving the clique partitioning problem as a maximally diverse grouping problem, *Optim. Lett.*, 11, 1123–1135 (2017)

[8] S. Bruckner, F. Hüffner, Ch. Komusiewicz & R. Niedermeier, Evaluation of ILP-based approaches for partitioning into colorful components. In V. Bonifaci *et al.* (eds.) *Experimental Algorithms*, pp. 176–187. Springer, Heidelberg (2013)

[9] L. Brunetta, M. Conforti & G. Rinaldi, A branch-and-cut algorithm for the equicut problem, *Math. Program.*, 77, 243–263 (1997)

[10] M.J. Brusco & H.F. Köhn, Clustering qualitative data based on binary equivalence relations: neighborhood search heuristics for the clique partitioning problem, *Psychometrika*, 74, 685–703 (2009)

[11] J.L. Burbidge, Production flow analysis for planning group technology. *J. Oper. Manag.*, 10, 5–27 (1991)

Table 12: Correlation instances.

Instance	n	Value	Difficulty
corr40-1	40	2191*	non-trivial
corr40-2	40	1852*	non-trivial
corr40-3	40	2310*	non-trivial
corr40-4	40	2084*	non-trivial
corr40-5	40	2245*	non-trivial
corr40-6	40	2516*	non-trivial
corr40-7	40	2294*	non-trivial
corr40-8	40	2184*	non-trivial
corr40-9	40	2129*	non-trivial
corr40-10	40	2301*	non-trivial
corr60-1	60	3678*	non-trivial
corr60-2	60	3445*	challenging
corr60-3	60	3595*	non-trivial
corr60-4	60	3565*	non-trivial
corr60-5	60	3313*	non-trivial
corr60-6	60	3295*	non-trivial
corr60-7	60	3506*	non-trivial
corr60-8	60	3540*	non-trivial
corr60-9	60	3372*	non-trivial
corr60-10	60	3570*	non-trivial
corr80-1	80	4724	challenging
corr80-2	80	4667	challenging
corr80-3	80	4993	challenging
corr80-4	80	4504	challenging
corr80-5	80	5090	challenging
corr80-6	80	4465	challenging
corr80-7	80	5088	challenging
corr80-8	80	4757	challenging
corr80-9	80	4430	challenging
corr80-10	80	5071	challenging

[12] M. Cantamessa & A. Turroni, A pragmatic approach to machine and part grouping in cellular manufacturing system design, *Int. J. Prod. Res.*, 35, 1031–1050 (1997)

[13] H.M. Chan & D.A. Milner, Direct clustering algorithm for group formation in cellular manufacturing, *J. Manuf. Syst.*, 1, 65-74 (1982)

[14] M.P. Chandrasekharan & R. Rajagopalan, MODROC: an extension of rank order clustering for group technology, *Int. J. Prod. Res.*, 24, 1221–1233 (1986)

[15] M.P. Chandrasekharan & R. Rajagopalan, ZODIAC — an algorithm for concurrent formation of part-families and machine-cells, *Int. J. Prod. Res.*, 25, 835–850 (1987)

[16] I. Charon & O. Hudry, Noising methods for a clique partitioning problem, *Discr. Appl. Math.*, 15, 754–769 (2006)

[17] S. Chopra & M.R. Rao, The partition problem, *Math. Program.*, 59, 87–115 (1993)

[18] M. Conforti, M.R. Rao & A. Sassano, The equipartition polytope I: formulations, dimension and basic facets, *Math. Program.*, 49, 49–70 (1990)

[19] S.G. De Amorim, L.-P. Barthélemy & C.C. Riberio, Clustering and clique partitioning: simulated annealing and tabu search approaches, *J. Classif.*, 9, 17–41 (1992)

[20] F. Dehne, M.A. Langston, X. Luo, S. Pitre, P. Shaw & Y. Zhang, The cluster editing problem: implementations and experiments. In H.L. Bodlaender & M.A. Langston (eds) *Proc. IWPEC '06*. Springer, Berlin (2006)

[21] M. Deza, M. Grötschel & M. Laurent, Clique-web facets for multicut polytopes, *Math. Oper. Res.*, 17, 981–1000 (1992)

[22] U. Dorndorf, F. Jaehn & E. Pesch, Modeling robust flight-gate scheduling as a clique partitioning problem, *Transp. Sci.*, 42, 292–301 (2008)

[23] U. Dorndorf & E. Pesch, Fast clustering algorithms, *ORSA J. Comput.*, 6, 141–153 (1994)

[24] Y. Du, G. Kochenberger, F. Glover, H. Wang, M. Lewis, W. Xie & T. Tsuyuguchi, Solving clique partitioning problems: a comparison of models and commercial solvers. *Int. J. Inf. Technol. Decis. Mak.*, 21, 59–81 (2022)

[25] D. Dua & C. Graff (2019) UCI Machine Learning Repository [archive.ics.uci.edu/ml/]. Irvine, CA: University of California, School of Information and Computer Science.

