Optim Lett
DOI 10.1007/s11590-014-0754-6

SHORT COMMUNICATION

Redundant constraints in the standard formulation for the clique partitioning problem

Atsushi Miyauchi · Noriyoshi Sukegawa

Received: 27 November 2013 / Accepted: 13 May 2014

Abstract The clique partitioning problem is among the most fundamental graph partitioning problems. Given a complete weighted undirected graph, we find a vertex-disjoint union of cliques so as to maximize the sum of the weights within the cliques. The problem is known to be NP-hard, and exact algorithms and heuristics have been developed so far. In 1989, Grötschel and Wakabayashi (Math Program 45:59–96, 1989) proposed an integer linear programming formulation for the problem. Although this formulation has been employed by many algorithms, it is prohibitive even for relatively small graphs because it contains numerous constraints called transitivity constraints. In this study, we theoretically derive a certain class of redundant constraints in the formulation. Furthermore, we show that the constraints in the derived class are also redundant in the linear programming relaxation of the formulation. By using our results, the number of constraints treated in exact algorithms and heuristics may be reduced. We confirmed that more than half of the transitivity constraints are redundant for some instances, and that computation times to find an optimal solution are shortened for most instances.

Keywords Graph partitioning · Mathematical programming · Integer linear programming · Transitivity constraints

1 Introduction

The clique partitioning problem (CPP for short) is among the most fundamental graph partitioning problems. We consider a complete weighted undirected graph

A. Miyauchi (✉) · N. Sukegawa
Graduate School of Decision Science and Technology, Tokyo Institute of Technology,
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
e-mail: miyauchi.a.aa@m.titech.ac.jp

Published online: 25 May 2014

$G = (V, E, c)$ consisting of $n = |V|$ vertices, $n(n-1)/2$ edges, and a weight function $c : E \to \mathbb{R}$. Note that the range of c can include both positive and negative values. For convenience, $c(\{i, j\})$ is denoted by c_{ij} in what follows. A set A of edges is called *clique partitioning* if there exists a partition $\{V_1, V_2, \ldots, V_p\}$ of V such that

$$A = \bigcup_{l=1}^{p} \{\{i, j\} \in E \mid i, j \in V_l\}.$$

The aim of CPP is to find a clique partitioning A so as to maximize the sum of the weights $\sum_{\{i,j\} \in A} c_{ij}$. CPP is an NP-hard problem [15] with a wide variety of applications such as qualitative data analysis [4,7], microarray analysis [16], group technology [5,8–11,13,14,17], and community detection [1,2,6,12].

In 1989, Grötschel and Wakabayashi [7] proposed an integer linear programming formulation for CPP. Let $V = \{1, 2, \ldots, n\}$. By introducing variables x_{ij} $(i < j)$ equal to 1 if $\{i, j\} \in A$ and 0 otherwise, the formulation can be written as follows.

$$\begin{aligned}
\text{(P):max.} \quad & \sum_{1 \le i < j \le n} c_{ij} x_{ij} & \\
\text{s. t.} \quad & x_{ij} + x_{jk} - x_{ik} \le 1 & \forall 1 \le i < j < k \le n, \\
& x_{ij} - x_{jk} + x_{ik} \le 1 & \forall 1 \le i < j < k \le n, \\
& -x_{ij} + x_{jk} + x_{ik} \le 1 & \forall 1 \le i < j < k \le n, \\
& x_{ij} \in \{0, 1\} & \forall 1 \le i < j \le n.
\end{aligned}$$

The first three sets of constraints, called *transitivity constraints*, stipulate that for any triples of edges $\{i, j\}$, $\{j, k\}$ and $\{i, k\}$, if $\{i, j\} \in A$ and $\{j, k\} \in A$, then $\{i, k\} \in A$. This formulation has been frequently discussed and applied to many heuristics and exact algorithms [1,2,7,13]. However, (P) is prohibitive even for relatively small graphs because the number of transitivity constraints is $3\binom{n}{3} = \mathrm{O}(n^3)$.

