Acta Universitatis Sapientiae

Informatica

Volume 7, Number 1, 2015

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

DE GRUYTER OPEN

Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 5–17

DOI: 10.1515/ausi-2015-0009

Online hypergraph coloring with rejection

Csanád IMREH
University of Szeged
Institute of Informatics
email: cimreh@inf.u-szeged.hu

Judit NAGY-GYÖRGY
University of Szeged
Bolyai Institute
email: Nagy-Gyorgy@math.u-szeged.hu

Abstract. In this paper we investigate the online hypergraph coloring problem with rejection, where the algorithm is allowed to reject a vertex instead of coloring it but each vertex has a penalty which has to be paid if it is not colored. The goal is to minimize the sum of the number of the used colors for the accepted vertices and the total penalty paid for the rejected ones. We study the online problem which means that the algorithm receives the vertices of the hypergraph in some order $v_1, \ldots, v_n$ and it must decide about v_i by only looking at the subhypergraph $H_i = (V_i, E_i)$ where $V_i = \{v_1, \ldots, v_i\}$ and E_i contains the edges of the hypergraph which are subsets of V_i. We consider two models: in the full edge model only the edges where each vertex is accepted must be well-colored, in the trace model the subsets of the edges formed by the accepted vertices must be well colored as well. We consider proper and conflict free colorings. We present in each cases optimal online algorithms in the sense that they achieve asymptotically the smallest possible competitive ratio.

1 Introduction

A coloring of a hypergraph is an assignment of positive integers to the vertices of the hypergraph so that every edge satisfy some property. We consider

Computing Classification System 1998: F.1.2
Mathematics Subject Classification 2010: 68W27
Key words and phrases: online algorithms, hypergraph coloring, competitive ratio

two different versions of coloring. In proper hypergraph coloring each edge must contain vertices having different colors. In conflict free (we will use the abbreviation cf) coloring each edge must contain a unique vertex which has different color to the other vertices of the edge. In the online hypergraph coloring problem the algorithm receives the vertices of the hypergraph with n vertices in some order $v_1, \ldots, v_n$ and it must color v_i by only looking at the subhypergraph $H_i = (V_i, E_i)$ where $V_i = \{v_1, \ldots, v_i\}$ and E_i contains the edges of the hypergraph which are subsets of V_i.

We will evaluate the efficiency of the online algorithms by the competitive ratio (see [4, 10]) where the online algorithm is compared to the optimal offline algorithm. We say that an online algorithm is C-competitive if its cost is at most C times larger than the optimal cost.

Online proper coloring of hypergarphs first was studied in [9] where it was proven that no online algorithm exists for 2-colorable k-uniform hypergraphs which can color them with less colors than $\lceil n/(k-1) \rceil$, and it was proved that algorithm FF colors these hypergraphs with this much colors. This means that the best possible competitive ratio is $\lceil n/(k-1) \rceil/2$ for this class of hypergraphs. Furthermore some special classes were also studied: the hypergraphs with given matching number and projective planes. Later randomized algorithms were studied for online proper coloring of hypergarphs in [8] where the deterministic $\Omega(n/k)$ lower bound was extended to randomized algorithms. This lower bound was also proved in the case of a more general transparent model. In [11] the online and quasionline hypergraph proper coloring problem was studied for intervals and wedges.

Online cf-coloring of hypergraph was defined in [5] where the authors considered the case where the input is a set of n points on the line, and R is the set of the intervals of the line. They present an algorithm which uses $O(\log^2(n))$ colors and also prove a matching lower bound. Online cf-coloring of intervals was further studied in [2] where several coloring models were defined and compared. The online cf-coloring of other more general hypergraphs was studied in [3] and [6].

In [7] the graph coloring problem with rejection was investigated. In this model a penalty value is assigned to each vertex and the algorithm has to choose a subset of vertices, and find a proper coloring of the induced subgraph defined by this subset. The elements of the subset are called accepted vertices the other ones are called rejected. The goal is to minimize the sum of the number of colors used to color the accepted vertices and the total penalty paid for the rejected vertices. In [7] both the online and the offline versions of the problems are investigated.

In this paper we extend graph coloring with rejection into hypergraph coloring with rejection. There are two ways to extend the model. In the full edge model we have to color correctly only the edges where each vertices are accepted from the edge. In the trace model we have to color correctly the subhypergraph which consists of the accepted vertices and the edges which are the accepted subsets of the original edges. Note that in the special case of graphs the two models are identical. We consider both proper and cf-coloring in both models.

Main results: We studied four models since we had two possibilities for the coloring (proper and cf) and two possibilities to handle rejection (full edge, trace). In the full edge model with proper coloring we present for every $\varepsilon > 0$ an online algorithm $\mathcal{A}_\varepsilon$ and n_ε such that $\mathcal{A}_\varepsilon$ is at most $\lceil n/(k-1)\rceil/2 + \varepsilon$ competitive on k-uniform hypergraphs with at least n_ε vertices for $k \geq 3$. This competitive ratio is asymptotically the best possible since it follows from online hypergraph coloring that no online algorithm exists with smaller competitive ratio than $\lceil n/(k-1)\rceil/2$ for k-uniform hypergraphs. In case of full edge model and cf-coloring we present an $((n-1)/\varphi + \varphi)$-competitive algorithm for hypergraphs of n-vertices where $\varphi = (1+\sqrt{5})/2$. In the trace model we present an algorithm which is $(2 + (n-2)/\varphi)$-competitive for both the proper and cf coloring models. All of these algorithms are asymptotically optimal since we prove that no online algorithm exists which is $(Cn + D$-competitive) in any these models for hypergraphs containing n vertices and some constants $C < 1/\varphi$, D.

2 Notation

In this paper on hypergraph we mean the structure $H = (V, E)$ where V is the finite set of the hypergraph's vertices and $E \subseteq \rho(V)$ is the set of the edges where $\rho(V)$ is the set of the nonempty subsets of V. We suppose that each edge has at least two elements.

We consider the following two colorings. A proper coloring of a hypergraph is an assignment of positive integers (called colors) to the vertices of the hypergraph so that each edge contains at least two vertices with different colors. For a hypergraph H the minimum number of colors which is enough to color the hypergraph is called the proper chromatic number of the hypergraph and denoted by $\chi_P(H)$. A conflict free (cf for short) coloring of a hypergraph is an assignment of positive integers (called colors) to the vertices of the hypergraph so that each edge contains a unique color, a vertex which has different color

to the other vertices of the edge. For a hypergraph H the minimum number of colors which is enough to cf-color the hypergraph is called the cf-chromatic number of the hypergraph and denoted by $\chi_{cf}(H)$.

We will consider the hypergraph coloring with rejection. This means that we can reject the coloring of some vertices, but each vertex v has a penalty denoted by $p(v)$ and our goal is to minimize the sum number of used colors for the accepted vertices and the total penalty paid for the rejection of the other ones. We consider two rejection models. In the full edge model we have to color correctly only the edges where each vertices are accepted from the edge. This means that rejecting some vertex of an edge ensures that it is well colored. We also consider a different model called trace model. In this new model we consider the subhypergraph which consists of the accepted vertices and the edges which are the accepted subsets of the original edges. And this subhypergraph must to be well-colored in each step. Therefore in the trace model the rejection of some vertices of an edge does not ensure that it is well colored we have to take care of the remaining vertices. We can define the problem for both the proper and the conflict free coloring.

We consider the online problem. An online hypergraph (defined first in [1]) is a structure $H_< = (H, <)$ where H is a hypergraph and $<$ is a linear ordering of its vertices. We call a vertex the first, second,..., and ending vertex of an edge according to the ordering $<$. An online hypergraph coloring algorithm has to color the i-th vertex only knowing the subhypergraph $H_i = (V_i, E_i)$ where V_i contains the first i vertices and E_i contains the edges of the hypergraph which are subsets of V_i. This means that the online algorithm receives information about the edges only when the last vertex of the edge arrives. We will use the well-known greedy algorithm FF (First Fit) to color the accepted vertices of the online hypergraphs. FF uses the smallest color for each vertex which does not hurt the rule of the coloring. In case of proper coloring it uses the smallest color which does not cause a monochromatic edge. In the case of cf-coloring it uses the smallest color which does not yield an edge where none of the colors is unique. We note that in the trace model it might happen that the online algorithm is forced to accept some vertices. If it has accepted and colored two vertices by the same colors then a further vertex which forms an edge with these vertices must be accepted since otherwise the remaining edge of the subhypergraph is not well-colored. In this situation when the rejection of a vertex causes an incorrect coloring of the remaining edges we say that the vertex is forced to be accepted.

Usually in the theory of online computation the efficiency of the online algorithms is measured by the competitive ratio (see [4, 10]) where the online

algorithm is compared to the optimal offline algorithm. We denote the cost of the online algorithm $\mathcal{A}$ on an online hypergraph $H_<$ and penalty function p by $\mathcal{A}(H_<, p)$ and we will denote the optimal cost by $\mathrm{opt}(H_<, p)$. An algorithm is called c-competitive if $\mathcal{A}(H_<, p) \le c \cdot \mathrm{opt}(H_<, p)$ for every H and p. Since no constant competitive online algorithm exists we will consider the competitive ratio as a function of the number of vertices, denoted by n.

We also use the following notion from the theory of hypergraphs. A hypergraph is called k-uniform if each edge contains k vertices.

3 Online coloring hypergraphs with rejection in the full edge model

3.1 Proper coloring

Note that a lower bound of $\lceil n/(k-1)\rceil/2$ on k-uniform 2-proper-colorable hypergraphs for $k \ge 3$ comes from the case without rejection. Surprisingly, the following theorem shows that one can reach this competitive ratio in the asymptotical sense for the more general case where rejection is also allowed.

Theorem 1 *If $k \ge 3$, then for every $\varepsilon < 1/8$ there is an online algorithm $\mathcal{A}_\varepsilon$ and n_ε, such that $\mathcal{A}_\varepsilon$ is at most $\lceil n/(k-1)\rceil/2 + \varepsilon$ competitive on k-uniform hypergraphs with at least n_ε vertices.*

Proof. Let $\delta_\varepsilon = 2\varepsilon/(k-2)$, $n_\varepsilon = \dfrac{2k-2}{\delta_\varepsilon - \delta_\varepsilon^2(2k-2)}$. Define the following algorithm:

Algorithm $\mathcal{A}_\varepsilon$: If the penalty of the next vertex is less then δ_ε reject it, otherwise color it by algorithm FF.

Denote by A the set of the colored and by B the set of the rejected vertices by $\mathcal{A}_\varepsilon$, and $\chi_{\mathcal{A}_\varepsilon}(A)$ the number of colors used by $\mathcal{A}_\varepsilon$. Let $n = |A| + |B|$. Then the cost of $\mathcal{A}_\varepsilon$ is $\chi_{\mathcal{A}_\varepsilon}(A) + p(B) \le \lceil |A|/(k-1)\rceil + \delta_\varepsilon |B|$.

We have three cases.

Case 1. Suppose that the optimal algorithm uses at least 2 colors. In this case its cost is at least 2. Therefore we obtain that

$$\frac{\mathrm{cost}_{\mathcal{A}_\varepsilon}(H_<, p)}{\mathrm{opt}(H_<, p)} \le \frac{\mathrm{cost}_{\mathcal{FF}_\varepsilon}(H_<, p)}{2} \le \frac{1}{2}\left\lceil \frac{n}{k-1} \right\rceil.$$

Case 2. Suppose that the optimal algorithm uses one color. We state that in this case it must reject at least $\chi_{\mathcal{A}_\varepsilon}(A) - 1$ vertices from A. First observe that FF uses color j for a new vertex only when the vertex ends for each $i < j$ and edge containing $k-1$ vertices colored by i. Therefore for each pair of color classes of A there exists an edge which contains only vertices from these color classes. This means that the optimal algorithm cannot accept all vertices from two different color classes of A since it could not color them correctly by 1 color. Thus it follows that the optimal algorithms must reject at least $\chi_{\mathcal{A}_\varepsilon}(A) - 1$ vertices from A.

If $|A| > 0$ then

$$\begin{aligned}
\frac{\mathrm{cost}_{\mathcal{A}_\varepsilon}(H_<, p)}{\mathrm{opt}(H_<, p)} &\le \frac{\chi_{\mathcal{A}_\varepsilon}(A) + p(B)}{1 + \delta_\varepsilon(\chi_{\mathcal{A}_\varepsilon}(A) - 1)} \\
&\le \frac{\chi_{\mathcal{A}_\varepsilon}(A)}{\delta_\varepsilon(\chi_{\mathcal{A}_\varepsilon}(A))} + p(B) \\
&\le \frac{1}{\delta_\varepsilon} + \delta_\varepsilon |B| \\
&\le \frac{1}{\delta_\varepsilon} + \delta_\varepsilon n \\
&\le \frac{n}{2(k-1)}
\end{aligned}$$

where the last inequality comes from the definition of n_ε.

If $|A| = 0$ then

$$\frac{\mathrm{cost}_{\mathcal{A}_\varepsilon}(H_<, p)}{\mathrm{opt}(H_<, p)} \le p(B) \le \delta_\varepsilon n \le \frac{n}{2(k-1)}.$$

Case 3. Suppose that the optimal algorithm uses no colors, i.e. it rejects all vertices. If $|A| = 0$ then $\mathcal{A}_\varepsilon$ optimal. Otherwise

$$\begin{aligned}
\frac{\mathrm{cost}_{\mathcal{A}_\varepsilon}(H_<, p)}{\mathrm{opt}(H_<, p)} &\leq \frac{\chi_{\mathcal{A}_\varepsilon}(A) + p(B)}{p(A) + p(B)} \\
&\leq \frac{\chi_{\mathcal{A}_\varepsilon}(A)}{p(A)} \\
&\leq \frac{\frac{|A|}{k-1} + \frac{k-2}{k-1}}{\delta_\varepsilon |A|} \\
&\leq \frac{1}{\delta_\varepsilon(k-1)} + \frac{k-2}{\delta_\varepsilon(k-1)} = \frac{1}{\delta_\varepsilon} \\
&\leq \frac{n}{2(k-1)}
\end{aligned}$$

where the first and third inequality come from the definition of the algorithm and the case, the fourth one by $|A| \geq 1$, the last two come from the definition of n_ε. □

3.2 Conflict-free coloring

In the full edge model for cf coloring we considered the following algorithm.

Algorithm $\mathcal{B}$: If the penalty of the next vertex is less then $1/\varphi$ reject it, otherwise color it by FF.

Theorem 2 *Algorithm $\mathcal{B}$ is $(n-1)/\varphi + \varphi$-competitive on hypergraphs on n vertices where $\varphi = (1+\sqrt{5})/2$.*

Proof. Consider an input hypergraph denoted by $H_<$. Denote A the set of the colored and B the set of the rejected vertices by $\mathcal{B}$, and $\chi_{\mathcal{B}}(A)$ the number of colors used by $\mathcal{B}$. Let $n = |A| + |B|$. Then the cost of $\mathcal{B}$ is

$$\chi_{\mathcal{B}}(A) + p(B) \leq |A| + |B|/\varphi.$$

We have three cases.
Case 1. Suppose that the optimal algorithm uses at least 2 colors. In this case the optimal cost is at least 2, therefore

$$\begin{aligned}\frac{\mathrm{cost}_{\mathcal{B}}(H_<,p)}{\mathrm{opt}(H_<,p)} &\le \frac{\chi_{\mathcal{B}}(A)+p(B)}{2} \le \frac{|A|+|B|/\varphi}{2}\\ &\le \frac{n}{2} \le \frac{n-1}{\varphi}+\varphi.\end{aligned}$$

Case 2. Suppose that the optimal algorithm uses one color. If the input hypergraph has edges colored by $\mathcal{B}$ then the optimal algorithm have to reject at least one vertex with penalty at least $1/\varphi$. Therefore the optimal cost is at least $1+1/\varphi$, thus we obtain that

$$\begin{aligned}\frac{\mathrm{cost}_{\mathcal{B}}(H_<,p)}{\mathrm{opt}(H_<,p)} &\le \frac{\chi_{\mathcal{B}}(A)+p(B)}{1+1/\varphi}\\ &\le \frac{|A|+|B|/\varphi}{\varphi} \le \frac{n}{\varphi}.\end{aligned}$$

If the input hypergraph has no edge colored by $\mathcal{B}$ then the optimal cost is at least 1 but in this case $\chi_{\mathcal{B}}(A) \le 1$ thus

$$\frac{\mathrm{cost}_{\mathcal{B}}(H_<,p)}{\mathrm{opt}(H_<,p)} \le \frac{\chi_{\mathcal{B}}(A)+p(B)}{1} \le 1+(|B|)/\varphi \le 1+\frac{n-1}{\varphi}.$$

Case 3. Suppose that the optimal algorithm uses no colors, i.e. it rejects all vertices. Then

$$\frac{\mathrm{cost}_{\mathcal{B}}(H_<,p)}{\mathrm{opt}(H_<,p)} \le \frac{\chi_{\mathcal{B}}(A)+p(B)}{p(A)+p(B)} \le \frac{\chi_{\mathcal{B}}(A)}{p(A)} \le \frac{|A|}{|A|/\varphi} = \varphi \le \frac{n-1}{\varphi}+\varphi.$$

□

Note that considering the online cf-coloring without rejection of 2-cf-colorable hypergraphs we can obtain the following result.

Lemma 3 *No online cf-coloring algorithm uses less then* $n-1$ *colors on 2-cf-colorable hypergraphs on* n *vertices.*

Proof. Give vertices until two of them are colored by the same color. Suppose that online algorithm colors v_i and v_j with the same color. Then reveal edges in the m-th phase $\{v_i, v_j, v_m\}$ and $\{v_i, v_j, v_\ell, v_m\}$ for all $\ell < m$, $\ell \neq i, j$. Then the online algorithm must use a new color for each new vertex.

This hypergraphs is 2-cf-colorable: v_i is in the first color class and the other vertices are in the second. □

This observation proves that no online algorithm can be better than $(n-1)/2$ competitive for online cf-coloring of hypergraphs, and this bounds holds as well for the model with rejection since we can use penalty ∞ for all vertices.

On the other hand we can extend the idea of this lower bound to the model with rejection and prove that the competitive ratio of $\mathcal{B}$ is the best possible in the asymptotical sense as the following theorem shows.

Theorem 4 *No online algorithm exists which is* $Cn + D$*-competitive in the problem of conflict free coloring with rejection in the full edge model for hypergraphs containing* n *vertices for some constants* $C < 1/\varphi$ *and* D*.*

Proof. Suppose that, on the contrary, there exist constants $C < 1/\varphi$ and D and an online algorithm $\mathcal{C}$ which is $Cn + D$-competitive. First we present vertices with penalty $1/\varphi$ and no edge until two of them are colored by the same color or the number of vertices reaches a number $n_1 > D/(1/\varphi - C)$. If none of these vertices received the same color then the sequence ends, the optimal algorithm colors these vertices by one color and its cost is 1. The online algorithms pays at least $1/\varphi$ for each of them thus the online cost is at least n_1/φ and we obtain a contradiction since $n_1/\varphi > Cn_1 + D$ by the definition of n_1.

Now suppose that the online algorithm colors two accepted vertices by the same colors. Let these vertices be v_i and v_k, where $i < k$. Note that the first phase of the inputs ends by vertex v_k. In this case we continue the sequence with the points $v_{k+1}, \ldots, v_n$ where $n > (D + k/\varphi)/(1/\varphi - C)$. Each such vertex v_q has penalty ∞, and each of them ends the edges (v_i, v_k, v_q) and (v_i, v_k, v_s, v_q) for $s < q$ and $s \neq i, k$. Then each such vertex must be accepted, and these edges force a new color for each of them. Therefore, v_i and v_k have the same color and the other accepted vertices are colored by different colors. Thus the cost of the online algorithm is $m/\varphi + n - m - 1$, where m is the number of rejected vertices in the first phase. Therefore its cost is at least $(k-2)/\varphi + n - k + 1$. On the other hand an optimal algorithm rejects v_i and accepts all the other vertices and colors them by color 1. Then its cost is $1 + 1/\varphi$. Therefore the ratio of the online and offline costs is at least

$$\frac{(k-2)/\varphi + n - k + 1}{1 + 1/\varphi} > \frac{n}{\varphi} - \frac{k}{\varphi} > Cn + D,$$

and the theorem follows. □

4 The trace model

In the trace model we analyze the following online algorithm. The same algorithm is defined for both the proper and cf-colorings, the difference comes from the fact that the algorithm uses FF to color the vertices, and it might assign different colors in the two models. Moreover the set of the accepted vertices might be different in the models since it depends on the previous vertices and also on the model whether a vertex is forced to be accepted or not.
$\mathcal{D}$*: If the penalty of the next vertex is less then* $1/\varphi$ *and the vertex is not forced to be accepted then reject it. Otherwise color the first accepted vertex by color 1, the second one by color 2 and the further accepted vertices by algorithm* FF.

Theorem 5 *Algorithm* $\mathcal{D}$ *is* $2+(n-2)/\varphi$*-competitive in both trace coloring models (proper and cf), where* n *is the number of vertices.*

Proof. Consider an input hypergraph by $H_<$. Again we have three cases.
Case 1. First suppose that the optimal algorithm uses at least two colors to color its accepted subhypergraph. In this case $\mathsf{opt}(H_<, p) \geq 2$. On the other hand $\mathsf{cost}_{\mathcal{D}}(H_<, p) \leq n$ is obviously valid since the algorithm pays less than 1 penalty for the rejected vertices and uses at most one color for the accepted ones. Therefore in this case the theorem follows by $\varphi < 2$.
Case 2. Now suppose that the optimal algorithm uses one color to color its accepted subhypergraph. Denote the set of its rejected vertices by R_{OPT}. Then the optimal cost is $1 + p(R_{\mathsf{OPT}})$.

If $\mathcal{D}$ rejects all of the vertices from R_{OPT} then its accepted vertices are colored by at most 2 colors. Thus the cost of the algorithm is at most $2+(n-2)/\varphi$ and the results follows since the optimal cost is at least 1.

Suppose R_{OPT} contains some vertex which is accepted by $\mathcal{D}$. The first such vertex is not forced to be accepted by $\mathcal{D}$, since otherwise the optimal algorithm could not color its accepted vertices by one color. Thus its penalty is at least $1/\varphi$. This yields that $\mathsf{opt}(H_<, p) \geq 1 + 1/\varphi$ and by $\mathsf{cost}_{\mathcal{D}}(H_<, p) \leq n$ we obtain that

$$\frac{\mathsf{cost}_{\mathcal{D}}(H_<, p)}{\mathsf{opt}(H_<, p)} \leq \frac{n}{1+1/\varphi} = \frac{n}{\varphi} \leq 2 + (n-2)/\varphi.$$

Case 3. Finally, suppose that the optimal algorithm uses 0 color which means that it rejects all vertices. Then its cost is the sum of the penalties of the vertices. No consider the following two subcases.