[26] S. Fortunato, Community detection in graphs, *Phys. Rep.*, 486, 75–174 (2010)

[27] M. Grötschel & Y. Wakabayashi, A cutting plane algorithm for a clustering problem, *Math. Program.*, 45, 59–96 (1989)

[28] F. Jaehn & E. Pesch, New bounds and constraint propagation techniques for the clique partitioning problem, *Discr. Appl. Math.*, 161, 2025–2037 (2013)

[29] R. Jovanovic, A.P. Sanfilippo & S. Voß, Fixed set search applied to the clique partitioning problem, *Eur. J. Oper. Res.*, 309, 65–81 (2023)

[30] I.A. Kattan, Design and scheduling of hybrid multi-cell flexible manufacturing systems, *Int. J. Prod. Res.*, 35, 1239–1257 (1997)

[31] J.R. King, Machine component group formation in group technology, *Omega*, 8, 193–199 (1980)

[32] J.R. King, Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm, *Int. J. Prod. Res.*, 18, 213–232 (1980)

[33] J.R. King & V. Nakornchai, Machine-component group formation in group technology: review and extension, *Int. J. Prod. Res.*, 20, 117–133 (1982)

[34] G. Kochenberger, F. Glover, B. Alidaee & H. Wang, Clustering of microarray data via clique partitioning, *J. Comb. Optim.*, 10, 77–92 (2005)

[35] K.R. Kumar, A. Kusiak & A. Vannelli, Grouping of parts and components in flexible manufacturing systems, *Eur. J. Oper. Res.*, 24, 387–397 (1986)

[36] L.H.N. Lorena, M.G. Quiles, L.A.N. Lorena, A.C.P.L.F. de Carvalho & J.G. Cespedes, Qualitative data clustering: a new integer linear programming model, *Proc. IJCNN '19*. IEEE, Piscataway, NJ (2019)

[37] Z. Lu, Y. Zhou & J-K. Hao, A hybrid evolutionary algorithm for the clique partitioning problem, *IEEE Trans. Cyber.*, 52, 9391–9403 (2022)

[38] J.F. Marcotorchino, *Aggregation of Similarities in Automatic Classification* (in French). Doctoral Thesis, Université Paris VI (1981)

[39] A. Masnata & L. Settineri, An application of fuzzy clustering to cellular manufacturing, *Int. J. Prod. Res.*, 35, 1077–1094 (1997)

[40] W.T. McCormick, Jr., P.J. Schweitzer & T.W. White, Problem decomposition and data reorganization by a clustering technique, *Oper. Res.*, 20, 993–1009 (1972)

[41] J. Miltenburg & W. Zhang, A comparative evaluation of nine well-known algorithms for solving the cell formation problem in group technology, *J. Oper. Manag.*, 10, 44–72 (1991)

[42] G.J.K. Nair & T.T. Narendran, Grouping index: a new quantitative criterion for goodness of block-diagonal forms in group technology, *Int. J. Prod. Res.*, 34, 2767–2782 (1996)

[43] M. Oosten, J.H.G.C. Rutten & F.C.R. Spieksma, The clique partitioning problem: facets and patching facets, *Networks*, 38, 209–226 (2001)

[44] G. Palubekis, A branch-and-bound approach using polyhedral results for a clustering problem, *INFORMS J. Comput.*, 9, 30–42 (1997)

[45] G. Palubeckis, A. Ostreika & A. Tomkevičius, An iterated tabu search approach for the clique partitioning problem, *Sci. World J.*, article 353101 (2014)

[46] D.F. Rogers & S.S. Kulkarni, Optimal bivariate clustering and a genetic algorithm with an application in cellular manufacturing, *Eur. J. Oper. Res.*, 160, 423–444 (2005)

[47] M. Schader & U. Tüshaus, A subgradient algorithm for classification of qualitative data (in German), *OR Spektrum*, 7, 1–5 (1985)

[48] H. Seifoddini, Machine grouping — expert systems: comparison between single linkage and average linkage clustering techniques in forming machine cells, *Comput. Indust. Eng.*, 15, 210–216 (1988)

[49] R.Y. Simanchev, I.V. Urazova & Y.A. Kochetov, The branch and cut method for the clique partitioning problem, *J. Appl. Ind. Math.*, 13, 539–556 (2019)

[50] M.M. Sørensen, A separation heuristic for 2-partition inequalities for the clique partitioning problem, *Working paper*, Department of Economics and Business Economics, Aarhus University (2020)

[51] N. Sukegawa, Y. Yamamoto & L. Zhang, Lagrangian relaxation and pegging test for the clique partitioning problem, *Adv. Data Anal. Classif.*, 7, 363–391 (2013)

[52] D.R. Sule, Machine capacity planning in group technology, *Int. J. Prod. Res.*, 29, 1909–1922 (1991)

[53] Y. Wakabayashi, *Aggregation of Binary Relations: Algorithmic and Polyhedral Investigations.* PhD Thesis, Augsburg University (1986)

[54] H. Wang, B. Alidaee, F. Glover & G. Kochenberger, Solving group technology problems via clique partitioning, *Int. J. Flex. Manuf. Syst.*, 18, 77–97 (2006)

[55] Y. Zhou, J.-K. Hao & A. Goëffon, A three-phased local search approach for the clique partitioning problem, *J. Combin. Optim.*, 32, 469–491 (2016)