Needless to say, (P) generally contains some redundant constraints, that is, we can obtain the same set of optimal solutions even if some constraints are removed from (P). Indeed, Grötschel and Wakabayashi [7] experimentally confirmed that, for many instances arising in qualitative data analysis, their cutting plane algorithms terminate when only a small fraction of transitivity constraints are added.

In this study, we theoretically derive a certain class of redundant transitivity constraints in (P). Furthermore, we show that the constraints in the derived class are also redundant in the linear programming relaxation of (P). By using our results, the number of constraints treated in exact algorithms and heuristics may be reduced. We note that Dinh and Thai [6] has recently derived redundant constraints in the standard formulation for the modularity maximization problem which is a special case of CPP. In this problem, given an unweighted undirected graph $G = (V, E)$, we find a partitioning of V that maximizes the quality measure called modularity. The analysis of Dinh and Thai focused on the presence or absence of some edges in the given graph. However, we see that their analysis essentially focused on just the signs of the weights of the edges in the corresponding instance of CPP. Our analysis is based on this simple observation, and our results substantially extend the application range of their results.

2 Main results

2.1 Redundant constraints in integer linear programming formulation

By focusing on the signs of the coefficients in the objective function, we present the following class of constraints in (P).

$$(*)\begin{cases} x_{ij}+x_{jk}-x_{ik}\le 1 & \forall 1\le i<j<k\le n,\ c_{ij}<0\wedge c_{jk}<0,\\ x_{ij}-x_{jk}+x_{ik}\le 1 & \forall 1\le i<j<k\le n,\ c_{ij}<0\wedge c_{ik}<0,\\ -x_{ij}+x_{jk}+x_{ik}\le 1 & \forall 1\le i<j<k\le n,\ c_{jk}<0\wedge c_{ik}<0. \end{cases}$$

We provide the formulation which is derived by removing all the constraints in $(*)$ from (P) as follows.

$$\begin{array}{lll} \text{(RP):max.} & \displaystyle\sum_{1\le i<j\le n} c_{ij}x_{ij} & \\ \text{s. t.} & x_{ij}+x_{jk}-x_{ik}\le 1 & \forall 1\le i<j<k\le n,\ c_{ij}\ge 0\vee c_{jk}\ge 0,\\ & x_{ij}-x_{jk}+x_{ik}\le 1 & \forall 1\le i<j<k\le n,\ c_{ij}\ge 0\vee c_{ik}\ge 0,\\ & -x_{ij}+x_{jk}+x_{ik}\le 1 & \forall 1\le i<j<k\le n,\ c_{jk}\ge 0\vee c_{ik}\ge 0,\\ & x_{ij}\in\{0,1\} & \forall 1\le i<j\le n. \end{array}$$

We show below that all the constraints in $(*)$ are redundant in (P).

We begin by introducing some notations. Let $\boldsymbol{x}^*=(x^*_{ij})_{1\le i<j\le n}$ be an arbitrary optimal solution of (RP). Here, we set

$$E_+=\{\{i,j\}\in E\mid c_{ij}\ge 0\}$$

and

$$E^*=\{\{i,j\}\in E\mid x^*_{ij}=1\}.$$

Let $\{(V_1,E_1^*),(V_2,E_2^*),\ldots,(V_p,E_p^*)\}$ denote the set of the connected components of (V,E^*). We now present the following lemma.

Lemma 1 *$(V_l,E_l^*\cap E_+)$ is connected for each $l\in\{1,2,\ldots,p\}$.*

Proof It suffices to show that for any partition $\{S,T\}$ of V_l, there exists an edge in $E_l^*\cap E_+$ whose one endpoint is in S and the other is in T. From the definition of V_l, there exists at least one edge in E_l^* between S and T. Now, suppose that all these edges are not in E_+, that is, the weights of these edges are all negative. Let us focus on a solution of (RP) obtained by changing the values of variables corresponding to those edges from 1 to 0 on $\boldsymbol{x}^*$. This operation can be seen as a partitioning V_l into S and T, and this solution is feasible for (RP). The reason is that every transitivity constraint including an edge between S and T is satisfied since there are at least two terms equal to 0. See Fig. 1 for an illustration of such transitivity constraints.