First suppose that some forced vertex is colored by $\mathcal{D}$. To have a forced vertex it needs at least two accepted vertices with the same color which means

that it has accepted at least 3 unforced vertices. On the other hand each of these vertices has penalty at least $1/\varphi$. This yields that the total penalty of the vertices thus the optimal cost is at least $3/\varphi > 1 + 1/\varphi$ and we obtain again that

$$\frac{\mathrm{cost}_{\mathcal{B}}(H_<, p)}{\mathrm{opt}(H_<, p)} \leq \frac{n}{1 + 1/\varphi} = \frac{n}{\varphi} \leq 2 + (n-2)/\varphi.$$

Finally suppose that the optimal algorithm uses 0 color and there exists no forced vertex accepted by $\mathcal{D}$. Then let A be the set of the vertices accepted by $\mathcal{D}$ and B be the set of vertices rejected by $\mathcal{D}$. We obtain that

$$\frac{\mathrm{cost}_{\mathcal{B}}(H_<, p)}{\mathrm{opt}(H_<, p)} \leq \frac{|A| + p(B)}{p(A) + p(B)} \leq \frac{|A|}{|A|/\varphi} = \varphi \leq 2 + \frac{n-2}{\varphi}.$$

□

Now we prove that the bound is tight in the sense that no asymptotically better online algorithm exists. We use a similar construction as we did in the case of cf coloring in the full edge model. The lower bound is again true for both the proper and cf colorings.

Theorem 6 *No online algorithm exists which is* $Cn + D$*-competitive in the trace model for proper or cf-coloring with rejection for hypergraphs containing* n *vertices and some constants* $C < 1/\varphi$ *and* D*.*

Proof. Suppose that we have an online algorithm which has better competitive ratio, denote it by $\mathcal{E}$. First we present vertices with penalty $1/\varphi$ and no edge until two of them are not colored by the same color or the number of vertices reaches $n_1 > D/(1/\varphi - C)$. If none of these vertices received the same color then the sequence ends the optimal algorithm colors these vertices by one color and its cost is 1. The online algorithms pays at least $1/\varphi$ for each of them thus the online cost is at least n_1/φ and we obtain a contradiction.

Now suppose that the online algorithm colors two accepted vertices by the same colors. Let these vertices be v_i and v_k where $i < k$. Note that the first phase of the inputs ends by vertex v_k. In this case we continue the sequence with the points $v_{k+1}, \ldots, v_n$ where $n > (D + k/\varphi)/(1/\varphi - C)$. Each such vertex v_q has penalty 0, and each of them ends the edges (v_i, v_k, v_q) and (v_p, v_q) for $p = 1, \ldots, q-1$. Then the first edge forces the acceptance of the vertex (otherwise the remaining edge (v_i, v_k) would not be well-colored). Moreover the other edges ensures that each vertex must receive a new color. Therefore, v_i and v_k have the same color and the other accepted vertices are colored by

different colors. Thus the cost of the online algorithm is $m/\varphi + n - m - 1$, where m is the number of rejected vertices in the first phase. This yields that its cost is at least $(k-2)/\varphi+n-k+1$. On the other hand an optimal algorithm accepts the vertices $v_1, \ldots, v_{k-1}$ colors them by color 1 and rejects the further vertices. Then its cost is $1 + 1/\varphi$. Therefore the ratio of the online and offline costs is at least

$$\frac{(k-2)/\varphi + n - k + 1}{1 + 1/\varphi} > \frac{n}{\varphi} - \frac{k}{\varphi} > Cn + D$$

and the theorem follows. □

Acknowledgements

This work was supported by the European Union and the European Social Fund through project Telemedicina (Grant no.: TÁMOP-4.2.2.A-11/1/KONV-2012-0073). Cs. Imreh was supported by the return fellowship of the Alexander von Humboldt Foundation. J. Nagy-György was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4. A/2-11-1-2012-0001 National Excellence Program.

References

[1] N. Alon, U. Arad, Y.Azar, Independent Sets in Hypergraphs with Applications to Routing Via Fixed Paths, *Proc. 2nd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX99)*, *Lecture Notes in Computer Science* **1671** (1999) 16–27. ⇒8

[2] A. Bar-Noy, P. Cheilaris and S. Smorodinsky, Conflict-free coloring for intervals: from offline to online, *Proc. 18th Annual ACM Parallelism Algorithms Architectures (SPAA06)*, 2006, pp. 128–137. ⇒6

[3] A. Bar-Noy, P. Cheilaris, S. Olonetsky, and S. Smorodinsky, Online conflict-free colorings for hypergraphs, *Proc. 34th International Colloquium on Automata, Languages and Programming (ICALP07)*, *Lecture Notes in Computer Science* **4596** (2007) 219–230. ⇒6

[4] A. Borodin, R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998. ⇒6, 8

[5] K. Chen, A. Fiat, H. Kaplan, M. Levy, J. Matousek, E. Mossel, J. Pach, M. Sharir, S. Smorodinsky, U. Wagner and E. Welzl, Online conflict-free coloring for intervals, *SIAM J. Computing* **36** (2006) 1342–1359. ⇒6

[6] K. Chen, H. Kaplan, M. Sharir, Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles, *ACM Trans. Algorithms* **5** (2009) Article No. 16. ⇒6

[7] L. Epstein, A. Levin, G. J. Woeginger, Graph coloring with rejection, *J. Comput. System Sci.*, **77,** 2 (2011) 439–447. ⇒6

[8] M. M. Halldorsson, Online coloring of hypergraphs, *Inform. Process. Lett.* **110,** 10 (2010) 370–372. ⇒6

[9] Cs. Imreh, J. Nagy-György, Online hypergraph coloring, *Inform. Process. Lett.* **109,** 4 (2008) 23–26. ⇒6

[10] Cs. Imreh, Competitive analysis, in: *Algorithms of Informatics, Vol. 1. Foundations* (ed. A. Iványi), mondAt Kiadó, Budapest, 2007, pp. 395–428. ⇒ 6, 8

[11] B. Keszegh, N. Lemons, D. Pálvölgyi, Online and quasi-online colorings of wedges and intervals, *Proc. 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2013)*, *Lecture Notes in Computer Science* **7741** (2013) 292–306. ⇒6

Received: August 11, 2014 • Revised: April 7, 2015

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 18–30

DOI: 10.1515/ausi-2015-0010

A note on label propagation for semi-supervised learning

Zalán BODÓ
Babeş–Bolyai University
email: zbodo@cs.ubbcluj.ro

Lehel CSATÓ
Babeş–Bolyai University
email: lehel.csato@cs.ubbcluj.ro

Abstract. Semi-supervised learning has become an important and thoroughly studied subdomain of machine learning in the past few years, because gathering large unlabeled data is almost costless, and the costly human labeling process can be minimized by semi-supervision. Label propagation is a transductive semi-supervised learning method that operates on the—most of the time undirected—data graph. It was introduced in [8] and since many variants were proposed. However, the base algorithm has two variants: the first variant presented in [8] and its slightly modified version used afterwards, e.g. in [7]. This paper presents and compares the two algorithms—both theoretically and experimentally—and also tries to make a recommendation which variant to use.

1 Introduction

Label propagation is a *transductive graph-based* method for *semi-supervised* classification. It is transductive because the algorithm can predict the labels of the points included in the unlabeled learning dataset, it does not output an inductive classifier applicable for a new point. However, it is true that using a simple *trick* one can obtain a formula for calculating the label of an unknown point without re-learning [1]. We call it graph-based, because it is interpreted

Computing Classification System 1998: I.1.2, I.2.6
Mathematics Subject Classification 2010: 68W40, 68T10
Key words and phrases: label propagation, semi-supervised learning

as building a graph connecting the data points and assigning weights to these edges according to some similarity measure, and then *propagating the labels from the labeled points towards the unlabeled ones*. It is semi-supervised, since some labeled points are needed—usually a small number of such points, and can have a much larger set of unlabeled ones—which direct the algorithm towards a stable labeling configuration.

The aim of this short paper is to analyze two variants of the label propagation algorithm proposed for semi-supervised learning: the first variant that appeared in [8] and its *sightly* modified version used afterwards. An interesting fact is that there is an unmentioned minor modification in the second variant that alters the problem and produces slightly different outputs. Most of the researchers use the second variant of the basic label propagation algorithm as appeared in [7], without any reference to the discrepancy.

We study the differences between the two methods by analyzing the labels output by the algorithms, as well as the underlying optimization problems, and show this on some benchmark datasets. Other variants of label propagation and related methods that will not be discussed here can be found in [1, ch.11].

Section 2 presents the generic algorithm, while Subsections 2.1 and 2.2 present the above-mentioned variants of it. Section 3 analyzes and compares the different outputs of the two variants, while in Section 4 the optimization problems corresponding to the variants of label propagation are examined. In Section 5 the algorithms are compared experimentally on various datasets and the results of the comparison are discussed.

2 Label propagation

The iterative algorithm of label propagation is shown in Alg. 1. The matrix $\mathbf{Y}$ is an $\mathsf{N} \times \mathsf{k}$ matrix, where $\mathsf{N} = \ell + \mathfrak{u}$ (ℓ denotes the number of labeled and $\mathfrak{u}$ the number of unlabeled points) and k represent the size of the dataset and the number of classes, respectively. Later we will split the dataset into labeled and unlabeled parts, putting the labeled examples at the beginning of the dataset and use the notation $\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{\mathsf{L}} \\ \mathbf{Y}_{\mathsf{U}} \end{bmatrix}$, where $\mathbf{Y}_{\mathsf{L}}$ denotes the known and $\mathbf{Y}_{\mathsf{U}}$ the unknown labels. This is a matrix with rows containing the probabilities that a point belongs to a given class.

The matrix $\mathbf{T}$ is an $\mathsf{N} \times \mathsf{N}$ *transition* matrix realizing the propagation of the labels. The construction of this matrix will be detailed in the following subsections. Step 3 is optional in the algorithm—only the first version requires

Algorithm 1 Label propagation

1: **repeat**
2: $\mathbf{Y} = \mathbf{T}\mathbf{Y}$
3: (Row-normalize $\mathbf{Y}$.)
4: Clamp the labeled data.
5: **until** convergence

it—this is the reason why this operation is put in paranthesis.

Label propagation is not necessarily an iterative method: the output labels of the unlabeled points can be expressed analytically.[1] First we partition the multiplication operation in step 2 of the algorithm:

$$\begin{bmatrix} \mathbf{Y}_L \\ \mathbf{Y}_U \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{LL} & \mathbf{T}_{LU} \\ \mathbf{T}_{UL} & \mathbf{T}_{UU} \end{bmatrix} \begin{bmatrix} \mathbf{Y}_L \\ \mathbf{Y}_U \end{bmatrix}, \tag{1}$$

from which we can express the recursive formula for the unknown labels, that is

$$\mathbf{Y}_U = \mathbf{T}_{UL}\mathbf{Y}_L + \mathbf{T}_{UU}\mathbf{Y}_U.$$

The labels output by label propagation then can be expressed as

$$\mathbf{Y}_U = (\mathbf{I} - \mathbf{T}_{UU})^{-1} \mathbf{T}_{UL}\mathbf{Y}_L.$$

Obviously, in order to be able to solve the problem, $\mathbf{I} - \mathbf{T}_{UU}$ must be invertible, but we assume this is the case.[2]

2.1 The first variant

The subtle—but unmentioned in the literature and undiscussed—difference between the two variants lies in the construction of the transition matrix $\mathbf{T}$. This matrix is based on the graph built to represent data similarities. First we define the matrix $\mathbf{W}$ containing the similarities. In [8] this is constructed using the Gaussian similarity,

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right), \quad i, j = 1, 2, \ldots, N, \tag{2}$$

[1] As will be shown in Section 4, the label propagation algorithms presented in this paper have corresponding optimization problems, that can be solved in many different ways.

[2] A detailed analysis of the convergence of label propagation can be found in [8].

but other similarities can be used as well, provided that $\mathbf{I} - \mathbf{T}_{UU}$ remains invertible. The similarity matrix can also be sparse—the description of some useful matrix construction techniques can be found in [5].

We also introduce here the diagonal degree matrix defined as

$$\mathbf{D} = \mathrm{diag}(\mathbf{W1}).$$

Based on the similarities we can define the transition probability matrix,

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W},$$

in which P_{ij} contains the probability—based on data similarities—that from point i we transition to point j.

The first variant propagates the labels from the labeled points towards the unlabeled points, thus $\mathbf{T} := \mathbf{P}'$, that is the label is determined by

$$Y_{ij} = P_{1i}Y_{1j} + P_{2i}Y_{2j} + \cdots + P_{Ni}Y_{Nj}.$$

Similarly to the partitioning of $\mathbf{Y}$ into labeled and unlabeled parts, we can also split $\mathbf{W}$, $\mathbf{D}$ and $\mathbf{P}$, thus obtaining

$$\begin{aligned} \mathbf{Y}_U &= \left(\mathbf{I} - \mathbf{P}'_{UU}\right)\mathbf{P}'_{LU}\mathbf{Y}_L \\ &= \mathbf{D}_U\left(\mathbf{D}_U - \mathbf{W}_{UU}\right)^{-1}\mathbf{W}_{UL}\mathbf{D}_L^{-1}\mathbf{Y}_L. \end{aligned} \tag{3}$$

In this case re-normalization of the rows of $\mathbf{Y}$ is needed in the iterative algorithm, because

$$\begin{aligned} \sum_{j=1}^{k} Y_{ij} &= P_{1i}\sum_{j=1}^{k} Y_{1j} + \cdots + P_{Ni}\sum_{j=1}^{k} Y_{Nj} \\ &= P_{1i} + P_{2i} + \cdots + P_{Ni} \end{aligned}$$

does not sum to one. However, steps 2 and 3 of Alg. 1 can be combined into $\mathbf{Y} = \overline{\mathbf{T}}\mathbf{Y}$, where $\overline{\mathbf{T}}$ is the row-normalized matrix of $\mathbf{T}$, $\overline{T}_{ij} = T_{ij}/\sum_{k=1}^{N} T_{ik}$, $\forall i, j$. Thus, it is sufficient to perform row-normalization once, right before starting to propagate the labels.

2.2 The second variant

In the second variant the labels are propagated "backwards", that is $\mathbf{T} := \mathbf{P}$ is used for the transition matrix, resulting in the following label determination:

$$Y_{ij} = P_{i1}Y_{1j} + P_{i2}Y_{2j} + \cdots + P_{iN}Y_{Nj}.$$

In this case we can say that the label of a point is defined as the convex combination of its forward neighbors' labels. The analytic formula for calculating the labels becomes

$$\begin{aligned} \mathbf{Y}_U &= (\mathbf{I} - \mathbf{P}_{UU})^{-1} \mathbf{P}_{UL} \mathbf{Y}_L \\ &= (\mathbf{D}_U - \mathbf{W}_{UU})^{-1} \mathbf{W}_{UL} \mathbf{Y}_L. \end{aligned} \tag{4}$$

This version of label propagation appeared in [7] and in papers published afterwards, without mentioning the minor modification to the original algorithm. In this variant re-normalization is not needed, since the rows of $\mathbf{Y}$ sum to one:

$$\begin{aligned} \sum_{j=1}^{k} Y_{ij} &= P_{i1} \sum_{j=1}^{k} Y_{1j} + \cdots + P_{iN} \sum_{j=1}^{k} Y_{Nj} \\ &= P_{i1} + P_{i2} + \cdots + P_{iN} = 1. \end{aligned}$$

All the formulae used in label propagation can be rewritten using the *graph Laplacian* [2, 5], a central concept of graph-based learning methods, defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

The Laplacian possesses some interesting and advantageous properties [5], and will be used in Section 4 mostly to simplify the expressions.

3 Analysis of the outputs

Let us denote the $u \times \ell$ matrix $(\mathbf{D}_U - \mathbf{W}_{UU})^{-1} \mathbf{W}_{UL}$ appearing in both analytical formulae by $\mathbf{A}$, which is a matrix with stochastic vectors in its rows. The matrix $\mathbf{A}$ equals $(\mathbf{I} - \mathbf{P}_{UU})^{-1} \mathbf{P}_{UL}$ and we will use this to prove our previous statement (i.e. stochastic rows property). Using this notation we can write the recursive formula of the two methods as

$$\mathbf{Y}_U = \mathbf{D}_U \mathbf{A} \mathbf{D}_L^{-1} \mathbf{Y}_L \tag{5}$$

$$\mathbf{Y}_U = \mathbf{A} \mathbf{Y}_L. \tag{6}$$

We can prove that the rows of $\mathbf{A}$ are stochastic in two steps: (i) $\sum_{j=1}^{\ell} A_{ij} = 1, i = 1, 2, \ldots, u$, (ii) $A_{ij} \geq 0, i = 1, 2, \ldots, u, j = 1, 2, \ldots, \ell$. For the first property we use eq. (6) and check the sum of the i-th row of $\mathbf{Y}_U$, for which it is easy to see that

$$\begin{aligned} \sum_{j=1}^{\ell} (\mathbf{Y}_U)_{ij} &= A_{i1} \sum_{j=1}^{k} Y_{1j} + \cdots + A_{i\ell} \sum_{j=1}^{k} Y_{\ell j} \\ &= A_{i1} + A_{i2} + \cdots + A_{i\ell}, \end{aligned}$$

and since this is the second variant, we know that the rows of $\mathbf{Y}$ sum to one, therefore the rows of $\mathbf{A}$ sum to one as well.

For the second property we consider the definition of $\mathbf{A}$ as $(\mathbf{I} - \mathbf{P}_{uu})^{-1}\mathbf{P}_{ul}$ and use the Neumann series to rewrite it as

$$\mathbf{A} = \sum_{i=0}^{\infty} \mathbf{P}_{uu}^{i}\mathbf{P}_{ul}. \tag{7}$$

Since both $\mathbf{P}_{uu}$ and $\mathbf{P}_{ul}$ contain only nonnegative values, $\mathbf{A}$ also contains only nonnegatives.

From (7) the value A_{ij} can be interpreted as the probability that the first labeled node of a random walk starting at unlabeled node i is j.

In the first variant we can leave out the multiplication with $\mathbf{D}_u$, observing that it does not influence the result. Therefore we are left with the following two *very similar* formulae:

$$\mathbf{Y}_u = \mathbf{A}\mathbf{D}_l^{-1}\mathbf{Y}_l \tag{8}$$

$$\mathbf{Y}_u = \mathbf{A}\mathbf{Y}_l. \tag{9}$$

4 Analysis of the optimization problems

The outputs of the presented methods can be viewed as solutions of specific (semi-supervised) optimization problems. In this section we present and analyze the corresponding problems. We start with the second variant, because the optimization problem of the first label propagation variant can be viewed as a special case of the second variant's minimization problem.

Statement 1 *The output labels (4) in the second variant of label propagation are also a solution of the following optimization problem:*

$$\min_{\mathbf{y}_i, i=\ell+1,\ldots,N} \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2, \tag{10}$$

where $\mathbf{y}_i$ *denotes the* i*-th row of* $\mathbf{Y}$*, that is the probabilistic vector assigned to the* i*-th data point.*

Proof. The minimizable expression can be written as $\operatorname{tr}(\mathbf{Y}\mathbf{Y}'\mathbf{L}) = \operatorname{tr}(\mathbf{Y}'\mathbf{L}\mathbf{Y})$, because

$$\begin{aligned}\frac{1}{2}\sum_{i,j=1}^{N} W_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2 &= \frac{1}{2}\sum_i \mathbf{y}_i'\mathbf{y}_i \sum_j W_{ij} + \frac{1}{2}\sum_j \mathbf{y}_j'\mathbf{y}_j \sum_i W_{ij} - \sum_{i,j} W_{ij}\mathbf{y}_i'\mathbf{y}_j \\ &= \sum_{i,j} \mathbf{y}_i'\mathbf{y}_i D_{ii} - \sum_{i,j} W_{ij}\mathbf{y}_i'\mathbf{y}_j = \operatorname{tr}(\mathbf{Y}'\mathbf{D}\mathbf{Y}) - \operatorname{tr}(\mathbf{Y}'\mathbf{W}\mathbf{Y}) \\ &= \operatorname{tr}(\mathbf{Y}'\mathbf{L}\mathbf{Y}).\end{aligned}$$

Hence we can rewrite the optimization problem (10) in the more compact form of

$$\min_{\mathbf{Y}_U} \operatorname{tr}(\mathbf{Y}'\mathbf{L}\mathbf{Y}). \tag{11}$$

The $\mathbf{Y}_U$ that minimizes the above expression can be found by taking the derivative of the trace with respect to $\mathbf{Y}_U$ and setting it equal to zero. First we partition $\mathbf{L}$ and $\mathbf{Y}$ into labeled and unlabeled blocks, similarly to the matrices in (1), and expand our formula

$$\mathbf{Y}'\mathbf{L}\mathbf{Y} = \mathbf{Y}_L'\mathbf{L}_{LL}\mathbf{Y}_L + \mathbf{Y}_U'\mathbf{L}_{UL}\mathbf{Y}_L + \mathbf{Y}_L'\mathbf{L}_{LU}\mathbf{Y}_U + \mathbf{Y}_U'\mathbf{L}_{UU}\mathbf{Y}_U.$$

Applying the trace operator we get

$$\operatorname{tr}(\mathbf{Y}'\mathbf{L}\mathbf{Y}) = \operatorname{tr}(\mathbf{Y}_L'\mathbf{L}_{LL}\mathbf{Y}_L + 2\mathbf{Y}_U'\mathbf{L}_{UL}\mathbf{Y}_L + \mathbf{Y}_U'\mathbf{L}_{UU}\mathbf{Y}_U).$$

Then we take the derivative of the trace with respect to $\mathbf{Y}_U$ and set it equal to zero, thus obtaining

$$\mathbf{Y}_U = -\mathbf{L}_{UU}^{-1}\mathbf{L}_{UL}\mathbf{Y}_L = (\mathbf{D}_U - \mathbf{W}_{UU})^{-1}\mathbf{W}_{UL}\mathbf{Y}_L. \tag{12}$$

as in (4).[3] □

Statement 2 *The output labels (3) of the first variant of label propagation minimize the following expression:*

$$\min_{\mathbf{y}_i, i=\ell+1,\dots,N} \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}\left\|\frac{\mathbf{y}_i}{D_{ii}} - \frac{\mathbf{y}_j}{D_{jj}}\right\|^2, \tag{13}$$

where D_{ii} *is the* i*-th diagonal element of* $\mathbf{D}$.