We see that the objective value of this solution is strictly greater than that of $\boldsymbol{x}^*$. This contradicts the optimality of $\boldsymbol{x}^*$. □

Our main result is the following theorem.

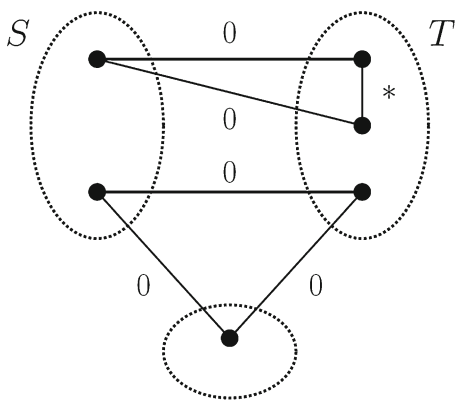

Fig. 1 All types of transitivity constraints containing an edge between S and T (labels of edges represent values of corresponding variables in $\boldsymbol{x}^*$; especially "$*$" means that the variable has a value in $\{0, 1\}$)

Theorem 1 (P) *and* (RP) *have the same set of optimal solutions.*

Proof It suffices to show that any optimal solution $\boldsymbol{x}^*$ of (RP) is feasible for (P). This is equivalent to the statement that (V_l, E_l^*) is complete for each $l \in \{1, 2, \ldots, p\}$.

From Lemma 1, for any distinct vertices $i, j \in V_l$, there exists a path $i = u_0, u_1, \ldots, u_q = j$ on $E_l^* \cap E_+$. Since $\{u_0, u_1\} \in E_+$ (and $\{u_1, u_2\} \in E_+$), the transitivity constraint

$$x_{u_0u_1} + x_{u_1u_2} - x_{u_0u_2} \leq 1$$

is contained in (RP). (Note that in this notation it is necessary that $u_0 < u_1 < u_2$ holds. If it is not the case, one should swap the order of the indices of the variables appropriately.) Thus, by using $x^*_{u_0u_1} = x^*_{u_1u_2} = 1$, we have $x^*_{u_0u_2} = 1$. Similarly, since $\{u_2, u_3\} \in E_+$, the transitivity constraint

$$x_{u_0u_2} + x_{u_2u_3} - x_{u_0u_3} \leq 1$$

is also contained in (RP). (Note that $\{u_0, u_2\}$ is not necessarily contained in E_+.) Hence, by substituting $x^*_{u_0u_2} = x^*_{u_2u_3} = 1$, we obtain $x^*_{u_0u_3} = 1$. Repeatedly applying this operation, we derive $x^*_{u_0u_q} = x^*_{ij} = 1$. This implies that (V_l, E_l^*) is complete. □

2.2 Redundant constraints in linear programming relaxation

The redundant constraints in (P), which we revealed above, are also redundant in its linear programming relaxation. Let $(\overline{\mathrm{P}})$ and $(\overline{\mathrm{RP}})$ denote the linear programming relaxations which are derived by replacing the set of constraints $x_{ij} \in \{0, 1\}$ by $x_{ij} \in [0, 1]$ in (P) and (RP), respectively. Clearly, $(\overline{\mathrm{RP}})$ is a relaxation of $(\overline{\mathrm{P}})$. We show below that all the constraints in $(*)$ are also redundant in $(\overline{\mathrm{P}})$.

We begin by introducing some tools and notations. Let $\boldsymbol{x}^* = (x^*_{ij})_{1 \leq i < j \leq n}$ be an arbitrary optimal solution of $(\overline{\mathrm{RP}})$. For convenience, we consider the residual graph of $\boldsymbol{x}^*$, denoted by $\boldsymbol{d}^*$. Namely,

$$\boldsymbol{d}^* = (d^*_{ij})_{1 \leq i < j \leq n} = \mathbf{1} - \boldsymbol{x}^*.$$

Then, we see that the transitivity constraints for $\boldsymbol{x}^*$ in ($\overline{\mathrm{P}}$) correspond to the triangle inequalities for $\boldsymbol{d}^*$. For instance,

$$x^*_{ij} + x^*_{jk} - x^*_{ik} \leq 1 \;\Leftrightarrow\; d^*_{ik} \leq d^*_{ij} + d^*_{jk}$$

holds for the first set of transitivity constraints. Here, we set

$$\overline{E}^* = \{\{i, j\} \in E \mid d^*_{ij} < 1 \,(\Leftrightarrow x^*_{ij} > 0)\}.$$

Let $\{(V_1, \overline{E}^*_1), (V_2, \overline{E}^*_2), \ldots, (V_p, \overline{E}^*_p)\}$ denote the set of the connected components of $(V, \overline{E}^*)$. As well as Lemma 1, we have the following fact. The proof is omitted since it is essentially the same as that of Lemma 1.

Lemma 2 *$(V_l, \overline{E}^*_l \cap E_+)$ is connected for each $l \in \{1, 2, \ldots, p\}$.*

We now present the following theorem.

Theorem 2 *($\overline{P}$) and ($\overline{RP}$) have the same set of optimal solutions.*

Proof It suffices to show that any optimal solution $\boldsymbol{x}^* = (x^*_{ij})_{1 \leq i < j \leq n}$ of ($\overline{\mathrm{RP}}$) is feasible for ($\overline{\mathrm{P}}$). As mentioned above, it is equivalent to the statement that the residual graph $\boldsymbol{d}^*$ satisfies all the triangle inequalities corresponding to the transitivity constraints in ($\overline{\mathrm{P}}$). Indeed, it is enough to confirm that the triangle inequalities for all triples of vertices in each connected component of $(V, \overline{E}^*)$ are satisfied. The reason is that the other inequalities are always satisfied because there are at least two terms equal to 1 in each inequality. Here we used the fact that, from the definition of $(V, \overline{E}^*)$, $d^*_{ij} = 1$ for any $i, j \in V$ in different connected components. See Fig. 2 for an illustration of the triangle inequalities over different connected components.

From Lemma 2, for any distinct vertices $i, j \in V_l$, there exists at least one path on $\overline{E}^*_l \cap E_+$. The shortest one of these paths and its length are denoted by $i = u_0, u_1, \ldots, u_q = j$ and d'_{ij}, respectively. Now, repeatedly applying the triangle inequalities, corresponding to the transitivity constraints in ($\overline{\mathrm{RP}}$), to $\boldsymbol{d}^*$ as

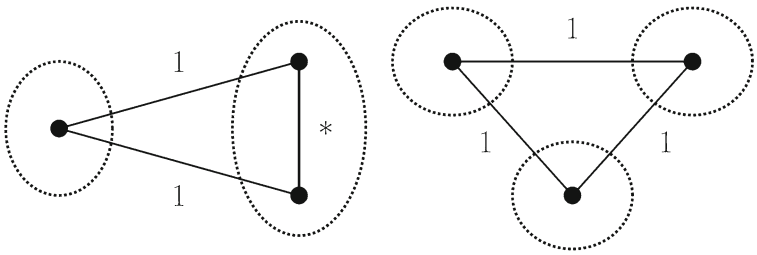

Fig. 2 All types of triangle inequalities over different connected components (labels of edges represent values of corresponding variables in $\boldsymbol{d}^*$; especially "$*$" means that the variable has a value in [0, 1])

$$\begin{aligned} d'_{ij} &= d^*_{u_0u_1} + d^*_{u_1u_2} + \cdots + d^*_{u_{q-1}u_q} \\ &\geq d^*_{u_0u_2} + d^*_{u_2u_3} + \cdots + d^*_{u_{q-1}u_q} \\ &\geq \cdots \geq d^*_{u_0u_{q-1}} + d^*_{u_{q-1}u_q} \geq d^*_{u_0u_q} = d^*_{ij}, \end{aligned}$$