[3] In the derivation we used the following properties of the trace [4]: $\operatorname{tr}(\mathbf{A}+\mathbf{B}) = \operatorname{tr}(\mathbf{A}) + \operatorname{tr}(\mathbf{B})$, $\operatorname{tr}(\mathbf{A}') = \operatorname{tr}(\mathbf{A})$, $\frac{\partial \operatorname{tr}(\mathbf{X}'\mathbf{A})}{\partial \mathbf{X}} = \mathbf{A}$, $\frac{\partial \operatorname{tr}(\mathbf{X}'\mathbf{A}\mathbf{X})}{\partial \mathbf{X}} = (\mathbf{A}+\mathbf{A}')\mathbf{X}$.

Proof. By using the substitution $\mathbf{z}_i := \mathbf{y}_i / D_{ii}$ (or $\mathbf{Z} := \mathbf{D}^{-1}\mathbf{Y}$) we arrive to the following optimization problem

$$\min_{\mathbf{y}_i, i=\ell+1,\ldots,N} \frac{1}{2} \sum_{i,j=1}^{N} W_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2 .$$

Similarly to our previous proof, we can write this as $\mathrm{tr}(\mathbf{Z}'\mathbf{L}\mathbf{Z})$, which equals $\mathrm{tr}(\mathbf{Y}'\mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-1}\mathbf{Y})$, where for simplicity we use the substitution $\mathbf{G} := \mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-1}$. Thus our optimization problem becomes

$$\min_{\mathbf{Y}_u} \mathrm{tr}(\mathbf{Y}'\mathbf{G}\mathbf{Y}).$$

Apart from the middle square matrix, this problem is identical to (11), therefore we can apply the same derivation. Using the results from (12) and substituting $\mathbf{L}$ back we obtain

$$\mathbf{Y}_u = -\mathbf{G}_{uu}^{-1}\mathbf{G}_{ul}\mathbf{Y}_l = \mathbf{D}_u(\mathbf{D}_u - \mathbf{W}_{uu})^{-1}\mathbf{W}_{ul}\mathbf{D}_l^{-1}\mathbf{Y}_l,$$

as in (3). □

The optimization problem in (10) can be explained as follows: we want to determine the stochastic vectors $\mathbf{y}_i$, belonging to the unlabeled points, so that to minimize the distance between these class *membership* vectors depending on, i.e. weighted by the similarities of the neighboring points. The first variant is a *normalized* version of the second: here, instead of $\mathbf{L}$ we use $\mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-1}$, a normalized graph Laplacian.[4] By dividing $\mathbf{y}_i$ by the degree of the point a greater weight is assigned to points having *fewer* or *distant* neighbors. This, as will be shown in the experiments, can yield a more balanced solution.

5 Experimental results and discussion

In the experiments we used 7 benchmark datasets from [1].[5] The main properties of these sets are summarized in Table 1. Every set has 2×12 splits: 12 random splits with 10 labeled points and 12 splits with 100 labeled points. In our experiments we used only the first split of the datasets with 10 labeled data.

[4] This Laplacian is the normalized version of the symmetric normalized Laplacian from [5].

[5] The datasets can be downloaded from `http://olivier.chapelle.cc/ssl-book/benchmarks.html`.

Dataset	Classes	Dimension	Points	Note
g241c	2	241	1500	artificial
g241n	2	241	1500	artificial
Digit1	2	241	1500	artificial
USPS	2	241	1500	imbalanced
$COIL_2$	2	241	1500	
BCI	2	117	400	
Text	2	11 960	1500	sparse discrete

Table 1: Properties of the datasets used in the experiments.

Dataset	σ	E	ΔE	$\Delta\mathbf{Y}_U$	Iterations	
g241c	5.8845	0.5013	0	5.1548	1219	1066
g241n	5.8914	0.5020	0	1.1718	1191	1133
Digit1	0.3941	0.2409	0	0.0334	7235	7270
USPS*	0.9	0.1906	0	0.7705	1	1
$COIL_2$*	400	0.4993	0	$3.12 \cdot 10^{-13}$	932	839
BCI	1.7296	0.5333	0	0.0378	3060	3083
Text*	1.3	0.4987	0	0.0045	780	781

Table 2: Experimental results obtained using Gaussian similarity with the indicated parameter: E – error, ΔE – error difference, $\Delta\mathbf{Y}_U$ – norm of the difference vector of the outputs.

Besides these we experimented with other two sets for visually demonstrating the difference between the analyzed methods. The first of these is the *2moons* dataset containing 2 labeled and 383 unlabeled points, while the second *simple* dataset contains 2 labeled and 8 unlabeled points.[6]

The results obtained are shown in Table 2. In all the experiments we used the Gaussian similarity (2) where the parameter was set using the procedure described in [8]. The minimum spanning tree of the data was constructed using Kruskal's algorithm [3]. The process of building the tree, i.e. connecting the points proceeds until the components being connected contain opposite labels; the length of this peculiar edge is denoted by d_0. Then—following the three-sigma or $68-95-99.7$ rule of the normal distribution [6]—the σ parameter of the Gaussian similarity is set to $d_0/3$. In this way it is expected that the "local propagation is mostly within classes" [8]. In four cases this method provided

[6]The datasets can be downloaded from `http://www.cs.ubbcluj.ro/~zbodo/datasets.html`.

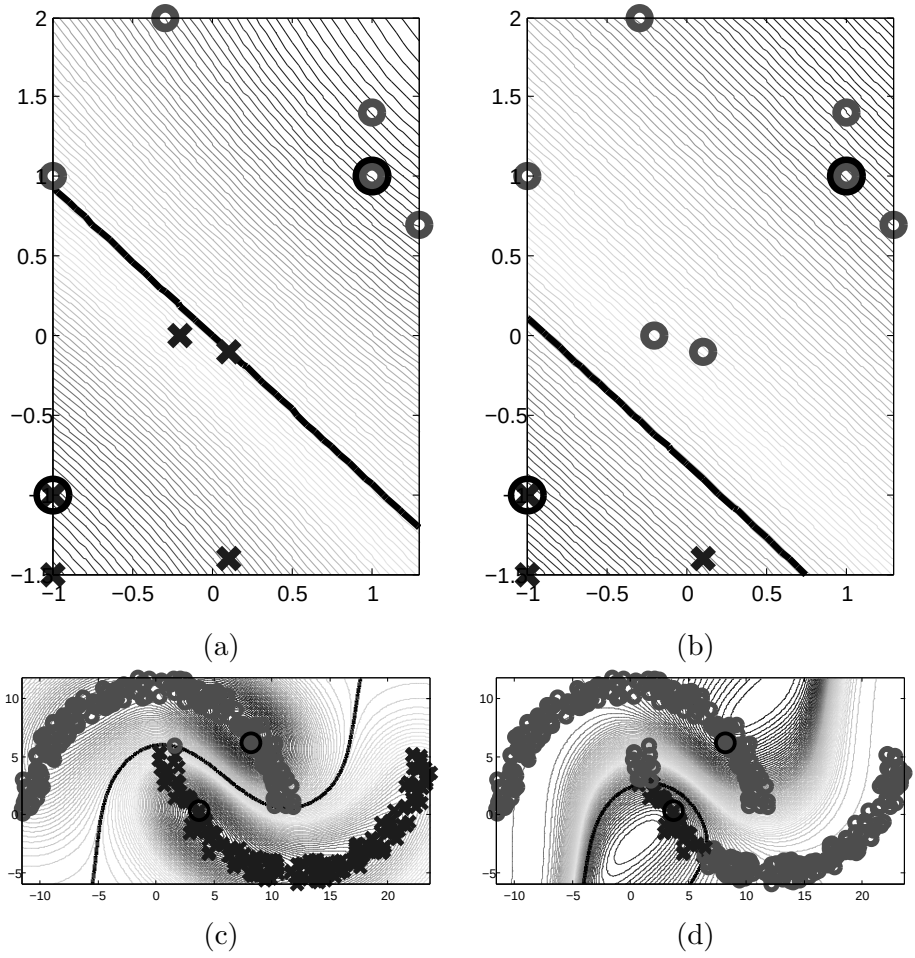

Figure 1: The output of (the first and second variant of) label propagation for the (a), (b) *simple* and (c), (d) *2moons* datasets.

acceptable values, but for the remaining sets (marked with a star in Table 2) resulted in ill-conditioned (nearly singular) ($\mathbf{D}_{\mathsf{U}} - \mathbf{W}_{\mathsf{UU}}$) matrices. For these datasets the indicated parameters were set by investigating the histogram of the kernel values.

We list four values: (i) the error, (ii) the error difference $\Delta\mathsf{E}$, that is $\Delta\mathsf{E} = |\mathsf{E}_1 - \mathsf{E}_2|$, where E_i is the error obtained by the i-th method, (iii) the Frobenius norm of the difference vector of the outputs, $\Delta\mathbf{Y}_{\mathsf{U}} = \left\|\mathbf{Y}^1_{\mathsf{U}} - \mathbf{Y}^2_{\mathsf{U}}\right\|_{\mathsf{F}}$ and (iv) the number of iterations. The first three of these are calculated using the analytical

formulae (3) and (4). In the last two columns of the table we show the number of iterations required for the iterative algorithms to converge; convergence was reached when the Frobenius norm of the difference matrix obtained from two consecutive steps did not exceed 10^{-3}. The initial matrix of $\mathbf{Y}_U$ was set to $\begin{bmatrix}\mathbf{1} & \mathbf{0}\end{bmatrix}$.[7]

In the contour plots of Figure 1 the outputs of label propagation are shown for two datasets: (a) and (b) show the results of the first and second variant of label propagation for the *simple* dataset, while (c) and (d) present the assigned labels for the *2moons* dataset. In both cases the Gaussian similarity was used with parameter $\sigma = 3$. The encircled points denote the labeled data and the thicker black curves show the decision boundaries of the classifier.[8] These were determined using the following methodology. Considering the optimization problems (10) and (13) one can determine the label of a newly arrived point by taking the derivative of the *new* objective functions with respect to y (y denoting the new, unknown label of the new point $\mathbf{x}$):

$$C_1 + \sum_{i=1}^{N} w(\mathbf{x}, \mathbf{x}_i) \left(\frac{y}{d} - \frac{y_i}{D_{ii}} \right)^2$$

$$C_2 + \sum_{i=1}^{N} w(\mathbf{x}, \mathbf{x}_i)(y - y_i)^2,$$

where C_1 and C_2 denote the unchanged parts of the objective functions and $w(\mathbf{x}, \mathbf{x}_i)$ is the similarity of points $\mathbf{x}$ and $\mathbf{x}_i$. Setting the derivatives equal to zero and expressing the label we obtain

$$y = \sum_{i=1}^{N} \frac{w(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{N} W_{ij}} y_i$$

$$y = \sum_{i=1}^{N} \frac{w(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{N} w(\mathbf{x}, \mathbf{x}_j)} y_i$$

for the two variants. For the labels of the unlabeled points (y_i) in the dataset the labels given by the algorithms were used in the above formulae (not the *true* labels).

[7] We also experimented with *class mass normalization* [9] that uses the prior class distribution to influence the predictions, but no significant differences were observed in the results, therefore these results are not divulged here.

[8] We used here the binary version of label propagation, where $\mathbf{Y}$ is an $N \times 1$ vector, $\mathbf{Y}_L \in \{-1, +1\}^{\ell}$; $y_i \geq 0$ denotes a positive class assignment and $y_i < 0$ a negative label.

Examining the $\Delta\mathsf{E}$, $\Delta\mathbf{Y}_{\mathsf{U}}$ values and the iteration counts in Table 2 we cannot notice considerable differences. The error values show the inadequacy of the learning algorithm or its parameters.[9] We obtained seemingly acceptable error rates for two datasets, but the only set where label propagation performed well was Digit1, taking into account the imbalanced class distribution in the USPS dataset. In order to somehow experimentally compare the two variants we decided to use some toy datasets where the results can be easily visualized. Comparing the results in Figure 1 one can see that the normalization included in the first variant resulted in more balanced (and correct) solutions. We saw that (8) and (9) only differ by the inverse of the diagonal degree matrix $\mathbf{D}_{\mathsf{L}}$, a neglectable computational load from a complexity point of view. The iterative algorithms differ in no steps, since in the first variant it is sufficient to perform row-normalization only once, as discussed in Section 2.1. Normalizing the labels by the points' degree, however, can result in a more natural, more balanced labeling. Therefore, concluding the theoretical analysis and the experiments performed, we recommend to prefer the first variant of label propagation over the second variant, where possible.

Acknowledgement

The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.

References

[1] O. Chapelle, B. Schölkopf, A. Zien, *Semi-Supervised Learning*, The MIT Press, Cambridge, 2006. ⇒18, 19, 25

[2] F. Chung, *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*, AMS, Philadelphia, 1997. ⇒22

[3] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, **7,** 1 (1956) 48–50. ⇒26

[4] H. Lütkepohl, *Handbook of Matrices*, John Wiley & Sons, Chichester, 1996. ⇒24

[5] U. von Luxburg, A tutorial on spectral clustering, *Stat. Comput.*, **17**, 4 (2007) 395–416. ⇒21, 22, 25

[6] M. W. Trosset, *An Introduction to Statistical Inference and Its Applications with R*, Chapman & Hall/CRC Texts in Statistical Science, CRC Press, Boca Raton, 2009. ⇒26

[9] By parameters we refer both to the similarity function and its parameters.

[7] X. Zhu, Semi-supervised learning *with graphs*, PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 2005. ⇒18, 19, 22

[8] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. ⇒18, 19, 20, 26

[9] X. Zhu, Z. Ghahramani, J. D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, *Proc. 20th ICML*, 2003. pp. 912–919. ⇒28

Received: August 11, 2014 • Revised: January 23, 2015

DE GRUYTER OPEN
Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 31–57

DOI: 10.1515/ausi-2015-0011

Bin packing with directed stackability conflicts

Attila BÓDIS
University of Szeged
email: bodis.attila@gmail.com

Abstract. The Bin Packing problem is a well-known and highly investigated problem in the computer science: we have n items given with their sizes, and we want to assign them to unit capacity bins such, that we use the minimum number of bins.

In this paper, some generalizations of this problem are considered, where there are some additional stackability constraints defining that certain items can or cannot be packed on each other. The corresponding model in the literature is the Bin Packing Problem with Conflicts (BPPC), where this additional constraint is defined by an undirected conflict graph having edges between items that cannot be assigned to the same bin. However, we show some practical cases, where this conflict is directed, meaning that the items can be assigned to the same bin, but only in a certain order. Two new models are introduced for this problem: Bin Packing Problem with Hanoi Conflicts (BPPHC) and Bin Packing Problem with Directed Conflicts (BPPDC). In this work, the connection of the three conflict models is examined in detail.

We have investigated the complexity of the new models, mainly the BPPHC model, in the special case where each item have the same size. We also considered two cases depending on whether re-ordering the items is allowed or not.

We show that for the online version of the BPPHC model with unit size items, every Any-Fit algorithm gives not better than $\frac{3}{2}$-competitive,

Computing Classification System 1998: F.2.2
Mathematics Subject Classification 2010: 68R05
Key words and phrases: Bin Packing Problem, conflicts, directed conflicts, Hanoi conflicts, unit size items, dynamic programming

when it is forbidden for the optimum to re-order the items, even if only 2 stackability classes, called Hanoi classes, are applied. This lower bound is generalized for arbitrary number of Hanoi classes. However, we also prove, that asymptotically the First-Fit algorithm is 1-competitive for this case.

Finally, we introduce an algorithm for the offline version of the BP-PHC model with unit size items, which has polynomial time complexity, if the number of the Hanoi classes and the capacity of the bins are constant.

1 Introduction

The Bin Packing Problem is one of the most known and investigated fields of the computer science, probably because it has several practical applications such as filling up boxes with certain products, loading trucks, etc. The problem is that we have n items given with their sizes, and we want to assign them to unit capacity bins so, that we use the minimum number of bins.

More formally, we have a set $N = \{1, 2, \ldots, n\}$ of items, each item i has a size s_i, the bins have a capacity c, and we want to assign each item to one bin such that the total size of items in each bin is not exceeding c, and we use the minimum number of bins.

There are several variants of this problem in the literature including multi-dimensional cases, fragile objects, class-constrained items, coloured items, etc. We will summarize these models in Section 2.

In this work, we investigate a variant of the Bin Packing Problem, where additional constraints are occurred because of practical directed stackability conflicts, like some items are fragile and others are too heavy to pack them on each other. This kind of conflicts are quite common in industrial applications, especially in logistical ones.

After a short summary of the relevant existing variants of the Bin Packing Problem in Section 2, two new models are introduced for the mentioned directed stackability conflicts in Section 3. Then, the new models are compared with the corresponding model from the literature in Section 4.

We examined the complexity of a special case of the new models, where each item has unit size. This case is described, and the complexities are investigated for the two new models in Section 5.

2 Previous results

Unfortunately, in the real world applications the bin packing problem, just like a lot of computational problems, rarely happens in a pure way. Usually, there are additional constraints permitting or forbidding the assignment of specific items to specific bins based on the content of the bins, or determining the order of items in the bins. Various models are published for the Bin Packing Problem handling different kind of additional constraints mostly inspired by some kind of practical application. In this section, we will overview some of the existing models, especially the ones that are created for a similar approach to the current work.

First of all, we recall some definitions in connection with the approximation algorithms. For an algorithm $\mathcal{A}$, the solution of $\mathcal{A}$ for a given input X is denoted by $\mathcal{A}(\mathsf{X})$, and the optimal offline solution is usually denoted by $\mathsf{OPT}(\mathsf{X})$. The approximation ratio (called also competitive ratio in online cases) for a minimization problem is the minimal C value such that $\mathcal{A}(\mathsf{X}) \leq \mathsf{C} \cdot \mathsf{OPT}(\mathsf{X})$ is true for any X input. The asymptotic approximation (or competitive) ratio is defined similarly, but with $\mathsf{OPT}(\mathsf{X})$ approaching infinity.

Our first discussed generalization is the Bin Packing Problem with Fragile Objects, where each item have a fragility value in addition to its size, and the corresponding constraint is that the sum of the item sizes in a bin cannot exceed the fragility of any item in that bin. This problem is studied by for example Bansal et al. [2] and Clautiaux et al. [4]. The idea of handling the fragile objects is similar to the practical applications inspiring the current work, but we believe this model is quite difficult to extend to more than only one special object type, meaning the fragile ones.

In the Class Constrained Bin Packing problem, proposed by Shachnai and Tamir [22, 23], we have an additional parameter for every items defining the class of that item, and each bin has a storage capacity beside the load capacity, meaning that the number of different classes in the bins cannot exceed this storage capacity. In their paper, Shachnai and Tamir introduced a PTAS for the offline version of this problem. The online version was first investigated in a paper of the same authors [21]. Xavier and Miyazawa published results of application of this problem to Video-on-Demand services [26]. Further research on approximation algorithms for special cases of the problem was presented by Epstein et al. [11].

The class-constraint is limiting the number of different items in each bins, but this does not say anything about the order of the items in the bins. The next studied variant is exactly a constraint specifying this order. This is the

Colored Bin Packing problem studied recently in some papers [1, 3, 7]. In this model, the items have a color value, and the additional constraint is that one cannot put two items with the same color successively in the same bin. Böhm et al. [3] showed that the classical algorithms First-Fit, Best-Fit and Worst-Fit are not constant competitive. A special case is investigated by Balogh et al. [1], when there are only two colors, black and white, and they have shown a lower bound about 1.7213 on the asymptotic competitive ratio for any online algorithm.

There is a model which is more similar to the one discussed in this work, than the above ones, this is the Bin Packing Problem with LIB ('Largest In Bottom') constraints, which has been also investigated by several authors [10, 9, 16, 17]. This additional constraint means that one cannot put an item on another one with smaller size. Epstein [10] proved that First-Fit gives not better than 2.5 competitive for the online case, which was improved by Dósa et al. [9] to about 2.1666, and they also mentioned a model of Generalized LIB constraint, where the constraint is defined by an undirected incompatibility graph based on the sizes of the items, and adjacent items cannot be packed into the same bin.

The Bin Packing Problem with Conflicts (BPPC) model, investigated by several authors, like Jansen and Öhring [14], Jansen [13], Sadykov and Vanderbeck [20] etc., is very similar to the Generalized LIB constraint in the sense that both models use a conflict graph $G = (V, E)$, where E is the set of edges so, that if $(i, j) \in E$, then the items i and j cannot be packed into the same bin.

The Bin Packing Problem with Conflicts model is very general as basically any kind of graph can be used as a valid conflict graph. There are papers also about special conflict graphs, for example McCloskey and Shankar published results for the case of clique-graphs [19], Jansen and Öhring [14], and also Epstein and Levin [12] studied perfect conflict graphs, and bipartite graphs. Khanafer et al. [15] investigated the two-dimensional variant of this problem.

Finally, our contribution to the field of constrained bin packing is that, in this work, we will introduce a new stackability constraint considering the order of the items, and we will generalize the BPPC model with directed edges.

3 Model definition

In this section, we show some of the real world applications where the undirected conflict graph of the BPPC model is not appropriate, and we define a

new model, which is partly based on the practical solutions, handling these special stackability constraints. Then, we introduce another model, which is generalizing all of the mentioned conflict models.

3.1 Problem definition

As we already mentioned, the computational problems rarely happen purely in the real-world applications, and this is true for the Bin Packing Problem as well. At several fields of the everyday life, we want to assign our items into bins such that some items cannot be packed on each other. This problem is partly handled by the existing BPPC model, but this is not able to handle the case where the conflict is directed. This direction means that the conflict occurs only when the items are packed into the same bin in a given order. For example, there are some fragile products, and we do not want to put them at the bottom of the bins, but we can put them on the top of the bins independently from the content of the bins.

These stackability problems are present in several logistical approaches, like palletisation, when heterogeneous unit loads are expected to be stable enough for transportation, or truck loading, where we want to consider the unloading order. This also occurs in the everyday life, when we want to pack into minimum number of bags at the shop, such that we do not want to put the milk carton on the tomatoes but it can be packed on the potatoes, or even the tomatoes can be placed on the cartons.

Many other examples could be written for the practical applications, where the conflicts are directed. However, this kind of constraints is not handled in the BPPC. Actually, when we have a fixed order of the items, then we can define the edges such that this constraint is taken into consideration, as we will show this in Section 4., but this is not the case for the general problem.

3.2 Bin packing problem with hanoi conflicts (BPPHC)

What is common in the mentioned applications is that there are constraints between the items specifying the order of the items in the bins. These constraints are describing some kind of stackability between the items, which can be defined by precedences or stability expectations, but in either cases it means that some items can or cannot be put on other ones. This definition is very similar to the one appearing at the mathematical game called Tower of Hanoi, in the sense that there are also specific items that cannot be put on other ones. So we will name these restrictions to Hanoi conflicts in this work.

Based on the observation of the real-world approaches, we introduce a generalized bin packing problem handling these stackability constraints. The generalization appears in the description of the items. In our model, each item i is described not only by its size s_i, but also by an additional value determining its Hanoi property.

Definition 1 *The Hanoi property of an item* i *is* $h_i \in H = \{1, \dots, m\}$*, where* H *is the set of possible Hanoi properties.*

Definition 2 *The Hanoi conflict is that one cannot put any item on another one with higher Hanoi property. More formally, if item* i *is assigned to a bin earlier than item* j*, then one can put item* j *into the same bin as item* i *if and only if* $h_i \leq h_j$*.*

Using the definition of the Hanoi conflict, the Bin Packing Problem with Hanoi Conflicts is defined as following. We have a set $N = \{1, 2, \dots, n\}$ of items, each item i has a size s_i and a Hanoi property h_i, the bins have a capacity c, and we want to assign each item to one bin such that the total size of items in each bin is not exceeding c, the Hanoi conflicts are considered, and we use the minimum number of bins.

In order to describe a possible mathematical formulation for this model, we have to introduce some indicator variables, based on the formulation for the standard BPP by Martello and Toth [18]. Let denote $x_{i,j}$ if item j is assigned to the bin i, or not, and let y_i denote if the bin j is used or not. More precisely:

$$x_{i,j} = \begin{cases} 1 & \text{if item } j \text{ is assigned to bin } i, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$y_i = \begin{cases} 1 & \text{if bin } i \text{ is used,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Then the model for the BPPHC can be defined as the following.

$$\text{minimize } z = \sum_{i=1}^{n} y_i \tag{3}$$

$$\text{subject to } \sum_{j=1}^{n} s_i x_{i,j} \leq c y_i \qquad i \in N = \{1, \ldots, n\} \tag{4}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad j \in N \tag{5}$$

$$\left(\sum_{k=1}^{n} x_{k,i} \cdot x_{k,j} \right) (i - j)\,(h_i - h_j) \geq 0 \qquad i \in N,\ j \in N \tag{6}$$

$$y_i \in \{0, 1\} \qquad i \in N \tag{7}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in N, j \in N. \tag{8}$$

This formulation differs from the one for the standard Bin Packing Problem only in the extra constraint (6). The sum is defining that item i and j are assigned to the same bin taking the value of either 1 or 0. The second and the third factors are assuring the consideration of Hanoi conflict between the two items: the sign of them must be the same, meaning that the item assigned later to the bin must have the higher Hanoi property.

We note that this formulation is not linear, because we take the product of variables in the constraint (6). We also mention that this constraint is assuming that item i is the i^{th} item being assigned to a bin, which means that re-ordering the items is not allowed. We will discuss this additional assumption in detail in Subsection 5.2.

3.3 Bin packing problem with directed conflicts (BPPDC)

Obviously, our Bin Packing Problem with Hanoi Conflicts model is somehow connected to the BPPC model, as the Hanoi constraints can be represented as a conflict graph. However, while in the BPPC an edge (i, j) means item i and item j cannot be assigned to the same bin, in the BPPHC model this edge means only that item j cannot be assigned to the same bin as item i later than item i, but it can be assigned to that bin earlier. Thus, in our model the graph is directed according to the Hanoi conflicts.

Since the Hanoi conflicts imply a directed conflict graph, let introduce the Bin Packing Problem with Directed Conflicts (BPPDC) model. We have the set of items $N = \{1, 2, \ldots, n\}$, each item i has a size s_i, the bins have a capacity

c, and we have a directed conflict graph $G = (V, E)$, where E is the set of edges so, that if $(i, j) \in E$, then the item i cannot be packed on top of item j in the same bin. We want to assign each item to one bin considering this conflict graph such that the total size of items in each bin is not exceeding c, and we use the minimum number of bins.

The mathematical formulation of this model can be created using the idea of the BPPHC model.

$$\text{minimize } z = \sum_{i=1}^{n} y_i \tag{9}$$

$$\text{subject to } \sum_{j=1}^{n} s_j x_{i,j} \leq c y_i \qquad i \in N = \{1, \ldots, n\} \tag{10}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad j \in N \tag{11}$$

$$\left(\sum_{k=1}^{n} x_{k,i} \cdot x_{k,j} \right) (i - j) \geq 0 \qquad (i, j) \in E \tag{12}$$

$$y_i \in \{0, 1\} \qquad i \in N \tag{13}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in N, j \in N \tag{14}$$

The notes for the model of the BPPHC are relevant for this model as well: this is not a linear formulation, and re-ordering the items is not allowed.

4 Connection to the bin packing problem with conflicts

In this section, firstly, we will show the connections between the mentioned three conflict models. Then, we will show the importance of the order of the input items, and we will analyse the models considering this order.

The connection of the models is briefly visualized, and also the theorems describing that certain connection is shown, on Figure 1.

4.1 Connections between the three conflict models

In this subsection, we will show that the BPPDC model is the most general among the three models, and that the BPPDC and the BPPHC models are equivalent under certain conditions.

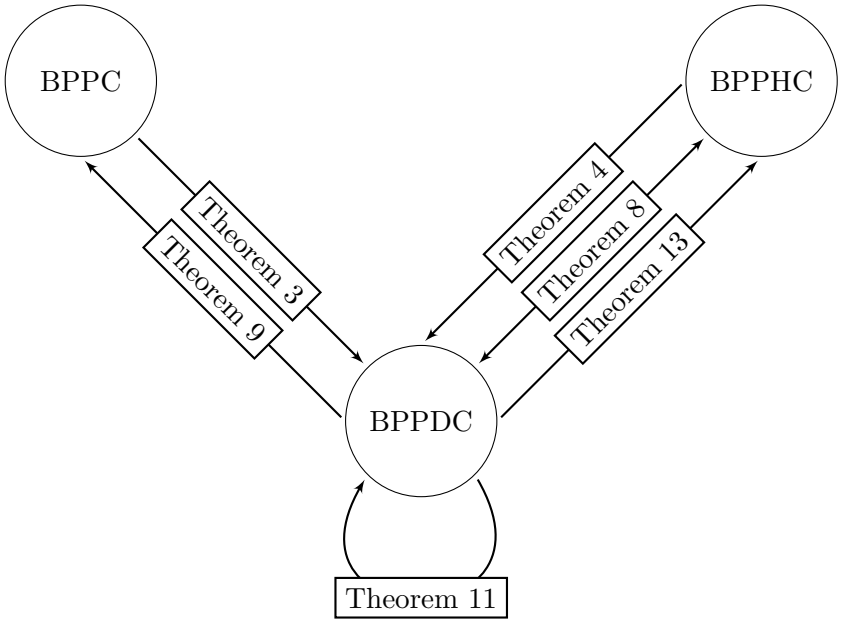

Figure 1: The connection of the three models and the corresponding theorems

First, we present a theorem about the connection of the BPPC and the BPPDC models.

Theorem 3 *The BPPDC model is a generalization of the BPPC model.*

Proof. We show that for any $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ undirected conflict graph, we can create a $\mathsf{G}' = (\mathsf{V}, \mathsf{E}')$ directed conflict graph so, that the corresponding BPPC and BPPDC models are equivalent.

This can be achieved quite simply by generating E' from E in the following way: $\mathsf{E}' = \{(\mathsf{i}, \mathsf{j}), (\mathsf{j}, \mathsf{i}) \mid (\mathsf{i}, \mathsf{j}) \in \mathsf{E}\}$, which means that we take two directed edges for each undirected edge. It can be easily seen that this will result exactly the same problem, because, if item i and j cannot be assigned to the same bin in the BPPC model, then one cannot pack them into the same bin in any order, so we have to take two directed conflicts. □

Now, let investigate the connection of the two new models: the BPPHC and the BPPDC models.

Theorem 4 *The BPPDC model is a generalization of the BPPHC model.*

Proof. We show that we can create a $G = (V, E)$ directed conflict graph so, that each of the Hanoi conflicts are covered by the edges meaning that, two items are in Hanoi conflict if, and only if, there is an edge between them with proper direction in the resulting G graph.

Let the Hanoi constraint be given by item i and j, and $h_i < h_j$, then we take an edge (i, j) into the directed conflict graph. It is trivial, that with this transformation we get an equivalent BPPDC model for any BPPHC input, because the Hanoi conflict of the item i and j represents that the item i cannot be assigned to the same bin as the item j later, than the item j, and the taken edge denotes exactly the same in the BPPDC model. □

Our next theorem will be about the equivalency of the BPPHC and the BPPDC models, but this is realized only under some certain conditions for the type of the directed conflict graph. So before saying that theorem, we have to recall some definitions in connection with the directed graphs.

Definition 5 *If in a directed graph* $G = (V, E)$ *having edges* $(u, v) \in E$ *and* $(v, w) \in E$ *implies also having the edge* $(u, w) \in E$*, then the graph is called transitive.*

We also have to define a special type of the graphs, for which the equivalency will occur.

Definition 6 *A directed graph* $G = (V, E)$ *is called a transitive path, if it is a path with additional edges such that* G *is transitive.*

Definition 7 *A directed graph* $G = (V, E)$ *is called a Hanoi graph, if the graph containing its independent sets as vertices is a transitive path. With other words,* G *is a Hanoi graph if it can be generated from a transitive path* $G' = (V', E')$ *in the following way. We create an independent set of vertices* $V_{i'} \subseteq V$ *for every vertex* $i' \in V'$*, and we take the edges such that* $E = \{(i, j) \ : \ i \in V_{i'},\ j \in V_{j'},\ (i', j') \in E'\}$.

Now, we can present our equivalency theorem.

Theorem 8 *A directed conflict graph is generated by Hanoi conflicts, if and only if, it is a Hanoi graph.*

Proof. First, we prove that the directed conflict graph generated by Hanoi conflicts must be a Hanoi graph.

It is easy to see by the definition of Hanoi conflicts (Definition 2), that if we consider the directed conflict graph $G = (V, E)$, created with the algorithm described in the proof of Theorem 4 from the Hanoi properties themselves, meaning we have exactly one item for every property, then G is a transitive path. Then, based on the definition of the Hanoi graph (Definition 7), we only have to create the independent sets of vertices from the items with equal Hanoi property, which is possible because there is not conflict between these items. So, we get a Hanoi graph $G' = (V', E')$. This means that we can always create a directed conflict graph from the Hanoi conflicts so, that it is a Hanoi graph.

Now, we have to prove that this directed conflict graph is unique. This is proved indirectly. Let assume that there is another directed conflict graph $G'' = (V'', E'')$, which defines exactly the same conflicts as the Hanoi conflicts, and which is different from G'. As the items are the same, $V' = V''$ must occur, so the difference can appear only in the edges, which is possible in two cases:

1. If there is an edge e such that $e \in E'$, but $e \notin E''$, then G'' skips a conflict, which is defined by the Hanoi conflicts, so it is not valid. This is a contradiction.

2. If there is an edge e such that $e \notin E'$, but $e \in E''$, then G'' has an extra conflict, which is not defined by the Hanoi conflicts, so it is also not valid. This is a contradiction, as well.

So we get, that the generated directed conflict graph is unique and it is Hanoi graph indeed.

Secondly, we prove that any $G = (V, E)$ Hanoi graph can be generated by Hanoi conflicts. For this, we show that one can set the Hanoi properties of the items such that, the resulting BPPHC model is equivalent to the original BPPDC model.

We can give an algorithm for this transformation. First, we create the transitive path $G' = (V', E')$ from the independent sets of vertices of G. Actually, we make this inversely as in the definition of the Hanoi graph (Definition 7). Then we have to set the Hanoi properties for every vertices in G' according to the edges. For this, we define for every vertex $i' \in V'$ the set of predecessors $P_{i'}?\{j' \mid (j', i') \in E'\}$. Then for every item i' with $|P_{i'}| = 0$, actually there is only 1 such item because G' is a transitive path, let $h_{i'} = 1$, and for every item i' with $|P_{i'}| > 0$ let $h_{i'} = 1 + \max_{j' \in P_{i'}} h_{j'}$. The resulting Hanoi properties are correct, because any item can be packed on the items without predecessors, and the other items cannot be packed below their predecessors, because of the definition of the Hanoi conflicts (Definition 2). With this algorithm, we set

the Hanoi properties only for the independent sets, but we need them for all items. As the items in the same independent sets are not in conflict, we can set the same Hanoi property for every item in the same independent set. So, we managed to show that G can be generated by Hanoi conflicts. □

So, we have proved, that the equivalency between the BPPHC and the BPPDC models is really existing under the quite strict condition of having a Hanoi graph.

4.2 Importance of the order of the items

By definition, the effect of the Hanoi conflicts are highly dependent on the order of the items. For example, given an item i and j, if $h_i < h_j$ and item i is assigned earlier to a bin, then item j can be assigned to the same bin. However, if item j is arrived earlier, then item i cannot be assigned to the same bin as item j. This means, that one can change if two items are assigned to the same bin by changing only the order of the items. This kind of importance of the input order is not apply in the BPPC model, because the conflict graph defines the conflicts independently from the order.

In this section, we will show that for a fixed order of the items, any directed conflict graph has an equal undirected conflict graph. Also, we will show that with fixed order of items there is a special type of the directed conflict graphs that can be transformed into Hanoi conflicts.

Theorem 9 *Let be given a fixed order of the input items. Then for any directed conflict graph one can find an undirected conflict graph defining exactly the same conflicts.*

Proof. Let (i, j) be an edge in the directed graph, which means that the item i cannot be assigned to the same bin as the item j later, than item j. Considering the given order of the items, there are two cases we have to investigate.

If item i is arrived earlier, then there is no way to assign this to a bin later, than the item j, which is still not arrived. This means, that we can ignore this edge, so there will not be (i, j) edge in the undirected graph.

If item j is arrived earlier, then item i definitely cannot be assigned to the same bin as item j, because of the (i, j) directed edge. Thus, we have to add an (i, j) edge to the undirected graph, and, according to the given order, this has exactly the same meaning as the directed edge. □

Now, we observe the general and the acyclic directed conflict graphs for this case. For this, we need the definition of the directed acyclic graph.

Definition 10 *A directed acyclic graph (DAG) is a directed graph without directed cycles, meaning that there is no way to get back to a vertex through a path following the directed edges* [24].

Theorem 11 *Let be given a fixed order of the input items. Then for any directed conflict graph* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *one can find an acyclic directed conflict graph* $\mathsf{G}' = (\mathsf{V}, \mathsf{E}')$ *defining exactly the same conflicts.*

Proof. Let C be a cycle in G. Considering the first vertex of C in the order, we get that the edge starting from this vertex must go to a vertex arrived later, so we can ignore this edge, because having a conflict with an item not arrived yet is irrelevant.

This means, that, according to the given input order, we can ignore at least one of the edges from any cycle. Thus, there exists an acyclic directed conflict graph for any directed conflict graph. □

Before we can introduce our theorem about under which conditions is it possible to transform a BPPDC model to a BPPHC model for a given order of the input items, we still have to recall the definition of the (weakly) connected directed graphs.

Definition 12 *A directed graph is called (weakly) connected, if removing the directions of the edges and considering them as undirected ones, results to a connected undirected graph.*

Theorem 13 *Let be given a fixed order of the input items and a directed conflict graph. Assuming that the items of each (weakly) connected components of the graph is arrived in continuous blocks, meaning there is no item from another component between any two items of the same component, and every such component is a Hanoi graph, one can set the Hanoi properties for the items such that, according to the given order, the resulting BPPHC model is equivalent to the original BPPDC model.*

Proof. In the second part of the proof of Theorem 8, we have shown that any directed conflict graph, which is a Hanoi graph, can be generated by Hanoi conflicts, and we described an algorithm to set the Hanoi properties based on that graph. To prove the current theorem, we have to expand that algorithm so, that it can handle multiple (weakly) connected components.

Let denote the (weakly) connected components of $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ by $\mathsf{G}_1 = (\mathsf{V}_1, \mathsf{E}_1), \mathsf{G}_2 = (\mathsf{V}_2, \mathsf{E}_2), \ldots, \mathsf{G}_m = (\mathsf{V}_m, \mathsf{E}_m)$. Without loss of generality, we can assume that the components are arriving in the order of their indices. We

note, that we use here the fact that the items of each components are arrived in a continuous block. This is important, because, if the components are merged in the order, then we cannot disjoin them to set the Hanoi properties correctly.

Using this notation, we can define Algorithm 1, which makes the necessary transformation. The $\mathtt{set\_properties\_for\_Hanoi\_graph}(G_i, \mathtt{start})$ function is actually the algorithm in the second part of the proof of Theorem 8 with a little change, that the lowest Hanoi property set for any item of G_i is at least start, and it also updates the value of globalmax by the maximal property appearing in G_i.

Algorithm 1: Algorithm setting the Hanoi properties for the items according to a directed conflict graph and a fixed input order

```
globalmax := 0 ;
foreach G_i in G do                          // in the fixed input order
    start := globalmax;
    set_properties_for_Hanoi_graph(G_i, start);
end
```

Now, we have to prove the correctness of this algorithm. Obviously, the Hanoi properties are set correctly inside each components as shown in the proof of Theorem 8. We have to prove only that the Hanoi properties are appropriate between the components, as well. In this case, appropriateness means that for any two items from two different components, the item arrived later has the greater Hanoi property. More formally, $\forall i \in G_t, \forall j \in \{G \backslash G_t\} : h_i \leq h_j$ if and only if item i is arrived earlier than item j. This is ensured by the usage of the globalmax variable so, that the items of the currently visited components get the greatest Hanoi properties, and we are visiting the components in the fixed order. So, the items of the earlier components get the lower Hanoi properties. □

4.3 Summary of the connections of the models

In this section, we reported our results on investigating the connections between the Bin Packing Problem with Conflicts (BPPC) from the literature, and our two proposed models: the Bin Packing Problem with Hanoi Conflicts (BPPHC) and the Bin Packing Problem with Directed Conflicts (BPPDC). We have shown that the last one is the most general as the other two models can be reduced to this one. Then we proved an equivalency theorem about BP-

PHC and BPPDC stating that a directed conflict graph is generated by Hanoi conflicts, if and only if, it is a Hanoi graph. This theorem actually characterizes the BPPHC model compared to the BPPDC model.

Then we discussed the importance of the order of the items, because, unlike the BPPC model, this is essential for the two new models. We examined possible transformations between the models for fixed order. We pointed out that for a fixed order, a directed conflict graph can be transformed into an undirected conflict graph. Furthermore, we presented an algorithm transforming a 'special type' of BPPDC input into BPPHC input that is this algorithm can set the Hanoi properties for the items based on the directed conflict graph. The 'special type' means here that the (weakly) connected components of the directed conflict graph must arrive in continuous blocks, meaning separately from the other components, in the fixed order, and each of these components must be a Hanoi graph.

5 Variants with unit size items

In this section, we will consider the above described models with a further assumption that each item have the same unit size. This case is also investigated for other variants of the Bin Packing Problem and other packing problems as well. Several authors, like Coffman et al. [5, 6], studied the ordinary Bin Packing Problem with discrete item sizes, which is a similar, but weaker assumption for the sizes. Shachnai and Tamir [21] proposed algorithms for the Class-Constrained Bin Packing Problem with unit sizes.

This case has practical applications, too, mainly in logistics. For example, we have to load truck with equal size pallets and we have to consider the order of unloading, meaning we cannot put some pallets in front of others. Also, this case can occur in several approaches, where different products have boxes with same size and the conflicts are based on some logical conditions such as one item has to be used earlier than others.

For this variant, we slightly modify our models such that $s_i = 1$ for every item i, and $c > 1$ that is the capacity of the bins is higher than 1. We will call the modified versions of the BPPHC and the BPPDC models BPPHCU and BPPDCU models respectively, adding the 'and Unit size items' suffix for the names of the models.

As shown in the Subsection 4.2, the order of the items is crucial, so we will consider two cases based on whether the algorithm can or cannot change the order of the items.

5.1 Ordering is allowed

If one can change the order of the input, then the problems usually becomes easier, and this is the case for our problems, as well. In this, part, we will introduce algorithms for both new models, when it is allowed to change the input order.

5.1.1 BPPHCU with ordering

For the case of Hanoi conflicts, if one can re-order the items, the problem becomes absolutely easy. Actually, in this case, there is no real effect of the Hanoi conflicts for the result, because we can always sort the items per bin such that we get a valid solution. This is concluded in the next theorem, where we define an algorithm based on Next Fit giving optimal solution.

Theorem 14 *The following algorithm gives optimal solution for the BPPHCU model.*

Firstly, we pack the items into bins using the Next-Fit algorithm not considering the Hanoi conflicts at all. Then we sort the items inside each bins separately such that the Hanoi conflicts do not occur, meaning the items with lower Hanoi property will be the earlier in the item-list of each bin.

Proof. Obviously, Next-Fit gives optimal solution for the ordinary Bin Packing Problem with Unit sizes. Then, we have to prove only, that we can sort the items in each bins such that the Hanoi conflicts are avoided. This is also quite trivial, because one can always sort positive integers, that is the Hanoi properties, into non-decreasing order. □

5.1.2 BPPDCU with ordering

The BPPDCU model usually also becomes relatively easy, if we can re-order the items, because we can make a topological order of the items resolving exactly the directed conflicts. However, the topological sort is possible only, if the graph is acyclic, meaning if there are items that cannot be packed into the same bin, independently from their order in the bin, then the problem is still not trivial.

In this work, we consider only the acyclic graphs for this problem, and the next theorem defines an efficient algorithm, similar to the one described in Theorem 14, to find the optimal solution in this case.

Theorem 15 *The following algorithm gives optimal solution for the BPPDCU model.*

Firstly, we pack the items into bins using the Next-Fit algorithm not considering the directed conflict graph at all. Then we sort the items inside each bins separately corresponding to the topological order, meaning the items appearing earlier in the topological order will be the earlier in the item-list of each bin.

Proof. The correctness of this algorithm trivially comes from the proof of Theorem 14 and the definition of the topological order. □

5.2 Ordering is forbidden (BPPHCU)

As we already described in Section 4.2, the order of the items is crucial in connection with the directed conflicts including the Hanoi conflicts as well. We have seen in the previous subsection that the problem becomes kind of easy if we have the ability to re-order the input items. However, this is not possible in several cases. In logistics, this is forbidden usually because of lack of puffer spaces. For example, we have to put the pallets on a truck when it is ready for transport, because we do not have enough space in the warehouse to wait all the pallets. This is similar to the definition of the online problems, but sometimes the full list of pallets is available, so the online and offline variants of the problem is also interesting.

In this subsection, we will first investigate the online case, meaning we get each item one-by-one, and we do not know anything about the later ones. We show a lower-bound on the approximation ratio of the Any-Fit algorithms. Then, we will prove that asymptotically the First-Fit algorithm, modified for Hanoi conflicts, is 1-approximation. Finally, we will show an offline optimal algorithm with polynomial time complexity if the number of different Hanoi properties and the capacity of the bins are considered to be constants.