we have $d'_{ij} \geq d^*_{ij}$. (Note again that in this notation it is necessary that $u_0 < u_1 < \cdots < u_q$ holds. If it is not the case, one should swap the order of the indices of the variables appropriately). Here, let $d_{ij} = \min\{d'_{ij}, 1\}$ and construct $\boldsymbol{d} = (d_{ij})_{i<j}$ by gathering d_{ij} for all $i, j \in V_l$. Clearly, $d_{ij} \geq d^*_{ij}$. Furthermore, we see that $\boldsymbol{d}$ satisfies the triangle inequalities because

$$\begin{aligned} d_{ij} + d_{jk} &= \min\{d'_{ij}, 1\} + \min\{d'_{jk}, 1\} \\ &\geq \min\{d'_{ij} + d'_{jk}, 1\} \\ &\geq \min\{d'_{ik}, 1\} = d_{ik}. \end{aligned}$$

The first inequality follows because $\boldsymbol{d}' = (d'_{ij})_{i<j}$ is nonnegative. The second inequality follows because $\boldsymbol{d}'$ satisfies the triangle inequalities. In fact, $d_{ij} = d^*_{ij}$ for all $i, j \in V_l$. If this is shown, then we see that $\boldsymbol{d}^*$ satisfies the triangle inequalities in each component, namely, the proof is completed.

Suppose that there exist some $i, j \in V_l$ such that $d_{ij} \neq d^*_{ij}$. Recalling $d_{ij} \geq d^*_{ij}$, we have $d_{ij} > d^*_{ij}$. Then, since $d^*_{ij} < d_{ij} \leq 1$, we obtain $\{i, j\} \in \overline{E}^*_l$. Suppose now that $\{i, j\}$ is also contained in E_+. Then, $\{i, j\}$ is one of the paths between i and j on $\overline{E}^*_l \cap E_+$. Since d'_{ij} is the length of the shortest path, we have $d_{ij} \leq d'_{ij} \leq d^*_{ij}$. Putting this condition together with $d_{ij} \geq d^*_{ij}$, we obtain $d_{ij} = d^*_{ij}$. Thus, under the first assumption $d_{ij} \neq d^*_{ij}$, we have $\{i, j\} \notin E_+$. Let us focus on a solution

$$\boldsymbol{x}' = \begin{cases} 1 - d_{ij} & \text{if } i, j \in V_l, \\ x^*_{ij} & \text{otherwise,} \end{cases}$$

of ($\overline{\text{RP}}$). This solution is feasible for ($\overline{\text{RP}}$) because, as observed above, $\boldsymbol{d}$ satisfies the triangle inequalities. By a simple calculation, we can confirm that the objective value of $\boldsymbol{x}'$ is strictly greater than that of $\boldsymbol{x}^*$. This contradicts the optimality of $\boldsymbol{x}^*$. □

3 Computational experiments

We compare the number of transitivity constraints of (P) and (RP). Moreover, we measure computation time to find an optimal solution of each formulation, and verify the effectiveness of the reduction. To this end, we conduct computational experiments on 21 instances commonly used to evaluate new methods for CPP.

The first 9 instances arise in group technology. Let us consider the situation where there is a set of p parts produced by a set of q machines. In group technology, given a $p \times q$ matrix (a_{ij}) where $a_{ij} = 1$ if a part i is processed by a machine j and $a_{ij} = 0$ otherwise, we find a suitable partitioning of a set of parts and machines to develop an