5.2.1 Online case

In this part, we investigate the online algorithms for the BPPHCU model, when the offline optimum is restricted such that the offline algorithm can neither re-order the items, so it has to pack the items in their incoming order.

First-Fit algorithm is widely investigated for different variants of the Bin Packing Problem. For the standard version, it was proved by Ullman [25] that its asymptotic approximation ratio is 1.7, and Dósa and Sgall [8] proved, that the absolute approximation ration is the same. We have to modify this algorithm to handle the Hanoi conflicts: the current item is assigned to the

first not-filled bin, where the top-item, meaning the last item packed into that bin has a lower or equal Hanoi property.

First-Fit is a member of a group of algorithms, called Any-Fit algorithms defined by a general idea. Any-Fit algorithms always assign the current item to any of the already open, incomplete bins, if it is possible, otherwise they open a new bin for this item.

In the next theorem, we present a lower-bound for the Any-Fit algorithm, when $m = 2$, meaning there are only 2 Hanoi classes.

Theorem 16 *Every Any-Fit algorithm gives not better than $\frac{3}{2}$-approximation for the BPPHCU model with $m = 2$.*

Proof. To prove this theorem, we show an example, where the approximation ratio of the Any-Fit algorithm is $\frac{3}{2}$.

Let $I = [1, 1, 1, 2, 2, 2, 2, 1]$ be the list of input items, where the items are given only with their Hanoi properties because we have unit sizes, and the capacity of the bins is 4.

This is packed by Any-Fit as follows:

$$AF(I) = \{1, 1, 1, 2\}, \{2, 2, 2\}, \{1\}.$$

OPT can pack the items, like following:

$$OPT(I) = \{1, 1, 1, 1\}, \{2, 2, 2, 2\}.$$

We can see, that Any-Fit packed the items into 3 bins, while the optimum can solve that problem instance with only 2 bins. So, the approximation ratio in this case is $\frac{AF(I)}{OPT(I)} = \frac{3}{2}$ □

This lower bound can be generalized to arbitrary number of Hanoi classes using the same structured of input. This result is stated in the next theorem.

Theorem 17 *Let assume m is the number of Hanoi classes, c is the capacity of the bins, and $\lfloor x \rfloor$ denotes the integer part of x. If c is a divisor of m, then every Any-Fit algorithm gives not better than $\frac{\left\lfloor \frac{m}{c} \right\rfloor (c-1) + m}{m}$ -approximation, otherwise they are not better than $\frac{2m - \left\lfloor \frac{m}{c} \right\rfloor - 1}{m}$ -approximation for the BP-PHCU model.*

Proof. To prove this theorem, we show an input structure for arbitrary m and c, where the approximation ratio of the Any-Fit algorithms is exactly the one mentioned in the theorem.

Let I be the list of input items as follows:

$$I = [1, 1, \ldots, 1; ; 2, 2, \ldots, 2; ; \ldots; ; m, m, \ldots, m; ; m, m-1, \ldots, 2, 1] .$$

Where the items are given only with their Hanoi properties because we have unit sizes, and the capacity of the bins is c. In the input above, the semicolons denote a separator of input blocks such that, the number of items in each block, except the last one, is $c-1$.

As the last block contains exactly one item for each Hanoi property, an optimal solution, denoted by $OPT(I)$, is to fill 1 bin for each Hanoi property, because there are exactly c items for each of them, thus $OPT(I) = m$.

Now, let consider how the Any-Fit algorithms work on such an input. As the items can be packed on each other in the order of input until the last block, Any-Fit packs these items on each other until the bin is filled, then open a new bin and do the same, which is basically a simple Next-Fit method. However, when the algorithm reaches the last block, then it has to pack each item into a new bin, so we get a lot of bins with only 1 item.

Firstly, let investigate the blocks containing $c-1$ items. As the capacity of the bins is c, every bin will contain items from the consecutive blocks so, that, when we iterate through the input, the packed bins will always contain one more item from the next block, than the previous bin. This is clearly seen on an example, where $c = 4$. In this case, Any-Fit packs these input-blocks as following:

$$AF(I) = \{1, 1, 1, 2\}, \{2, 2, 3, 3\}, \{3, 4, 4, 4\}, \{5, 5, 5, 6\}, \{6, 6, 7, 7\}, \ldots$$

This means that after c blocks $c-1$ bins, called bin-block, are filled and the next block starts with a new empty bin. Based on this property of Any-Fit, we can divide into 3 groups the packed bins according to which part of the input items are packed into them. The bins that are packed with items from the blocks containing $c-1$ items have 2 groups: the bins that are in a complete bin-block, meaning having exactly $c-1$ bins, and the ones that are in the remainder bin-block. The third group of bins contains the ones being packed with items from the last block. We note, that the first item of the last block can be packed on the items of the previous block, and, in this case, this bin is also in the third group.

So, to determine the bins used by the Any-Fit algorithms, we have to count them in each groups. The number of bins in the first groups is quite trivial considering the already mentioned fact, that after every c blocks $c-1$ bins are filled. As the number of such input blocks is m, the number of bins in the first groups is $\left\lfloor \frac{m}{c} \right\rfloor (c-1)$.

The third group contains exactly m bins, because every item in the last block is placed into a new bin. We note again, that the first item is possibly packed into the same bin as the last item of the previous block, and we count this bin also for the third group.

The second group, that is the remainder bin-block, is a bit more complicated. We have to consider two cases. If c is a divisor of m, then there are not any bin in the remainder block, so the number of bins in the second group is 0. Otherwise, let denote r the number of remained input blocks for this part. Then $r = m - 1 - \left\lfloor \frac{m}{c} \right\rfloor c$, which is similar to the definition of the remainder of the division $\frac{m}{c}$, except we decrease this by one, because the last block is counted in the third group. As $r < c$, the number of bins used to pack these items is exactly r, because we should have at least c blocks to save one bin.

So by summarizing the number of bins in the 3 groups, we get that if c is a divisor of m, then the approximation ratio is the following:

$$\frac{AF(I)}{OPT(I)} = \begin{cases} \dfrac{\left\lfloor \frac{m}{c} \right\rfloor (c-1) + m}{m} & \text{if } c \text{ is a divisor of } m \\ \dfrac{\left\lfloor \frac{m}{c} \right\rfloor (c-1) + m - 1 - \left\lfloor \frac{m}{c} \right\rfloor c + m}{m} & \\ = \dfrac{2m - \left\lfloor \frac{m}{c} \right\rfloor - 1}{m} & \text{otherwise.} \end{cases}$$

□

Although, we have seen that Any-Fit has at least $\frac{3}{2}$-approximation ratio even for $m = 2$, asymptotically the situation is much better for the First-Fit algorithm. Before we show our results about this, firstly we have to introduce an intermediate result about the incomplete bins (the ones that are not filled up totally) during the execution of this algorithm.

Theorem 18 *There are no two incomplete bins having top-items with the same Hanoi property at any time during the execution of the First-Fit algorithm.*

Proof. This is proved by induction on the number of items arrived (let denote this by i, and the Hanoi property of the last item by h_i):

1. If $i = 1$, then we have only one item, and we have to put it into a bin, which is incomplete, but there cannot be any other incomplete bins, so the statement is true.

2. Let assume, we have $i = k$ and the statement is true.

3. If we have $i = k+1$, meaning we get another item, then we have 2 cases:

   (a) We can put the new item into an already incomplete bin. Then, we have to investigate 2 sub-cases:

      i. If there is not any incomplete bin with top-item having Hanoi property h_i, then we put this item into the first incomplete bin with top-item with Hanoi property less, than h_i. Thus, we have only 1 bin with this Hanoi property on the top after assigning this item, so the statement is true.

      ii. If there is already a bin b with top-item having Hanoi property h_i, then we have to prove that the i^{th} item is assigned to this bin by First-Fit. We show this indirectly. Let suppose, that First-Fit packs this item into a bin b' such that $b' \neq b$. This can happen only if b' is earlier, than b, but in this case, the top-item of b would have been assigned to b', which is a contradiction. So the statement is true.

   (b) We cannot put the new item into any of the incomplete bins because of the Hanoi conflicts. In this case, there is not any bin with top-item having Hanoi property h_i, so, when we open a new bin to assign the i^{th} item, then this will be the only bin with such top-item. So the statement is true in this case, too.

□

We note, that, although it might looks like Theorem 18 is true for the optimal solution, this is not the case. This is shown by the following example:

Let $I = [1, 2, 2, 4, 4, 4, 3, 3, 3, 3]$ be the list of input items, where each item is given by its Hanoi property, and the capacity of the bins is 5.

The optimal solution for this instance is:

$$OPT(I) = \{1, 2, 4, 4, 4\}, \{2, 3, 3, 3, 3\}.$$

However, if we want to keep true for every step, that there are no two incomplete bins having top-items with the same Hanoi property, then we get the following:

$$A(I) = \{1, 2, 2, 4, 4\}, \{4\}, \{3, 3, 3, 3\}.$$

Now, using this theorem for First-Fit, we can introduce our main result about the asymptotical approximation ratio of this algorithm.

Theorem 19 *First-Fit algorithm is 1-approximation asymptotically for the BPPHCU model.*

Proof. The proof is based on the idea that First-Fit algorithm uses at most $\mathsf{OPT}(I) + m$ bins, where $\mathsf{OPT}(I)$ is the number of used bins in the optimal solution and m is the number of Hanoi classes. Consequently, we get that $\lim_{|I|\to\infty} \frac{\mathsf{OPT}(I)+m}{\mathsf{OPT}(I)} = 1$. To prove this, we have to consider only the incomplete bins, because these are the ones making difference to the optimum, as even the optimum cannot put more items to the complete bins. As we have seen in Theorem 18, there are no two incomplete bins with top-item having the same Hanoi property, which implies that there are at most as many incomplete bins as Hanoi classes, that is m. Thus, we get that $\mathsf{FF}(I) - \mathsf{OPT}(I) \leq m$, which implies the statement. □

5.2.2 Offline case

As we already mentioned, there are some practical applications, when, despite the fact that all the items are known in advance, we cannot sort them arbitrarily, for example because of lack of space or other resources. In this part, we investigate this case: we have a full list of unit size items with Hanoi conflicts, and we want to assign them into minimum number of bins. We propose an offline algorithm giving optimal solution and we examine its complexity.

In the BPPHCU model, we have finite number of Hanoi properties, thus we can define a finite number of patterns representing the possible loads of the bins. Each pattern p looks like the following:

$$p = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,c} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,c} \end{bmatrix}.$$

Here $p_{i,j}$ is the number of the used bins for the bin-pattern with load j and top-item with Hanoi property i. This implies, that there are $n^{c\cdot m}$ different possible patterns, where n is the number of input items, c is the capacity of each bin and m is the number of Hanoi classes, because there are $c \cdot m$ different bin-patterns, and we can have n used bins from any of them.

Considering these patterns, we can introduce an algorithm, giving optimal solution using a dynamic programming approach represented by the Algorithm 2. Before explaining this method, we have to introduce some notations.

Let P be the set of possible patterns, that is $P \subseteq N^{H\times\{1,2,\ldots,c\}}$. We will generate a matrix $A \subseteq \mathcal{P}(P)^{P\times H}$, giving the set of possible patterns reached from a specific pattern through a transition generated by an item with a specific Hanoi property. This means, that $A[p,h]$ is the set of patterns that can be reached from the pattern p by assigning an item with Hanoi property h to a bin. Furthermore, we will handle a set R_i of possible patterns for each item i in the input, that is $R_i \subseteq \mathcal{P}(P)$.

Now, with these notations we can describe our algorithm finding optimal solution for the sort-restricted BPPHCE model. Firstly, we open the first bin for the first item by adding a pattern to R_1. The added pattern p_{start} is defined as follows:

$$p_{start_{i,j}} = \begin{cases} 1 & \text{if } i = h_1 \text{ and } j = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 2: Dynamic programming algorithm solving the BPPHCU model for fixed order

```
1  R_1.add(p_start) ;                    // initialize with the first item
2  for i := 1 to n − 1 do                    // iterate through the items
3  |  foreach p ∈ R_i do
4  |  |  foreach q ∈ A[p, h_{i+1}] do
5  |  |  |  R_{i+1}.add(q);
6  |  |  |  S[i + 1, q] := p;
7  |  |  end
8  |  end
9  end

   // find the optimum;
10 min := inf;
11 min_p := nil;
12 foreach p ∈ R_n do
13 |  x := sum_{i∈H,j∈{1,2,···,c}} p_{i,j};
14 |  if x < min then
15 |  |  min := x;
16 |  |  min_p := p;
17 |  end
18 end
```

Then, we iterate through the items, and meanwhile we add the possible patterns to R_{i+1}. This is done such that for every item i the elements of $A[p, h_{i+1}]$, that is the possible patterns reached from the pattern p through a transition generated by the item $i+1$, are inserted. Furthermore, the algorithm creates backward pointers for every added pattern to help determine the assignments for the optimal solution. This is handled by the matrix $S \subseteq P^{N \times P}$ containing a pointer to source pattern in R_i.

At the end of this loop, the set R_n contains the possible patterns after every item is assigned, so we only have to count the used items in each patterns and choose the minimal one. The optimal assignment of items into bins can be retrieved by following the pointers of the matrix S.

To determine the running time of this algorithm, we have to find out the sizes of the sets in the matrix A. As each item i can be assigned to any bin having a top-item with Hanoi property less or equal than h_i, and the elements of matrix A contains exactly these successors, we get that for any pattern p and Hanoi property h, $|A[p, h]| = c \cdot h$, because there are c different loads for every top-item, which means, that $c \cdot h$ different patterns can be reached from p through a transition generated by an item with Hanoi property of h. As $h \leq m$, we can say that $c \cdot m$ is an upper-bound on the sizes of the sets in the matrix. Then, we have to specify the size of the R_i sets for every item i, which is clearly upper-bounded by the number of all possible patterns, that is $n^{c \cdot m}$, as shown earlier. So the time complexity of the algorithm is $O\left(n^{c \cdot m+1} \cdot c \cdot m\right)$.

We can see that in this time complexity only c and m occur in exponent, meaning that the running time of the algorithm is polynomial, if we consider c, the capacity of the bins, and m, that is the number of Hanoi classes, as constants, meaning these are independent from the input. This consideration is not uncommon in practical applications, because in several fields fix Hanoi classes are used, such as fragile, or heavy products, so the number of these classes is really independent from the current input items, and the capacity of the containers are also often fix.

6 Conclusion

As conclusion, we can say that we have found some practical applications of the Bin Packing Problem that are not fully handled by the already existing models, so we introduced two relevant variants: the Bin Packing Problem with Hanoi Conflicts (BPPHC) and the Bin Packing Problem with Directed Conflicts (BPPDC).

We deeply investigated the connection of the new models to the Bin Packing Problem with Conflicts (BPPC), which exists in the literature. We pointed out some important connections depending on the type on the conflict graphs. We also examined the importance of the order of the input items. We found that our BPPDC model is the most general one.

Furthermore, we presented results about the complexity of the problems if all the items have unit size. We showed that in this case every Any-Fit algorithm gives not better than $\frac{3}{2}$-approximation for the online version of the Hanoi Conflicts model even for only 2 Hanoi classes, if re-ordering the input is forbidden for the optimum. This lower bound is also generalized for arbitrary number of Hanoi classes. However, we proved that the First-Fit algorithm is asymptotically 1-approximation for this case.

Last, but not least, we proposed an offline algorithm giving optimal solution for the sort-restricted Hanoi Conflicts model with unit size items which has polynomial time complexity, if the capacity and the number of the Hanoi classes are considered as constants.

We believe this work introduced practically important models for the Bin Packing Problem, and further research will be useful to help solving industrial problems.

Acknowledgements

I am very grateful to Csanád Imreh for his sustained help and the useful comments.

References

[1] J. Balogh, J. Békési, Gy. Dósa, L. Epstein, H. Kellerer, Zs. Tuza, Online results for black and white bin packing, *Theory Comput. Syst.*, **56,** 1 (2015) 137–155. ⇒34

[2] N. Bansal, Z. Liu, A. Sankar, Bin-packing with fragile objects and frequency allocation in cellular networks, *Wireless Networks*, **15,** 6 (2009) 821–830. ⇒33

[3] M. Böhm, J. Sgall, P. Vesely, Online colored bin packing, *arXiv:1404.5548 [cs.DS]* (2014) ⇒34

[4] F. Clautiaux, M. Dell'Amico, M. Iori, A. Khanafer, Lower and upper bounds for the Bin Packing Problem with Fragile Objects, *Discrete Appl. Math.*, **163,** 1 (2014) 73–86. ⇒33

[5] Jr., E. G. Coffman, C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, M. Yannakakis, Bin packing with discrete item sizes part I: Perfect packing theorems and the average case behavior of optimal packings, *SIAM J. Discrete Math.*, **13,** 3 (2000) 384–402 ⇒45

[6] Jr., E. G. Coffman, D. S. Johnson, L. A. McGeoch, R. R. Weber, Bin packing with discrete item sizes part II: tight bounds on First Fit, *Random Structures Algorithms*, **10,** 1–2 (1997) 69–101. ⇒45

[7] Gy. Dósa, L. Epstein, Colorful bin packing, *Algorithm Theory – SWAT 2014*, *Lecture Notes in Comput. Sci.*, **8503** (2014) 170–181. ⇒34

[8] Gy. Dósa, J. Sgall, First Fit bin packing: a tight analysis, *30th International Symposium on Theoretical Aspects of Computer Science: STACS*, Dagstuhl, Germany, 2013, pp. 538–549. ⇒47

[9] Gy. Dósa, Zs. Tuza, D. Ye, Bin packing with 'Largest In Bottom' constraint: tighter bounds and generalizations, *J. Comb. Optim.*, **26,** 3 (2013) 416–436. ⇒34

[10] L. Epstein, On online bin packing with LIB Constraints, *Naval Res. Logist.*, **56,** 8 (2009) 780–786. ⇒34

[11] L. Epstein, Cs. Imreh, A. Levin, Class constrained bin packing revisited, *Theoret. Comput. Sci.*, **411,** 34–36 (2010) 3073–3089. ⇒33

[12] L. Epstein, A. Levin, On bin packing with conflicts, *Approximation and Online Algorithms*, *Lecture Notes in Comput. Sci.* **4368** (2007) 160–731. ⇒34

[13] K. Jansen, An approximation scheme for bin packing with conflicts, *J. Comb. Optim.*, **3,** 4 (1999) 363–377. ⇒34

[14] K. Jansen, S. Öhring, Approximation algorithms for time constrained scheduling, *Inform. and Comput.*, **132,** 2 (1997) 85–108. ⇒34

[15] A. Khanafer, F. Clautiaux, E. G. Talbi, Tree-decomposition based heuristics for the two-dimensional bin packing problem with conflicts, *Computers and Operations Research*, **39,** 1 (2012) 54–63. ⇒34

[16] P. Manyem, Uniform sized bin packing and covering: Online version, *Topics in industrial mathematics*, Springer US, 2000. ⇒34

[17] P. Manyem, R. L. Salt, M. S. Visser, Approximation lower bounds in online LIB bin packing and covering, *J. Autom. Lang. Comb.*, **8,** 4 (2003) 663–674 ⇒34

[18] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, 1990. ⇒36

[19] B. McCloskey, A. Shankar, Approaches to bin packing with clique-graph conflicts, *EECS Department, University of California, Berkeley* (2005) ⇒34

[20] R. Sadykov, F. Vanderbeck, Bin packing with conflicts: a generic branch-and-price algorithm, *INFORMS J. Comput.*, **25,** 2 (2013) 244–255. ⇒34

[21] H. Shachnai, T. Tamir, Tight bounds for online class-constrained packing, *Theoret. Comput. Sci.*, **321,** 1 (2004) 103–123. ⇒33, 45

[22] H. Shachnai, T. Tamir, Polynomial time approximation schemes for class-constrained packing problems, *Journal of Scheduling*, **4,** 6 (2001) 313–338. ⇒33

[23] H. Shachnai, T. Tamir, On two class-constrained versions of the multiple knapsack problem, *Algorithmica*, **29,** 3 (2001) 442–467. ⇒33

[24] K. Thulasiraman, M. N. S. Swamy, *5.7 Acyclic Directed Graphs*, *Graphs: Theory and Algorithms*, John Wiley and Sons, 1992. 118. ⇒43

[25] J. D. Ullman, The performance of a memory allocation algorithm., *Princeton University, Department of Electrical Engineering, Computer Science Laboratory*, (1971) ⇒47

[26] E. C. Xavier, F. K. Miyazawa, The class constrained bin packing problem with applications to video-on-demand, *Theoret. Comput. Sci.*, **393,** 1–3 (2008) 240–259. ⇒33

Received: January 24, 2015 • Revised: May 2, 2015

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 58–71

DOI: 10.1515/ausi-2015-0012

Minimization of the Perron eigenvalue of incomplete pairwise comparison matrices by Newton iteration

Kristóf ÁBELE-NAGY
Department of Operations Research and Actuarial Sciences, Corvinus University of Budapest, Hungary
email: kriszo_5@yahoo.de

Abstract. Pairwise comparison matrices are of key importance in multi-attribute decision analysis. A matrix is incomplete if some of the elements are missing. The eigenvector method, to derive the weights of criteria, can be generalized for the incomplete case by using the least inconsistent completion of the matrix. If inconsistency is indexed by CR, defined by Saaty, it leads to the minimization of the Perron eigenvalue. This problem can be transformed to a convex optimization problem. The paper presents a method based on the Newton iteration, univariate and multivariate. Numerical examples are also given.

1 Introduction

When faced with a multi-attribute decision problem, where all alternatives are already evaluated with respect to all relevant criteria, one has to determine the subjective weights of criteria to rank the alternatives. When a decision maker is asked to determine his own subjective weights of criteria, it is often impossible to determine them directly. However, it may be simpler to tell

Computing Classification System 1998: G.1.6
Mathematics Subject Classification 2010: 91B06, 49M15, 90B50
Key words and phrases: incomplete pairwise comparison matrix, Perron eigenvalue, Newton iteration

how many times more important a criterion is compared to another. Ratios are arranged in a positive $n \times n$ matrix $A = [a_{ij}]_{i,j=1,\ldots,n}$, which is called a *pairwise comparison matrix* (PCM), where n is the number of criteria. For a PCM $a_{ij} = \frac{1}{a_{ji}}$ holds for all $i, j = 1, \ldots, n$, thus every element is 1 in the diagonal. A PCM is *consistent*, if the cardinal transitivity property $a_{ik}a_{kj} = a_{ij}$ holds for all $i, j, k = 1, \ldots, n$, otherwise it is called *inconsistent*. The aim is to derive weight vector $\underline{w} = (w_1, w_2, \ldots, w_n)^T$ from a PCM that includes the decision maker's subjective judgments.

PCMs can also be used to rank alternatives with respect to a given criterion. Another application of PCMs is to determine the voting power of each decision maker in a group decision problem.

There are several methods for deriving the weight vector [4], we however use the so called eigenvector method proposed by Saaty [14, 15]. For a consistent PCM the following eigenvector equation holds: $A\underline{w} = n\underline{w}$, where $w_i/w_j = a_{ij}, w_i > 0, \sum w_i = 1$. However, PCMs given by real decision makers are rarely consistent. Following the previous equation, Saaty proposed the following method (called the eigenvector method), to gain a weight vector, and also to measure inconsistency. Applying the Perron-Frobenius theorem to a (consistent or inconsistent) PCM, it yields that there is a unique positive Perron eigenvalue λ_{max}, and the corresponding right eigenvector is also positive. It is also known that $\lambda_{max} \geqq n$. Weights can be approximated even in the inconsistent case by the right eigenvector corresponding to λ_{max} (normalizing such that the sum of the weights equal 1), also denoted $\underline{w}$. The eigenvector method provides the weights as the normalized right eigenvector corresponding to the Perron eigenvalue of the PCM.

There are several inconsistency indices in the literature [4], but in the paper we will only discuss one of them: Saaty defined [14] inconsistency index $CR = \frac{CI}{ACI}$, where $CI = \frac{\lambda_{max}-n}{n-1}$, and ACI denotes the mean value of CI calculated from randomly generated PC matrices of size $n \times n$. Saaty also proposed that PCMs below the threshold $CR = 0.1$ are to be considered acceptably inconsistent. CR is a positive linear transformation of λ_{max}: the higher λ_{max} is, the more inconsistent the given PCM is.

In some cases, not all elements of a PCM can be or are desired to be filled in. It can take a lot of effort to obtain all $\frac{n(n-1)}{2}$ pairwise comparisons, especially for large PCMs. In this case, missing elements are allowed in the matrix. Such a matrix is called an incomplete pairwise comparison matrix [7, 9], and has the following general form:

$$A = \begin{pmatrix} 1 & a_{12} & * & \dots & a_{1n} \\ 1/a_{12} & 1 & a_{23} & \dots & * \\ * & 1/a_{23} & 1 & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/a_{1n} & * & 1/a_{3n} & \dots & 1 \end{pmatrix},$$

where $*$ stands for missing elements. They can be in any position except the diagonal, and are symmetric in the sense that if a_{ij} is missing, then a_{ji} is missing, too.

Substitute a variable for each missing element while keeping the reciprocal symmetry rule in mind, and let M denote the number of missing elements above the main diagonal:

$$A(\underline{x}) = A(x_1, x_2, \dots, x_m, \dots, x_M) = \begin{pmatrix} 1 & a_{12} & x_1 & \dots & a_{1n} \\ 1/a_{12} & 1 & a_{23} & \dots & x_M \\ 1/x_1 & 1/a_{23} & 1 & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/a_{1n} & 1/x_M & 1/a_{3n} & \dots & 1 \end{pmatrix}.$$

The aim is still to obtain a weight vector from the matrix. To facilitate this, the eigenvector method can be generalized to the incomplete case, as proposed by Shiraishi, Obata and Daigo [16, 17]. The solution to the eigenvector method shall be the Perron eigenvector corresponding to the least inconsistent completion of the incomplete PCM. Let inconsistency index CR be applied, therefore, the aim is to minimize CR, or equivalently, the Perron eigenvalue λ_{max}:

$$\min_{\underline{x} \in \mathbb{R}_+^M} \lambda_{max}(A(\underline{x})), \tag{1}$$

where $\mathbb{R}_+^M$ denotes the positive orthant of the M-dimensional Eucledian space. This will be our basic problem from now on.

Key to the existence of the minimum of λ_{max} is that problem (1) can be transformed into a convex optimization problem [2], using the following method: Parametrize incomplete PCM $A(\underline{x}) = A(x_1, x_2, \dots, x_m, \dots, x_M)$ such that $x_m = e^{t_m}, (m = 1, 2, \dots, M)$. This way we gain matrix B:

$$A(\underline{x}) = B(\underline{t}) = B(t_1, t_2, \dots, t_m, \dots, t_M) = A(e^{t_1}, e^{t_2}, \dots, e^{t_m}, \dots, e^{t_M}).$$

$\lambda_{max}(B(\underline{t}))$ is now a convex function of $\underline{t}$ [2].

Bozóki et al. [2] characterized when a unique solution exists to problem (1). The graph corresponding to an incomplete pairwise comparison matrix is defined as follows: $G = (V, E)$, $V = 1, 2, \ldots, n$ (vertices correspond to criteria), $E = \{e(i,j) \mid a_{ij}$ is given in the matrix$, i < j\}$ (edges correspond to pairwise comparisons). In other words, two vertices are connected by an edge if the element corresponding to their pairwise comparison is not missing.

Theorem 1 *[2] There exists a unique solution to* $\min_{\underline{x} \in \mathbb{R}^M_+} \lambda_{max}(A(\underline{x}))$ *if and only if the graph corresponding to matrix* A *is connected.*

We will also need the partial derivatives of λ_{max} with respect to the elements of the PCM. According to Harker's formulas [8] both the first and second derivatives can be calculated. These formulas can be found in the Appendix.

In the next section three methods are presented to solve problem (1). First one is the method used by Bozóki et al. [2], which is based on the method of cyclic coordinates, and uses Matlab's *fminbnd* function to solve univariate problems. Second and third ones are the main contributions of the paper. Both methods apply Newton iteration, univariate (Section 2.1.2) and multivariate (Section 2.2). The univariate method is similar to the method using *fminbnd*. It also uses cyclic coordinates, but the inner univariate problem is solved by Newton iteration. The multivariate method is based on the multivariate Newton iteration. A numerical example is presented in section 3. Some of the issues presented in the paper have already been considered, in Hungarian, in [1].

2 Algorithms for optimal completion

2.1 Cyclic coordinates

Let us consider an incomplete pairwise comparison matrix A with a connected graph. Let d denote the number of missing elements from the upper triangle of A, so $A = A(x_1, \ldots, x_M)$. Bozóki et al. [2] proposed a completion method based on cyclic coordinates, as follows. Every variable is given a starting value of $x_m^{(0)}, m = 1, 2, \ldots, M$. Every iteration is composed of M steps. In the first step of the first iteration, let x_1 be the only free variable, while the others are fixed at their starting values $x_m^{(0)}, m = 2, 3, \ldots, M$. Let the single optimum of this single variable optimization problem ($\min_{x_1} \lambda_{max}$) be $x_1^{(1)}$. In the second step of the first iteration, let x_1 be fixed at the value of $x_1^{(1)}$, and let x_2 be the free variable, while all other variables are fixed at their value of $x_m^{(0)}, m = 3, 4, \ldots, M$. Again, from optimizing λ_{max} in x_2, we obtain the optimum $x_2^{(1)}$.

Continue these steps, until we obtain $x_M^{(1)}$. In the second iteration the starting values are $x_m^{(1)}, m = 1, 2, \ldots, M$. So the mth step of the kth iteration is as follows:

$$x_m^{(k)} = \arg\min_{x_m} \lambda_{max}\left(A(x_1^{(k)}, \ldots, x_{m-1}^{(k)}, x_m, x_{m+1}^{(k-1)}, \ldots, x_M^{(k-1)})\right), m = 1, 2, \ldots, M.$$

For the stopping criteria they propose the following: the algorithm stops at the end of the kth iteration if k is the smallest integer for which

$$\max_{m=1,2,\ldots,M} |x_m^{(k)} - x_m^{(k-1)}| < T, \tag{2}$$

where T is the tolerance threshold ($T = 10^{-4}$ is chosen for their and our tests as well).

Another important question is the choice of the starting values. Bozóki et al. [2] in their numerical example used $x_m^{(0)} = 1, m = 1, 2, \ldots, M$. In the paper we will use values based on the solution of the incomplete logarithmic least squares method (ILLSM) [11]. This method determines weight vector $\underline{w}$ by minimizing

$$\min \sum_{\substack{i,j=1 \\ a_{ij} \text{ is given}}}^{n} \left[\log a_{ij} - \log\left(\frac{w_i}{w_j}\right)\right]^2, \tag{3}$$

where $\sum_{i=1}^{n} w_i = 1$ and $w_i > 0 \quad i = 1, \ldots, n$. Solving this problem is based on solving a system of linear equations [2]. Although the ILLSM method can generate ordering different from that of the eigenvector method, it provides reasonable starting values for our iteration [10]. Therefore, the solution of the ILLSM problem will be used for the starting values for $\underline{x}$. Let $w_i^L, i = 1, \ldots, n$ denote the ith component of the weight vector derived from solving the ILLSM problem, and let x_m be in position (i, j). The starting values will be $x_m^{(0)} = w_i^L / w_j^L, m = 1, 2, \ldots, M$.

Again, in order to transform problem (1) to a convex optimization problem, rescaling $x_m = e^{t_m}, m = 1, \ldots, M$ is done [2]. Let $L(t_m) = \lambda_{max}(e^{t_m})$.

The global convergence of cyclic coordinates is stated and proved in [12, pages 253–254].

The cyclic coordinates method presented above will be the framework for the single variable method presented here as well, with the fundamental difference being in how we obtain the optimum.

2.1.1 Cyclic coordinates with Matlab's *fminbnd*

Bozóki et al. [2] used a general optimization function in Matlab (*fminbnd*) for obtaining $\min_{t_m} \lambda_{max}$. Function *fminbnd* uses an algorithm which combines golden section search and parabolic interpolation [3, 5, 13]. A method tailored for this problem and based on the Newton iteration is presented next.

2.1.2 Cyclic coordinates with univariate Newton iteration

Using the method of cyclic coordinates, we are optimizing in only one variable at a time. Let us denote this variable by x, while the other variables are fixed while the minimization occurs. Our goal is to write the Newton iteration of this optimization. Let $x = e^t$ (similarly $x^{(r)} = e^{t^{(r)}}$) and $L(t) = \lambda_{max}(e^t)$. With these notions, we are searching for t where $L'(t) = 0$. Because of this, the rth iteration of Newton's method can be written as

$$t^{(r+1)} = t^{(r)} - \frac{L'(t^{(r)})}{L''(t^{(r)})}.$$

According to Harker [8] the derivatives $\frac{\partial \lambda_{max}(x)}{\partial x}$ and $\frac{\partial^2 \lambda_{max}(x)}{(\partial x)^2}$ are known, and depend on the position (i, j) of x in of the matrix.

To write the Newton iteration we need $\frac{\partial L(t)}{\partial t}$ and $\frac{\partial^2 L(t)}{(\partial t)^2}$.

$$\frac{\partial L(t)}{\partial t} = \frac{\partial \lambda_{max}(e^t)}{\partial t} = \frac{\partial \lambda_{max}(x)}{\partial x} \cdot \frac{\partial e^t}{\partial t} = \frac{\partial \lambda_{max}(x)}{\partial x} \cdot e^t. \tag{4}$$

Similarly

$$\begin{aligned} \frac{\partial^2 L(t)}{(\partial t)^2} = \frac{\partial^2 \lambda_{max}(e^t)}{(\partial t)^2} = \frac{\frac{\partial \lambda_{max}(x)}{\partial x} \cdot e^t}{\partial t} = \\ = \frac{\frac{\partial \lambda_{max}(x)}{\partial x}}{\partial t} \cdot e^t + \frac{\partial \lambda_{max}(x)}{\partial x} \cdot \frac{\partial e^t}{\partial t} = \frac{\partial^2 \lambda_{max}(x)}{(\partial x)^2} \cdot e^{2t} + \frac{\partial \lambda_{max}(x)}{\partial x} \cdot e^t. \end{aligned} \tag{5}$$

Newton iteration can now be written as

$$\begin{aligned} t^{(r+1)} = t^{(r)} - \frac{L'(t^{(r)})}{L''(t^{(r)})} = t^{(r)} - \frac{\frac{\partial \lambda_{max}(x)}{\partial x}(x^{(r)}) \cdot e^{t^{(r)}}}{\frac{\partial^2 \lambda_{max}(x)}{(\partial x)^2}(x^{(r)}) \cdot e^{2t^{(r)}} + \frac{\partial \lambda_{max}(x)}{\partial x}(x^{(r)}) \cdot e^{t^{(r)}}} = \\ = t^{(r)} - \frac{\frac{\partial \lambda_{max}(x)}{\partial x}(x^{(r)})}{\frac{\partial^2 \lambda_{max}(x)}{(\partial x)^2}(x^{(r)}) \cdot e^{t^{(r)}} + \frac{\partial \lambda_{max}(x)}{\partial x}(x^{(r)})}. \end{aligned} \tag{6}$$

As mentioned, $\dfrac{\partial\lambda_{max}(x)}{\partial x}$ and $\dfrac{\partial^2\lambda_{max}(x)}{(\partial x)^2}$ depend on the position of the element x in the matrix (i,j). In a particular step of an iteration all the other variables are temporarily fixed, as described earlier.

A full step of the Newton iteration (which is only a subroutine of a step of the cyclic coordinate iteration) consists of the following steps, where x is in position (i,j):

1. $t^{(r)} = \ln x^{(r)}$,
2. Apply (6),
3. $x^{(r+1)} = e^{t^{(r+1)}}, 1/x^{(r+1)} = e^{-t^{(r+1)}}$.

With this algorithm we managed to apply the Newton iteration specifically for the problem of minimizing the Perron eigenvalue of incomplete PCMs.

2.2 Multivariate Newton iteration

Instead of using the method of cyclic coordinates and optimizing in one variable at a time, one can optimize in all of the variables at the same time, using the multivariate Newton iteration. Let $L(\underline{t}) = \lambda_{max}(e^{t_1}, \ldots, e^{t_M})$. We want to minimize L, so we need:

$$\underline{t}^{(r+1)} = \underline{t}^{(r)} - \gamma[HL(\underline{t}^{(r)})]^{-1}\nabla L(\underline{t}^{(r)}), \tag{7}$$

where $HL(\underline{t}^{(r)})$ is the Hessian matrix of $L(\underline{t})$, and $\nabla L(\underline{t}^{(r)})$ is the gradient vector of $L(\underline{x})$ (both in the rth Newton iteration), and γ is the step size as usual in multivariate Newton iteration. Again, because of the parametrization $\underline{x} = e^{\underline{t}}$, we have to adapt this formula for our case. All of the elements of the gradient vector $\nabla L(\underline{t}) = \left(\dfrac{\partial L(\underline{t})}{\partial t_1}, \ldots, \dfrac{\partial L(\underline{t})}{\partial t_M}\right)$ can be calculated with the method described earlier, namely (4).

Now let us write the Hessian matrix:

$$HL(\underline{t}) = \begin{pmatrix} \dfrac{\partial^2 L(\underline{t})}{\partial t_1^2} & \dfrac{\partial^2 L(\underline{t})}{\partial t_1 \partial t_2} & \cdots & \dfrac{\partial^2 L(\underline{t})}{\partial t_1 \partial t_M} \\ \dfrac{\partial^2 L(\underline{t})}{\partial t_2 \partial t_1} & \dfrac{\partial^2 L(\underline{t})}{\partial t_2^2} & \cdots & \dfrac{\partial^2 L(\underline{t})}{\partial t_2 \partial t_M} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 L(\underline{t})}{\partial t_M \partial t_1} & \dfrac{\partial^2 L(\underline{t})}{\partial t_M \partial t_2} & \cdots & \dfrac{\partial^2 L(\underline{t})}{\partial t_M^2} \end{pmatrix}.$$

Diagonal elements are calculated by (5). We still need to reformulate the off-diagonal elements of the Hessian matrix, where we differentiate with respect to different variables. From now on, $x_p = e^{t_p}$ is in position (i, j) and $x_q = e^{t_q}$ is in position (u, v).

$$\frac{\partial^2 L(\underline{t})}{\partial t_p \partial t_q} = \frac{\partial^2 \lambda_{max}(e^{t_1}, \ldots, e^{t_M})}{\partial t_p \partial t_q} = \frac{\partial \left(\frac{\partial \lambda_{max}(e^{t_1}, \ldots, e^{t_M})}{\partial t_q} \right)}{\partial t_p}.$$

We can apply (4), and get $\frac{\partial \lambda_{max}(e^{t_1}, \ldots, e^{t_M})}{\partial t_q} = \frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \cdot e^{t_q}$. Including the case $p = q$ as well,

$$\frac{\partial^2 L(\underline{t})}{\partial t_p \partial t_q} = \frac{\partial \left(\frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \cdot e^{t_q} \right)}{\partial t_p} = \frac{\partial \left(\frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \right)}{\partial t_p} \cdot e^{t_q} + \frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \cdot \frac{\partial e^{t_q}}{\partial t_p}. \tag{8}$$

Here

$$\frac{\partial e^{t_q}}{\partial t_p} = e^{t_q} \cdot \chi_{\{p=q\}}, \tag{9}$$

where

$$\chi_{\{p=q\}} = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}.$$

On the other hand,

$$\frac{\partial \left(\frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \right)}{\partial t_p} = \frac{\partial \left(\frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \right)}{\partial x_p} \cdot \frac{\partial x_p}{\partial t_p} = \frac{\partial \left(\frac{\partial \lambda_{max}(\underline{x})}{\partial x_q} \right)}{\partial t_p} \cdot \frac{\partial e^{t_p}}{\partial t_p} = \frac{\partial^2 \lambda_{max}(\underline{x})}{\partial x_p \partial x_q} \cdot e^{t_p}, \tag{10}$$

which also includes the case $p = q$. Writing (9) and (10) back into (8) we get the final form of

$$\frac{\partial^2 L(\underline{t})}{\partial t_p \partial t_q} = \frac{\partial^2 \lambda_{max}(\underline{x})}{\partial x_p \partial x_q} \cdot e^{t_p + t_q} + \frac{\partial \lambda_{max}(\underline{x})}{\partial x_p} \cdot e^{t_p} \cdot \chi_{\{p=q\}}. \tag{11}$$

Note that (5) is a special case of (11) with $p = q$. In (11) we can calculate $\frac{\partial^2 \lambda_{max}(\underline{x})}{\partial x_p \partial x_q}$ and $\frac{\partial \lambda_{max}(\underline{x})}{\partial x_p}$ according to Harker's formulas [8], and they depend on the positions of the variables in the matrix.

Using these formulas, and given a starting value of $\underline{t}_0$ for $\underline{t}$, we can calculate the gradient vector and Hessian matrix for every iteration of the multivariate Newton's method (7). Again, the iteration continues while (2) is satisfied. The stopping criteria is determined for $\underline{x}$ (not $\underline{t}$), because a small difference in $\underline{t}$ can lead to large differences in $\underline{x}$.

3 Numerical example

Let us consider the following incomplete pairwise comparison matrix:

$$\begin{pmatrix} 1 & 5 & 3 & 7 & 6 & 6 & 1/3 & 1/4 \\ 1/5 & 1 & x_1 & 5 & x_2 & 3 & x_3 & 1/7 \\ 1/3 & 1/x_1 & 1 & x_4 & 3 & x_5 & 6 & x_6 \\ 1/7 & 1/5 & 1/x_4 & 1 & x_7 & 1/4 & x_8 & 1/8 \\ 1/6 & 1/x_2 & 1/3 & 1/x_7 & 1 & x_9 & 1/5 & x_{10} \\ 1/6 & 1/3 & 1/x_5 & 4 & 1/x_9 & 1 & x_{11} & 1/6 \\ 3 & 1/x_3 & 1/6 & 1/x_8 & 5 & 1/x_{11} & 1 & x_{12} \\ 4 & 7 & 1/x_6 & 8 & 1/x_{10} & 6 & 1/x_{12} & 1 \end{pmatrix}.$$

This is an incomplete version of Saaty's "buying a house" example matrix [15]. Further, it is the same incomplete PCM which was used as an example by Bozóki et al. [2], where it is shown that the graph corresponding to this matrix is connected.

In Tables 1 and 2 the values of each variable in each iteration are shown using the univariate Newton iteration and the multivariate Newton iteration, respectively. However, iteration does not mean the same thing in these cases. For the univariate Newton method, an iteration k is the outer iteration which contains $m = 12$ complete univariate Newton iterations for each k. For the multivariate Newton iteration, an iteration r is one iteration of the multivariate Newton method itself.

As mentioned earlier, $T = 10^{-4}$ in (2), and the starting values $x_m^{(0)}$ are chosen to be equal to the optimal solution of the ILLSM problem (3). For the multivariate case, several γ values have been experimented with. $\gamma = 0.45$ yielded the lowest number of iterations, therefore this value was used for the results in Table 2. The number of iterations required was $k = 14$ in the univariate case, and $r = 14$ in the multivariate case.