Table 1 The results of computational experiments

Instance	n	negative (%)	(P)		(RP)	
			$\#constr.$	time(s)	$\#constr.$	time(s)
KKV [9]	24	37.3	6,072	0.19	5,209	0.19
MALa [10]	25	34.7	6,900	0.20	6,100	0.20
MALb [10]	25	32.7	6,900	0.22	6,187	0.20
KIN [8]	38	39.4	25,308	0.84	21,211	0.77
GRO [9]	43	38.3	37,023	328.09	31,642	942.60
BUR [14]	55	38.2	78,705	2.67	66,598	2.36
CHA [5]	59	32.8	97,527	316.98	84,297	231.38
MIL [11]	60	44.6	102,660	162.04	82,047	77.78
LEE [14]	70	43.4	164,220	5.98	133,174	5.08
Wildcats [7]	30	37.2	12,180	0.34	10,043	0.31
Cars [7]	33	27.3	16,368	0.52	14, 708	0.47
Workers [7]	34	37.8	17,952	0.66	14, 896	0.58
Cetacea [7]	36	72.7	21,420	0.61	9,598	0.31
Micro [7]	40	47.8	29,640	0.94	22,201	0.73
Soybean [4]	47	58.6	48,645	1.58	31,171	1.00
UNO [7]	54	59.2	74,412	2.50	45,756	1.59
Human [3]	132	77.3	1,123, 980	4,618.19	451,098	1,242.36
UNO1b [7]	139	52.3	1,313,967	69.13	910,908	46.33
UNO2b [7]	145	24.4	1,492,920	81.22	1,310,497	71.98
UNO1a [7]	158	60.3	1,934,868	114.07	1,161,623	62.87
UNO2a [7]	158	35.5	1,934,868	114.27	1,542,583	90.24

efficient cellular manufacturing system. An instance of CPP is constructed as follows. Let V be a union of a set of parts and a set of machines. An edge between a part i and a machine j has weight 1 if $a_{ij} = 1$ and -1 otherwise. An edge between two parts or two machines has weight 0. The remaining 12 instances arise in the problem of aggregation of binary relations. In this problem, given p binary relations on a set N, we find an equivalence relation on N which presumably aggregates the given relations. An instance of CPP is constructed as follows. Let V be N. If $i, j \in V$ are said to be equivalent on many given relations, c_{ij} is set to a positive value. On the other hand, if $i, j \in V$ are said to be equivalent on only a few given relations, c_{ij} is set to a negative value. For more detailed description, see the literature [7].

All instances were solved using Gurobi Optimizer 4.5.0 on a Windows-based computer with 3.6 GHz quad-core processors and 4 GB RAM. We note that Gurobi Optimizer was used with default parameters, and it executed branch-and-cut algorithms for the formulations.

The results are shown in Table 1. The first column lists the name and source of each instance. The second and third columns list the number of vertices and the fraction of negative edges of each instance. The columns of (P) and (RP) provide the

number of transitivity constraints and computation time to find an optimal solution of each formulation. We see that the number of transitivity constraints is certainly reduced by our result. In particular, our reduction is quite effective for instances with relatively large fraction of negative edges. In fact, 55.2 and 59.9 % of the transitivity constraints are removed for `Cetacea` and `Human`, respectively. Moreover, except for `KKV`, `MALa`, and `GRO`, computation times to find an optimal solution are all shortened. We presume that the computation time for `GRO` increased considerably because our reduction interfered with generation of some effective cutting planes.

Finally, we note that most instances were solved at the root node of the branch-and-bound tree by adding a large number of cutting planes by Gurobi Optimizer.

4 Concluding remarks

In this study, we introduced a class of redundant transitivity constraints in the standard formulation for CPP. Furthermore, we showed that the transitivity constraints in the derived class are also redundant in the linear programming relaxation. As a result, we obtained a compact formulation and a compact linear programming relaxation for CPP.

Replacing the standard formulation and its linear programming relaxation by our compact formulations in some existing algorithms may shorten their computation time and improve their memory-efficiency. In fact, for the modularity maximization problem, Miyauchi and Miyamoto [12] utilized the compact linear programming relaxation by Dinh and Thai [6] in their methods, and succeeded in solving the linear programming relaxation for large graphs.