Tests were also done for starting values $x_m^{(0)} = 1, m = 1, \ldots, M$ (again with $\gamma = 0.45$ in the multivariate case). The univariate method required $k = 15$ iterations, while the multivariate method required $r = 26$ iterations.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$	$x_5^{(k)}$	$x_6^{(k)}$	$x_7^{(k)}$	$x_8^{(k)}$	$x_9^{(k)}$	$x_{10}^{(k)}$	$x_{11}^{(k)}$	$x_{12}^{(k)}$
0	0.3823	1.8430	0.4758	8.9920	4.2690	0.5228	0.5361	0.1384	0.8855	0.1085	0.2916	0.4200
1	0.3460	1.7620	0.4699	9.6590	4.7370	0.5667	0.5320	0.1434	0.9206	0.1091	0.2932	0.4008
2	0.3338	1.7320	0.4690	9.8410	4.8230	0.5671	0.5283	0.1431	0.9270	0.1088	0.2924	0.4016
3	0.3315	1.7270	0.4678	9.8840	4.8370	0.5681	0.5271	0.1428	0.9283	0.1090	0.2919	0.4023
4	0.3308	1.7240	0.4671	9.9000	4.8440	0.5687	0.5264	0.1427	0.9295	0.1091	0.2916	0.4026
5	0.3305	1.7220	0.4668	9.9090	4.8470	0.5691	0.5259	0.1426	0.9302	0.1092	0.2914	0.4028
6	0.3303	1.7210	0.4666	9.9140	4.8500	0.5693	0.5256	0.1425	0.9306	0.1093	0.2913	0.4029
7	0.3302	1.7210	0.4665	9.9170	4.8510	0.5694	0.5255	0.1425	0.9309	0.1093	0.2913	0.4030
8	0.3301	1.7200	0.4664	9.9180	4.8520	0.5695	0.5254	0.1425	0.9310	0.1093	0.2913	0.4030
9	0.3301	1.7200	0.4664	9.9190	4.8520	0.5696	0.5253	0.1425	0.9311	0.1093	0.2912	0.4031
10	0.3300	1.7200	0.4664	9.9200	4.8520	0.5696	0.5253	0.1424	0.9311	0.1093	0.2912	0.4031
11	0.3300	1.7200	0.4664	9.9200	4.8520	0.5696	0.5253	0.1424	0.9312	0.1093	0.2912	0.4031
12	0.3300	1.7200	0.4664	9.9200	4.8520	0.5696	0.5253	0.1424	0.9312	0.1093	0.2912	0.4031
13	0.3300	1.7200	0.4664	9.9200	4.8520	0.5696	0.5253	0.1424	0.9312	0.1093	0.2912	0.4031
14	0.3300	1.7200	0.4664	9.9210	4.8520	0.5696	0.5253	0.1424	0.9312	0.1093	0.2912	0.4031

Table 1: Univariate Newton

r	$x_1^{(r)}$	$x_2^{(r)}$	$x_3^{(r)}$	$x_4^{(r)}$	$x_5^{(r)}$	$x_6^{(r)}$	$x_7^{(r)}$	$x_8^{(r)}$	$x_9^{(r)}$	$x_{10}^{(r)}$	$x_{11}^{(r)}$	$x_{12}^{(r)}$
0	0.3823	1.8430	0.4758	8.9920	4.2690	0.5228	0.5361	0.1384	0.8855	0.1085	0.2916	0.4200
1	0.3385	1.7160	0.4484	9.7830	4.7170	0.5786	0.5006	0.1276	0.9592	0.1221	0.2739	0.4427
2	0.3430	1.7540	0.4707	9.5490	4.6180	0.5500	0.5368	0.1446	0.9082	0.1087	0.2946	0.4040
3	0.3285	1.7070	0.4581	9.9020	4.8210	0.5755	0.5169	0.1376	0.9436	0.1149	0.2848	0.4172
4	0.3323	1.7260	0.4672	9.8070	4.7800	0.5640	0.5297	0.1435	0.9243	0.1096	0.2924	0.4035
5	0.3287	1.7140	0.4637	9.9140	4.8430	0.5715	0.5234	0.1412	0.9349	0.1112	0.2894	0.4074
6	0.3302	1.7200	0.4664	9.8880	4.8320	0.5682	0.5268	0.1428	0.9294	0.1096	0.2916	0.4034
7	0.3295	1.7180	0.4656	9.9170	4.8490	0.5702	0.5251	0.1422	0.9322	0.1099	0.2908	0.4043
8	0.3299	1.7200	0.4664	9.9110	4.8470	0.5693	0.5258	0.1426	0.9308	0.1094	0.2913	0.4033
9	0.3298	1.7190	0.4662	9.9190	4.8520	0.5698	0.5253	0.1424	0.9315	0.1095	0.2911	0.4035
10	0.3300	1.7200	0.4664	9.9180	4.8510	0.5696	0.5255	0.1425	0.9311	0.1094	0.2913	0.4032
11	0.3299	1.7200	0.4663	9.9200	4.8520	0.5697	0.5253	0.1424	0.9313	0.1094	0.2912	0.4032
12	0.3300	1.7200	0.4664	9.9200	4.8520	0.5696	0.5253	0.1425	0.9312	0.1093	0.2912	0.4031
13	0.3300	1.7200	0.4663	9.9200	4.8520	0.5696	0.5253	0.1424	0.9312	0.1093	0.2912	0.4031
14	0.3300	1.7200	0.4664	9.9200	4.8520	0.5696	0.5253	0.1424	0.9312	0.1093	0.2912	0.4031

Table 2: Multivariate Newton

4 Conclusions

When using the generalized eigenvector method for incomplete pairwise comparison matrices, the matrix is to be completed optimally with regard to its inconsistency, or equivalently minimizing its Perron eigenvalue λ_{max}. Although eigenvalue minimization problems are generally difficult due to nonconvexity, the special case of incomplete PCMs proves to be convex. In order to solve the problem, a method based on the Newton iteration (both univariate and multivariate) is presented in the paper.

Future research can be focused on the choice of γ in case of the multivariate method. Fülöp [6] has recently proposed an alternative method for minimizing λ_{max}, further research includes a comparative analysis of the algorithms.

Acknowledgements

The author thanks János Fülöp (Institute for Computer Science and Control, Hungarian Academy of Sciences – MTA SZTAKI; Óbuda University) for his remarks. The author thanks his PhD supervisor Sándor Bozóki (Institute for Computer Science and Control, Hungarian Academy of Sciences – MTA SZTAKI; Corvinus University of Budapest) for valuable discussions. The support of OTKA grant K 111797 is gratefully acknowledged.

References

[1] K. Ábele-Nagy, *Incomplete pairwise comparison matrices in multi-attribute decision making (In Hungarian, Nem teljesen kitöltött páros összehasonlítás mátrixok a többszempontú döntésekben)*, Master's Thesis, Eötvös Loránd University, Budapest, 2010. ⇒61

[2] S. Bozóki, J. Fülöp, L. Rónyai, On optimal completion of incomplete pairwise comparison matrices, *Math. Comput. Modelling* **52,** 1–2, (2010) 318–333. ⇒60, 61, 62, 63, 66

[3] R. P. Brent, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973. ⇒63

[4] M. Brunelli, *Introduction to the Analytic Hierarchy Process*, SpringerBriefs in Operations Research, Springer, New York, 2015. ⇒59

[5] G. E. Forsythe, M. A. Malcolm, C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1976. ⇒63

[6] J. Fülöp, An optimization approach for the eigenvalue method, *IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI 2013)*, Timişoara, Romania, 23–25 May 2013. ⇒69

[7] P. T. Harker, Alternative modes of questioning in the Analytic Hierarchy Process. *Math. Model.* **9,** 3 (1987) 353–360. ⇒59
[8] P. T. Harker, Derivatives of the Perron root of a positive reciprocal matrix: with application to the Analytic Hierarchy Process. *Appl. Math. Comput.* **22,** 2–3 (1987) 217–232. ⇒61, 63, 65, 70
[9] P. T. Harker, Incomplete pairwise comparisons in the Analytic Hierarchy Process. *Math. Model.* **9,** 11 (1987) 837–848. ⇒59
[10] A. Ishizaka, M. Lusti, How to derive priorities in AHP: a comparative study. *Cent. Eur. J. Oper. Res.*, **14,** 4 (2006) 387–400. ⇒62
[11] M. Kwiesielewicz, The logarithmic least squares and the generalised pseudoinverse in estimating ratios. *European J. Oper. Res.* **93,** 3 (1996) 611–619. ⇒62
[12] D. G. Luenberger, Y. Ye, *Linear and Nonlinear Programming* (3rd Edition), Series: International Series in Operations Research & Management Science, **116**, Springer, New York, 2008. ⇒62
[13] Matlab Documentation: fminbnd `http://www.mathworks.com/help/matlab/ref/fminbnd.html`. ⇒63
[14] T. L. Saaty, A scaling method for priorities in hierarchical structures. *J. Math. Psych.*, **15,** 3 (1977) 234–281. ⇒59
[15] T. L. Saaty, *The Analytic Hierarchy Process*, McGraw-Hill, New York, 1980. ⇒59, 66
[16] S. Shiraishi, T. Obata, M. Daigo, Properties of a positive reciprocal matrix and their application to AHP. *J. Oper. Res. Soc. Jpn.* **41,** 3 (1998) 404–414. ⇒60
[17] S. Shiraishi, T. Obata, On a maximization problem arising from a positive reciprocal matrix in AHP. *Bull. Inform. Cybernet.* **34,** 2 (2002) 91–96. ⇒60

Received: January 24, 2015 • Revised: April 29, 2015

Appendix

Harker's [8] formulas for the derivatives of the Perron eigenvalue are presented in the appendix.

Let A denote a PCM, and let $\underline{x} = \underline{x}(A)$ and $\underline{y} = \underline{y}(A)$ denote its right and left Perron eigenvectors, and $\lambda_{max} = \lambda_{max}(A)$ its Perron eigenvalue, so $A\underline{x} = \lambda_{max}\underline{x}$ and $\underline{y}^T A = \lambda_{max}\underline{y}^T$. The normalization for the eigenvectors in this case is $\underline{y}^T\underline{x} = 1$. Let $Q = \lambda_{max} I - A$. Also let Q^+ denote the pseudoinverse of Q, which satisfies the following properties: $QQ^+Q = Q$, $Q^+QQ^+ = Q^+$, $Q^+Q = QQ^+$. Finally, ∂a_{ij} denotes differentiation with respect to the element in position (i, j) in A, and similarly ∂a_{kl} denotes differentiation with respect to the element in position (k, l). Using these notations, the formulas are as follows:

The first derivatives:

$$\frac{\partial \lambda_{max}}{\partial a_{ij}} = y_i x_j - \frac{y_j x_i}{a_{ij}^2}, \text{ if } i > j.$$

The second derivatives:

$$\frac{\partial^2 \lambda_{max}}{\partial a_{ij} \partial a_{kl}} = \begin{cases} (\underline{x}\underline{y}^T)_{li} Q^+_{jk} + (\underline{x}\underline{y}^T)_{jk} Q^+_{li} - \frac{(\underline{x}\underline{y}^T)_{ki} Q^+_{jl} + (\underline{x}\underline{y}^T)_{jl} Q^+_{ki}}{a_{kl}^2} - \\ - \frac{(\underline{x}\underline{y}^T)_{lj} Q^+_{ik} + (\underline{x}\underline{y}^T)_{ik} Q^+_{lj}}{a_{ij}^2} + \frac{(\underline{x}\underline{y}^T)_{kl} Q^+_{il} + (\underline{x}\underline{y}^T)_{il} Q^+_{kj}}{a_{ij}^2 a_{kl}^2} \\ \qquad\qquad \text{if } i > j,\ k > l,\ (i,j) \neq (k,l) \\ 2\frac{(\underline{x}\underline{y}^T)_{ij}}{a_{ij}^3} + 2(\underline{x}\underline{y}^T)_{ji} Q^+_{ji} - \\ - 2\frac{(\underline{x}\underline{y}^T)_{ii} Q^+_{jj} + (\underline{x}\underline{y}^T)_{jj} Q^+_{ii}}{a_{ij}^2} + 2\frac{(\underline{x}\underline{y}^T)_{ij} Q^+_{ij}}{a_{ij}^4} \\ \qquad\qquad \text{if } i > j,\ (i,j) = (k,l). \end{cases}$$

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 72–106

DOI: 10.1515/ausi-2015-0013

Tripartite graphs with given degree set

Antal IVÁNYI
Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary
email: tony@inf.elte.hu

Shariefuddin PIRZADA
University of Kashmir, Srinagar, India
email:
pirzadasd@kashmiruniversity.ac.in

Farooq A. DAR
University of Kashmir
Srinagar, India
email: sfarooqdar@yahoo.co.in

Abstract. If $k \geq 1$, then the *global degree set* of a k-partite graph $G = (V_1, V_2, \ldots, V_k, E)$ is the set of the distinct degrees of the vertices of G, while if $k \geq 2$, then the *distributed degree set* of G is the family of the k degree sets of the vertices of the parts of G. We propose algorithms to construct bipartite and tripartite graphs with prescribed global and distributed degree sets consisting from *arbitrary* nonnegative integers. We also present a review of the similar known results on digraphs.

1 Introduction

In this paper we follow the terminology used in the monography of Chartrand, Lesniak and Zhang [5] and the handbook of Gross, Yellen and Zhang [11].

Let $m \geq 0$ and $n \geq 1$ be integers, $G = (V, E)$ be a finite, simple graph with vertex set $V(G) = \{v_1, \ldots, v_n\}$ and edge set $E(G) = \{e_1, \ldots, e_m\}$. The

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C07
Key words and phrases: global degree set, distributed degree set, bipartite graph, tripartite graph

degree d_i of a vertex v_i is the number of edges of G which are incident on v_i. The *degree sequence* $d = [d_1, d_2, \ldots, d_n]$ of G is the sequence of degrees of G, usually put in nonincreasing or nondecreasing order. The number of vertices is called the *order* of G, while the number of edges is called the *size* of G. A degree sequence d is said to be *graphic* if it is the degree sequence of some finite graph. The set of the disjoint degrees $\gamma = \{g_1, g_2, \ldots, g_p\}$ of the vertices of G is called its *degree set*. If $k \geq 2$ is an integer and $G = (V_1, V_2, \ldots, V_k, E)$ is a k-partite graph, then the degree set of the vertex set V_i $(1 \leq i \leq k)$ is called the *global degree set of the* i*-th part* of G and is denoted by $\gamma_i(V_i)$; the union $\prod_{i=1}^{k} \gamma_i$ is called the *global degree set of* G and is denoted by $\gamma(G)$. In the literature instead of the global degree set usually the shorter *degree set* expression is used, but we wish to underline the difference between the simple and multipartite graphs. The family of the degree sets γ_i is called the *distributed degree set of* G and is denoted by $\delta(G)$.

The papers of Tyshkevich and Chernyak [55, 56, 57, 58] contain a review on the different generalizations of score sequences.

Some early results on degree sets of simple graphs and trees (acyclic connected graphs) were published in 1977 by Kapoor, Polimeni and Wall. They introduced the concept of degree set and proved the following theorem

Theorem 1 (Kapoor, Polimeni, Wall [25]) *If* p *is a positive integer, then any set* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *of positive integers with* $g_1 < g_2 < \cdots < g_p$ *is the degree set of a connected graph* G *and the minimum order of such a graph is* $g_p + 1$.

Proof. See [25, 37, 54]. □

In 1979 Koukichi and Katsuhiro [28] reproved Theorem 1. They defined (n, k)-sets as sets of integers $\{h_1, \ldots, h_k\}$ with $n-1 \geq h_1 > h_2 > \cdots > h_k \geq 0$. Further they defined $DG_n(k)$ for any positive n and k with $1 \leq k \leq n-1$ as the set of all degree sets D of graphs G of order n with $|D| = k$, and $F_n(k)$ as the set of all (n, k)-sets $D = \{d_1, \ldots, d_k\}$ satisfying: (i) if $d_1 = n-1$, then $d_k > 0$ and (ii) if $n = 1 \pmod k$ then D contains at least one even number.. Among others they expressed $DG_n(2)$ in terms of $F_n(2)$, and proved $DG_n(3) = F_n(3)$ and $DG_n(n-2) = F_n(n-2)$ for $n > 2$.

A short proof of this result is due to Tripathi and Vijay [54].

A simple consequence of Theorem 1 is the following assertion allowing $0 \in \gamma$ and not containing the condition of sorted γ.

Corollary 2 *If* $p \geq 1$ *is an integer, then any set* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *of nonnegative integers is the degree set of a graph* G *and if* $0 \notin \gamma$, *then the minimum*

order of such graphs is $\max(\gamma)+1$*, otherwise* $\max(\gamma)+2$*.*

Proof. If $0 \notin \gamma$, then we can use the proof of Theorem 1, otherwise we can add an isolated vertex to the minimal size graph corresponding to $\gamma \setminus 0$. □

In 2006 Ahuja and Tripathi investigated the possible sizes of graphs having prescribed degree set and extended Theorem 1 giving *all* possible size of graphs having a prescribed degree set. We say, that a graph G is a (p,γ)-graph, if its size is p and its degree set is γ.

Theorem 3 (Ahuja, Tripathi [1]) *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be any finite, nonempty set of positive integers and let* $m = \max(\gamma)$*. If all members of* γ *are odd, then there exists a* (p,γ)*-graph if and only if* $p > m$ *and* p *is even; otherwise there exists a* (p,γ)*-graph if and only if* $p > m$*, provided also that* $p \neq m+2$ *in the special case, where* $\gamma = \{1, m\}$ *for any even integer* $m \geq 4$*.*

Proof. See [1]. □

One can ask, what is the answer, if we allow $0 \in \gamma$? Using Theorem 3 it is easy to show, that in this case all sizes greater than p are possible.

In 1980 Sipka investigated the problem or the possible orders of graphs having a prescribed global degree set $\gamma = \{g_1, g_2, \ldots, g_n\}$ of integers with $1 \leq g_1 < g_2 < \cdots < g_n$. His results are summarized in the following theorem.

Theorem 4 (Sipka [52]) *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a nonempty set of positive integers with* $g_1 < g_2 < \cdots < g_p$*.*

(i) If p *is even,* g_i *is odd for all* $1 \leq i \leq p$*,* $p > g_p$*, then there exists a graph* G *of order* p *with* $\gamma(G) = \gamma$*.*

(ii) If g_i *is even for some* $1 \leq i \leq p$*,* $t \geq 2$ *and* $\gamma \neq \{1, 2, \ldots, 2t\}$*, then there exists a graph* G *of order* p *such that* $\gamma(G) = \gamma$ *for all positive* $p > g_n$*, then there exists a graph* G *of order* p *such that* $\gamma(G) = \gamma$*.*

(iii) If $t \geq 2$ *is an integer,* $\gamma = \{1, 2, \ldots, 2t\}$*, then there exists a graph* G *of order* p *for all positive* p *exceeding* g_p*, with the exception of* $g_n + 2$*.*

Proof. See [52]. □

Let $p \geq 1$ be an integer, further let $\gamma = \{g_1, g_2, \ldots, g_p\}$ be a set of integers with $0 \leq g_1 < g_2 < \cdots < g_p$. Then $\mu_{dc}(\gamma) = \mu_{dc}\{g_1, g_2, \ldots, g_p\}$ denotes the minimum order of disconnected graphs G for which $\gamma(G) = \gamma$.

In 2004 Manoussakis et al. investigated disconnected graphs. Let $\gamma = \{g_1, g_2, \ldots, g_p\}$ be a nonempty set of nonnegative integers with $0 \leq g_1 < g_2 < \cdots < g_p\}$ and let $\mu_{dc}(\gamma) = \mu_{dc}\{g_1, g_2, \ldots, g_p\}$ denote the minimum order of a disconnected graph G for which $\gamma(G) = \gamma$.

Manoussakis, Patil and Sankar assert [32], that if g is a nonnegative integer, then $\mu(g) = 2(g+1)$. This assertion is not correct. We give the correct formula.

Theorem 5 *If g is a nonnegative integer, then $\mu(g) = g + 2$.*

Proof. If G is disconnected and has a vertex with degree g, then it has at least $(g+1)+1 = g+2$ vertices. But a star with $g+1$ vertices plus an isolated vertex form a corresponding graph. □

For the case $p \geq 2$ Manoussakis et al. proved the following assertion.

Theorem 6 (Manoussakis, Patil, Sankar [30, 31, 32]). *Let $p \geq 2$ be an integer and $\gamma = \{g_1, g_2, \ldots, g_p\}$ be a set of nonnegative integers with $g_1 < g_2 < \cdots < g_p$. Then there exists a disconnected graph G such that $\gamma(G) = \gamma$. Further, $\mu_{dc} = g_1 + g_p + 2$.*

Proof. See [31, 32]. □

Manoussakis, Patil and Sankar [30, 32] also investigated degree sets for k-connected graphs, k-edge connected graphs, unicyclic graphs and maximal k-degenerate graphs.

Let m be a positive integer. An (m, γ)-*graph* has m edges and degree set γ. We denote by $e(\gamma)$ the least m, for which there exists an (m, γ)-graph, and by $l_e(\gamma)$ this least $e(\gamma)$.

In 2006 Tripathi and Vijay determined $l_e(\gamma)$ in the following cases:

a) $|\gamma| \leq 3$;

b) $t \geq 1$ is an integer and $\gamma = \{1, 2, \ldots, t\}$;

c) $\min(\gamma) \geq |\gamma|$.

Further, they gave lower and upper bounds for $l_e(\gamma)$ in all cases and exhibited the cases, when the bounds are tight.

In their paper Tripathi and Vijay use the following notations. If $s = [d_1, d_2, \ldots, d_n]$ is an increasing degree sequence, then its short form is $s = [d_1^{m_1}, d_2^{m_2}, \ldots, d_p^{m_p}]$, where $d_i^{m_i}$ denotes m_i copies of d_i.

Theorem 7 (Tripathi, Vijay [53]). *Let g be a nonnegative integer. If $\gamma = \{g\}$, then there exists a (q, γ)-graph if and only if*

$$e \in \begin{cases} \left\{ mg : m \geq \dfrac{g+1}{2} \right\} & \text{if } g \text{ is odd,} \\ \left\{ m\dfrac{g}{2} : m \geq g+1 \right\} & \text{if } g \text{ is even.} \end{cases}$$

In particular, $l_e(\{g\}) = \frac{1}{2}g(g+1)$.

Theorem 8 (Tripathi, Vijay [53]). *Let* a *and* b *be positive integers with* $a > b$ *and let* $\gamma = \{a, b\}$. *Then there exists a* (q, γ)*-graph if and only if* q *has the form* $\frac{1}{2}(ma + mb)$, *where* m *and* n *are positive integers,* $m + n \geq a + b$, *and*

$$1 \leq m \leq b \text{ or } m \geq a + 1 \text{ or } m(a + 1 - m) \leq mb \text{ with } b + 1 \leq m \leq a.$$

In particular,

$$l_q(\{a, b\}) = \begin{cases} \dfrac{a(b+1)}{2} & \text{if } a \text{ is even or } b \text{ is odd,} \\ \dfrac{a(b+1)+(a-b)}{2} & \text{if } a \text{ is odd or } b \text{ is even.} \end{cases}$$

Theorem 9 (Tripathi, Vijay [53]). *Let* t *be a positive integer and let* $\gamma = \{1, 2, \ldots, t\}$. *Then there exists an* (e, α)*-graph if and only if*

$$e \geq \left\lceil \frac{t}{2} \right\rceil + 1).$$

In particular, $l_e(\gamma = \lceil \frac{t}{2} \rceil (\lfloor \frac{t}{2} \rfloor + 1)$.

In their paper Tripathi and Vijay constructed a special $\gamma = \{g_1, g_2, \ldots, g_t\}$ and determined its $l(\gamma)$ as follows:

$$l(\gamma) = \begin{cases} \frac{1}{2}p_0(\gamma) & \text{if } p_0(\gamma) \text{ is even} \\ (\frac{1}{2}p_0(\gamma) + g_r - g_p) & \text{if } p_0(\gamma) \text{ is odd and } g_p \text{ is even} \\ \min(\frac{1}{2}p_0(\alpha) + a_r - a_t, \frac{1}{2}(p_0(\alpha) + a_p)) & \text{if both } p_0(\gamma) \text{ and } g_p \text{ are odd.} \end{cases}$$

Their following result shows that the above bounds are achieved for infinite number of sets.

Theorem 10 (Tripathi, Vijay [53]). *Let* γ *be a finite set of positive integers such that* $\min(\gamma) \geq |\gamma|$. *Then* $l_e(\gamma) = l(\gamma)$.