We think that there are two future directions. The first direction is to strengthen our results, that is, to derive a larger class of redundant constraints in (P) and ($\overline{\text{P}}$). For instance, let us focus on the following class of constraints

$$(**)\ \begin{cases} x_{ij} + x_{jk} - x_{ik} \leq 1 & \forall 1 \leq i < j < k \leq n,\ c_{ij} + c_{jk} < 0, \\ x_{ij} - x_{jk} + x_{ik} \leq 1 & \forall 1 \leq i < j < k \leq n,\ c_{ij} + c_{ik} < 0, \\ -x_{ij} + x_{jk} + x_{ik} \leq 1 & \forall 1 \leq i < j < k \leq n,\ c_{jk} + c_{ik} < 0, \end{cases}$$

which contains the class ($*$). Are the constraints in ($**$) redundant in (P) and ($\overline{\text{P}}$)? We have not found any NO-instance of this question yet. The second future direction is to apply our analysis to formulations of other optimization problems. There are a large number of problems which are formulated using transitivity constraints. Is it possible to derive a class of redundant constraints in such formulations by similar analysis?

Acknowledgments The authors would like to thank two anonymous referees for their valuable suggestions. The second author is supported by the Grant-in-Aid for JSPS Fellows.

References

1. Agarwal, G., Kempe, D.: Modularity-maximizing graph communities via mathematical programming. Eur. Phys. J. B **66**, 409–418 (2008)

2. Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Liberti, L., Perron, S.: Column generation algorithms for exact modularity maximization in networks. Phys. Rev. E **82**, 046112 (2010)
3. Blake, C.L., Merz, C.J.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (1998). http://archive.ics.uci.edu/ml/. Accessed 17 Apr 2014
4. Brusco, M.J., Köhn, H.F.: Clustering qualitative data based on binary equivalence relations: neighborhood search heuristics for the clique partitioning problem. Psychometrika **74**, 685–703 (2009)
5. Chan, H.M., Milner, D.A.: Direct clustering algorithm for group formulation in cellular manufacturing. J. Manuf. Syst. **1**, 65–74 (1982)
6. Dinh, T.N., Thai, M.T.: Towards optimal community detection: From trees to general weighted networks, Internet Math. (accepted pending revision)
7. Grötschel, M., Wakabayashi, Y.: A cutting plane algorithm for a clustering problem. Math. Program. **45**, 59–96 (1989)
8. King, K.R.: Machine component group formation in group technology. Omega **8**, 193–199 (1980)
9. Kumar, R.K., Kusiak, A., Vannelli, A.: Grouping of parts and components in flexible manufacturing systems. Eur. J. Oper. Res. **24**, 387–397 (1986)
10. Malakooti, B., Yang, Z.: Multiple criteria approach and generation of efficient alternatives for machine-part family formulation in group technology. IIE Trans. **34**, 837–846 (2002)
11. Miltenburg, J., Zhang, W.: A comparative evaluation of nine well-known algorithms for solving the cell formation in group technology. J. Oper. Manag. **10**, 44–72 (1991)
12. Miyauchi, A., Miyamoto, Y.: Computing an upper bound of modularity. Eur. Phys. J. B **86**, 302 (2013)
13. Oosten, M., Rutten, J.H.G.C., Spieksma, F.C.R.: The clique partitioning problem: facets and patching facets. Networks **38**, 209–226 (2001)
14. Rogers, D.F., Kulkarni, S.S.: Optimal bivariate clustering and a genetic algorithm with an application in cellular manufacturing. Eur. J. Oper. Res. **160**, 423–444 (2005)
15. Wakabayashi, Y.: Aggregation of binary relations: algorithmic and polyhedral investigations, Ph.D. Thesis, Universität Augsburg (1986)
16. Wang, H., Alidaee, B., Glover, F., Kochenberger, G.: Clustering of microarray data via clique partitioning. J. Comb. Optim. **10**, 77–92 (2005)
17. Wang, H., Alidaee, B., Glover, F., Kochenberger, G.: Solving group technology problems via clique partitioning. Int. J. Flex. Manuf. Syst. **18**, 77–97 (2006)