The proofs of these theorems due to Tripathi and Vijay can be found in [53].

In 2011 Volkmann extended Theorem 1 to multigraphs, proving the following assertion.

Theorem 11 (Volkmann [59]) *Let* $p \geq 1$ *and integer and and* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a set of integers such that*

$$g_1 > g_2 > \cdots > g_p \geq 1.$$

(i) If $k = 1$*, then* γ *is the degree set of a multigraph of order two.*

Assume now that $k \geq 2$ *and* $g_1 \leq \sum_{i=2}^{p} g_i$.

(ii) If $\sum_{i=2}^{p} g_i$ *is even, then* γ *is the degree set of a multigraph of order* k.

(iii) If If $\sum_{i=2}^{p} g_i$ *is odd, then* γ *is the degree set of a multigraph of order* $k+1$.

Next assume that $k \geq 2$ *and* $g_1 > \sum_{i=2}^{p} g_i$.

(iv) If $g_1 + \sum_{i=1}^{k} g_i$ *even, then* γ *is the degree set of a multigraph of order* $k+1$.

In addition, assume in the following that $g_1 + \sum_{i=1}^{p} g_i$ *is odd.*

(v) Let $\sum_{i=1}^{k} g_i$ *be even. If there exists an index* $2 \leq k$ *such that* g_j *is even and* $g_1 \leq g_j + \sum_{i=2}^{k} g_i$*, then* δ *is the degree set of a multigraph of order* $k+1$*. If there is no such index, then* γ *is the degree set of a multigraph of order* $k+2$.

(vi) Let $\sum_{i=1}^{k} g_i$ *be odd. If there exists an index* $2 \leq j \leq k$ *such that* g_j *is odd and* $g_1 + \sum_{i=2}^{k} g_i$*, then* γ *is the degree set of a multigraph of order* $k+2$.

In all cases of the multigraph is the least possible one.

Proof. See [59]. □

The *girth* of a graph is defined as the length of a shortest cycle in the graph. For integers $r \geq 2$ and $g \geq 3$ $f(r, g)$ is defined as the smallest order of an r-regular graph, having girth g. Such graphs are called *cages* [5, 37]. In 1963 Erdős and Sachs [5, 10] not only proved the existence of all cages but gave an upper bound for their order.

Theorem 12 (Erdős, Sachs [10])**.** *If* $r \geq 2$ *and* $g \geq 3$*, then*

$$1 + r \sum_{t=0}^{\lceil (g-3)/2 \rceil} (r-1)^t \leq f(r, g) \leq 4 \sum_{t=1}^{g} (r-1)^t.$$

Proof. See [10]. □

Erdős and Sachs remarked that Theorem 12 can be improved using the method proposed by Ferenc Kárteszi [26].

For $k \geq 1$, $n \geq 3$ and a set of integers $\gamma = \{g_1, g_2, \ldots, g_k\}$ with $2 \leq g_1 < \cdots < g_k$ we define

$$f(\gamma, g) = f(g_1, g_2, \ldots, g_k; g)$$

to be the smallest order of graph having girth g and degree set D.

In 1981 Chartrand, Gould and Kapoor proved the following four theorems on the values of $f(D, g)$.

Theorem 13 (Chartrand, Gould, Kapoor [3]) *If* $n \geq 1$ *and* $\gamma = \{g_1, g_2, \ldots, g_n\}$ *is a set of integers with* $2 \leq g_1 < g_2 < \cdots < g_n$, *then* $f(\gamma; 3) = 1 + a_n$.

Theorem 14 (Chartrand, Gould, Kapoor [3]) *For* $m \geq 3$ *and* $n \geq 3$

$$f(2, m; n) = \begin{cases} \dfrac{m(n-2)+4}{2} & \text{if } n \text{ is even,} \\ \dfrac{m(n-1)+2}{2} & \text{if } n \text{ is odd.} \end{cases}$$

Theorem 15 (Chartrand, Gould, Kapoor [3]) *If* $2 \leq s$, *then*

$$f(r, s; 4) = s.$$

Theorem 16 (Chartrand, Gould, Kapoor [3]) $f(3, 4; 5) = 13$, $f(3, 4; 6) = 18$.

In 1982 Wang published a survey [61] on the results connected with cages.

In 1988 Chernyak [6] continued the investigation of $f(r, g)$.

A graph having the minimum number of vertices in the class of graphs with degree set $\gamma = \{g_1, g_2, \ldots, g_p\}$ and girth m is called a $(\gamma; p)$-cage; the order of a $(\gamma; p)$-cage is denoted by $f(\gamma; p)$. In this paper some new values of the function f are determined constructively: $f(3, 4, r; 5) = 3k + 1$ for $k = 5$ and $r = 4$, as well as for $k \geq 6$ and $4 \leq r \leq 3 + 2\lceil (k-5/3)/2 \rceil$; $f(3, 4, k; 6) = 4k + 1$ for $k \geq 5$; $f(3, k; 6) = 4k + 2$ for $k \geq 4$; $f(3, 4, k; 7) = 7k + 1$ for $k \geq 4$, and $f(3, 4; 8) = 39$.

In 1985 Mynhardt [34] determined the condition of the existence of a degree uniform graph having prescribed global degree set.

A *signed graph* G is a graph in which to each edge is assigned a positive or negative sign. The set of distinct signed degrees D of a signed graph G is called its *global signed degree set.*

The concept of signed graphs was introduced and firstly characterized by Harary in 1953 [15]. In the first paper he proved the following assertions. According to his paper a signed graph, $G = (V, L^+, L^-)$ consists of a vertex set $V = \{V_1, V_2, \ldots, V_n\}$, and two disjoint sets of edges L^+ and L^-.

Theorem 17 (Harary [15]) *A complete signed graph is balanced if and only if its vertex set* V *can be partitioned into two disjoint subsets* V_1 *and* V_2, *one of which may be empty, such that all edges between the vertices of the same subset are positive, and all edges between vertices of the two different subsets are negative.*

Proof. See [15]. □

Theorem 18 (Harary [15]) *A signed graph is balanced if and only if for each pair of distinct vertices* A *and* B *all paths joining* A *and* B *are positive.*

Proof. See [15]. □

Theorem 19 (Harary [15]) *A signed graph is balanced if and only if its vertex set can be partitioned into two disjoint subsets* V_1 *and* V_2 *in such a way that eachl positive edge of* G *joins two vertices of different subsets.*

Proof. See [15]. □

In 1955 Harary [16] presented enumeration results on the different types of graphs including also signed graphs.

A sequence $\sigma = (d_1, d_2, \ldots, d_n)$ of integers is *standard*, if it is nonincreasing, the sum of its element is even, $d_1 > 0$, each $|d_i| < n$, and $|d_1| \geq |d_n|$.

In 1968 Chartrand et al. published the following assertion, similar to the well-known theorem of Hakimi [13] for the degree sets of graphs.

Theorem 20 (Chartrand, Gavlas, Harary, Schulz [2]) *If* $\gamma = (g_1, g_2, \ldots, g_p)$ *is a standard sequence, then there exists a signed graph* G *with global signed degree set* γ *if and only if there exists a signed graph* G' *with signed global degree set*

$$\sigma' = (g_2-1, g_3-1, \ldots, g_{g_1+s+1}-1, g_{g_1+s+2}, \ldots, g_{p-s}, \ldots, g_{p-s+1}+1, \ldots, g_{p+1})$$

for some s, $0 \leq s \leq \frac{p-1-g_1}{2}$.

Theorem 21 (Yan, Lih, Kuo, Chang [62]) *let* γ *be a standard sequence. There exists a signed graph with global signed degree sequence* γ *if and only if there exist integers* r *and* s *with* $g_1 = r - s$, $0 \leq s \leq \frac{p-1-g_1}{2}$ *such that there exist a a isigned graph* G' *with global signed score set*

$$\gamma' = \{g_2 - 1, g_3 - 1, \ldots, g_{g_1+m+1} - 1, g_{g_1+m+2}, \ldots, g_{p-m}, g_{p-m+1} + 1, g_p + 1\}.$$

In 2007 Pirzada et al. improved Theorem 1 proved by Kapoor, Polimeni and Wall in 1977.

Theorem 22 (Pirzada, Naikoo, Dar [47]) *Let* p *be a positive integer and* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a set of integers with* $g_1 < g_2 < \cdots < g_p$. *Then* γ *is the signed global score set of some connected signed graph* G *and the minimal order of such signed graphs is* $g_p + 1$.

Proof. See [47] □

In 2013 Kumar, Sarma, and Sawlami [29] studied the number of vertices and multiplicity of degrees as parameters of directed and undirected tree realizations of prescribed degree sets.

2 Bipartite graphs with prescribed global degree sets

A graph $B(V, E)$ is said to be bipartite (or bigraph or 2-partite graph) if its vertex set V can be partitioned into two disjoint sets V_1 and V_2 with $V = V_1 \cup V_2$ so that if $uv \in E$, then u and v belong to different vertex sets. We will use the notation $B(V_1, V_2, E)$. A bipartite graph is *complete* if $uv \in E$ for every $u \in V_1$ and every $v \in V_2$. If $|V_1| = n_1$ and $|V_2| = n_2$, then the complete bipartite graph is denoted by K_{n_1,n_2}. Examples of bipartite graphs are trees, cycle graphs with even number of vertices, planar graphs whose faces all have even length (special cases of this are grid graphs and square graphs), hypercube graphs, partial cubes and median graphs. Bipartite graphs can be characterized in several different ways such as (i) A graph is bipartite if and only if it does not contain an odd cycle, (ii) A graph is bipartite if and only if it is 2-colorable.

The set of distinct degrees $\{g_1, g_2, \ldots, g_p\}$ of $B = (V_1, V_2, E)$ is called the *global degree set* of B and is denoted by $\gamma(B)$ (or simply by γ). For any non-empty subset U of $V_1 \cup V_2$ $\gamma(U)$ denotes the set of degrees of vertices in U. Then, the *global degree set* of a bipartite graph B with a bipartition (V_1, V_2) is the set $\gamma(B)$ which is the union of the sets of degrees in V_1 and in V_2, i.e. $\gamma(B) = \gamma(V_1) \cup \gamma(V_2)$.

In 1977 Kapoor et al. proved the following assertion on the existence of trees (ie., connected, bipartite acyclic graphs) having prescribed global degree set.

Theorem 23 (Kapoor, Polimeni, Wall [25]) *Let $\gamma = \{g_1, g_2, \ldots, g_p\}$ be a non-empty set of positive integers. Then there exists a nontrivial tree T (i.e. a connected, acyclic bipartite graph) with global degree set $\gamma(T) = \gamma$ if and only if $1 \in \gamma$. Further, if $1 \in \gamma$, then the minimum order of a nontrivial tree T with $\gamma(T) = \gamma$ is $\sum_{i=1}^{n}(g_i - 1) + 2$.*

Proof. See [25]. □

If $q \geq 2$, then every even cycle C_{2q} is bipartite with $\gamma(C_{2q}) = \{2\}$ and moreover, $\mu(C_{2q}) = 4$.

In 1979 Kapoor and Lesniak [24] studied the minimal order of bipartite graphs, having a prescribed global degree set. They received partial results: in

some special cases determined the minimal order of triangle-free graphs having prescribed degree set.

In 1994 Ellis [9] published a paper on layered graphs called by him $(k, 2)$-partite graphs in which he proposed effective sequential and parallel algorithms to decide whether a given graph is $(k, 2)$-layered, and also for the effective solution of several connected problems.

In 2007 Pirzada, Naikoo and Dar proved the following assertion.

Theorem 24 (Pirzada, Naikoo, Dar [48]) *Every set of positive integers is the global degree set of some connected bipartite graph.*

Proof. See [48]. □

Recently Manoussakis and Patil determined the families of connected unicyclic bipartite graphs having prescribed global degree set.

Theorem 25 (Manoussakis, Patil [33]). *Let* $p \geq 2$ *be an integer and* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a set of positive integers with* $g_1 < g_2 < \cdots < g_p$. *Then there exists a connected unicyclic bipartite graph* B *with* $\gamma(B) = \gamma$ *if and only if either (a) or (b) below holds:*

a) $p = 2$, $g_1 = 1$ *and* $g_2 \geq 3$. *In this case* $\mu(\gamma) = 4(g_2 - 1)$.

b) $p \geq 3$ *and* $g_1 = 1$. *In this case*

$$\mu(\gamma) = \begin{cases} 3g_2 + g_3 - 4, & \text{if } p = 3, \\ 2g_2 + g_3 + g_4 - 4, & \text{if } p = 4, \\ \sum_{i=2}^{n}(g_i - 1), & \text{if } p \geq 5. \end{cases}$$

Proof. See [33]. □

The paper of Manoussakis and Patil contains the following lemma too.

Lemma 26 (Manoussakis, Patil [33]). *For any given positive integer* n, *there exists a complete bipartite graph* B *with bipartition* (X, Y) *such that* $\gamma(B) = \{n\}$ *if and only if* $n = |X| = |Y|$.

Proof. See [33]. □

Here we prove that every finite set of positive integers is the global degree set of some connected bipartite graph. Our approach is constructive and is different from that used in [33, 48, 39].

At first we prove a useful lemma.

Lemma 27 *If* $g_1, g_2, \ldots, g_p$ *is a nonempty set of nonnegative integers with* $0 \leq g_1 < g_2 < \cdots < g_p\}$, *then there exists a bipartite graph* $B = (V_1, V_2, E)$ *with global degree set* $\gamma(B) = \left\{g_1, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i\right\}$.

Proof. We consider two cases, (a) when $g_1 = 0$ and (b) $g_1 > 0$.

Case (a). Let $g_1 = 0$. We have three subcases to consider.

(i) Suppose $p = 1$. We choose the null bipartite graph $G(V_1, V_2, E)$ with $|V_1| = |V_2| = 1$ and $E = \emptyset$. In this case the degrees of the vertices $v_1 \in V_1$ and $v_2 \in V_2$ are $g_{v_1} = g_{v_2} = 0 = g_1$. So degree set of $G(V_1, V_2, E)$ is $\gamma = g_1$.

(ii) Now let $n = 2$. We construct a bipartite graph $G(V_1, V_2, E)$ as follows.

Let $V_1 = X_1 \cup X_2, V_2 = Y_1$ with $X_1 \cap X_2 = \phi$, $|X_1| = 1, |X_2| = |Y_1| = g_2$. Take an edge from each vertex of X_2 to every vertex of Y_1. The degrees of the vertices of $G(V_1, V_2, E)$ are as follows.

For $x_1 \in X_1$, $g_{x_1} = 0 = g_1$ and for all $x_2 \in X_2$, $g_{x_2} = |Y_1| = g_2 = g_1 + g_2$; and for all $y_1 \in Y_1$, $g_{y_1} = |X_2| = g_2 = g_1 + g_2$.

Thus the degree set of $G(V_1, V_2, E)$ is $\gamma = \{g_1, g_1 + g_2\}$.

(iii) For $n \geq 3$, we construct a bipartite graph $G(V_1, V_2)$ whose

$$V_1 = (\bigcup_{i=1}^{p} X_i) \bigcup (\bigcup_{i=3}^{n} X'_i) \text{ and } V_2 = (\bigcup_{i=1}^{n-1} Y_i) \bigcup (\bigcup_{i=2}^{n-1} Y'_i),$$

where for all $i \neq j$, $X_i \cap X_j = \phi, X_i \cap X'_j = \phi, Y_i \cap Y'_j = \phi$, $Y'_i \cap X'_j = \phi$;

for all i, $2 \leq i \leq p$, $|X_1| = 1, |X_i| = |Y_{i-1}| = d_i$;

for all i, $3 \leq i \leq p$, $|X'_1| = |Y'_{i-1}| = \sum_{r=2}^{i-1} g_r$.

We choose an edge from each vertex of X_i to every vertex of Y_j whenever $i \geq j$; an edge from each vertex of X'_i to every vertex of Y_{i-1} for all i, $3 \leq i \leq p$; and an edge from each vertex of X'_i to every vertex of Y'_{i-1} for all i, $3 \leq i \leq p$.

The degrees of the vertices of the bipartite graph $G(V_1, V_2.E)$ constructed above are as follows.

For $x_1 \in X_1$, $g_{x_1} = 0 = g_1$ and for $2 \leq i \leq p$, for all $x_i \in X_i$,

$$\begin{aligned} g_{x_i} &= \sum_{j=1}^{i-1} |Y_j| = \sum_{j=1}^{i-1} |g_{j+1}| \\ &= g_2 + g_3 + \cdots + g_i = g_1 + g_2 + \cdots + g_i; \end{aligned}$$

for $3 \leq i \leq n$, for all $x'_i \in X'_i$,

$$\begin{aligned} d(x'_i) &= |Y_{i-1}| + |Y'_{i-1}| \\ &= g_i + g_2 + g_3 + \cdots + g_{i-1} \\ &= g_1 + g_2 + g_3 + \cdots + g_i; \end{aligned}$$

for all $y_1 \in Y_1$,

$$\begin{aligned} d(y_1) &= \sum_{i=2}^{n} |X_i| = \sum_{i=2}^{p} g_i \\ &= g_2 + g_3 + \cdots + g_n \\ &= g_1 + g_2 + g_3 + \cdots + g_n; \end{aligned}$$

for $2 \leq i \leq n-1$, for all $y_i \in Y_i$

$$\begin{aligned} d(y_i) &= \sum_{j=i+1}^{p} |X_j| + |X'_{i+1}| = \sum_{j=i+1}^{p} g_j + (g_2 + \cdots + g_i) \\ &= g_{i+1} + g_{i+1} + \cdots + g_p + g_2 + g_3 + \cdots + g_i \\ &= g_1 + g_2 + \cdots + g_p; \end{aligned}$$

for $2 \leq i \leq p-1$, for all $y'_i \in Y'_i$

$$\begin{aligned} d(y'_i) &= |X'_{i+1}| = g_2 + \cdots + g_i \\ &= g_1 + g_2 + g_3 + \cdots + g_i. \end{aligned}$$

Therefore we see that the degree set of $G(V_1, V_2)$ is

$$\gamma = \{g_1, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i\}.$$

Case (b) Now let $g_1 > 0$. We have two subcases.
(i) Let $p = 1$. We consider the bipartite graph $G(V_1, V_2)$ with $V_1| = |V_2| = g_1$ in which there is an edge from each vertex of V_1 to every vertex of V_2. Here the degrees of the vertices of $G(V_1, V_2)$ are given as $d(v_1) = |V_2| = g_1$ and $d(v_2) = |V_1| = g_1$, for all $v_1 \in V_1, v_2 \in V_2$. Thus the degree set of $G(V_1, V_2)$ is $D = g_1$.
(ii) Let $p \geq 2$. Consider the bipartite graph $G(V_1, V_2)$ whose

$$V_1 = (\bigcup_{i=1}^{n} X_i) \bigcup (\bigcup_{i=2}^{n} X'_i),$$

$$V_2 = (\bigcup_{i=1}^{p} Y_i) \bigcup (\bigcup_{i=2}^{p} Y'_i),$$

where (for $i \neq j$) each $X_i \cap X_j$, $X_i \cap X'_j$, $X'_i \cap X'_j$, $Y_i \cap Y_j$, $Y_i \cap Y'_j$ and $Y'_i \cap Y'_j$ are empty, for all i, $1 \leq i \leq p$, $|X_i| = |Y_i| = g_i$. Also for all i, $2 \leq i \leq p$, we have $|X'_i| = |Y'_i| = g_1 + g_2 + \cdots + g_{i-1}$.

Take an edge from each vertex of X_i to every vertex of Y_j whenever $i \geq j$; an edge from each vertex of X'_i to every vertex of Y_i for all i, $2 \leq i \leq n$ and an edge from each vertex of X'_i to every vertex of Y'_i for all i, $2 \leq i \leq p$. The following are the degrees of the vertices of the bipartite graph $G(U, V)$ constructed above.

For $1 \leq i \leq p$, for all $x_i \in X_i$,

$$d(x_i) = \sum_{j=1}^{i-1} |Y_j| = \sum_{j=1}^{i} g_j = g_1 + g_2 + \cdots + g_i.$$

For $2 \leq i \leq p$, for all $x'_i \in X'_i$,

$$d(x'_i) = |Y_i| + |Y'_i| = g_i + (g_1 + g_2 + \cdots + g_{i-1}) = g_1 + g_2 + \cdots + g_i.$$

For $1 \leq i \leq p$, for all $y_i \in Y_i$,

$$\begin{aligned} d(y_i) &= \sum_{j=1}^{n} |X_j| + |X'_i| \\ &= \Big(\sum_{j=1}^{n} g_j\Big) + (g_1 + g_2 + \cdots + g_{i-1}) \\ &= (g_i + g_{i+1} + \cdots + g_n) + (g_1 + g_2 + \cdots + g_{i-1}) \\ &= g_1 + g_2 + \cdots + g_p. \end{aligned}$$

For $2 \leq i \leq p$, for all $y'_i \in Y'_i$,

$$d(y'_i) = |X'_i| = g_1 + g_2 + \cdots + g_{i-1}.$$

Therefore the degree set of the bipartite graph $G(U, V)$ constructed above is $D = \{g_1, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i\}$. □

Using Lemma 27, we can prove the following assertion.

Theorem 28 *Every set of nonnegative integers is the global degree set of some bipartite graph.*

Proof. Let $\gamma = \{a_1, a_2, \ldots, a_n\}$ be any set of distinct nonnegative integers. We choose

$$b_1 = a_2 - a_1, b_2 = a_3 - a_2, \cdots, b_{p-1} = a_p - a_{p-1}.$$

Now γ can be rewritten as

$$\begin{aligned}\gamma &= \{a_1, a_2 - b_1 + b_1, a_3 - b_2 + b_2, \cdots, a_p - b_{p-1} + b_{p-1}\} \\ &= \{a_1, a_1 + b_1, a_2 + b_2, \ldots, a_{p-1} + b_{p-1}\} \\ &= \{a_1, (a_1 + b_1), (a_1 + b_1 + b_2), \ldots, (a_1 + b_1 + b_2 + \cdots + b_{p-1})\}.\end{aligned}$$

As seen in Theorem 1, the set $\gamma = \{a_1, (a_1 + b_1), (a_1 + b_1 + b_2), \ldots, (a_1 + b_1 + b_2 + \cdots + b_{p-1})\}$ is the global degree set of some bipartite graph which is equivalent to say that the set $\gamma = \{a_1, a_2, \ldots, a_p\}$ is the global degree set of some bipartite graph. □

Example 29 *Consider* $\gamma = \{0, 5, 7\}$. *We can rewrite* γ *as* $\gamma = \{0, 5{+}0, 5{+}0{+}2\}$ *and apply Corollary 2, then we get the bipartite graph with degree set* γ. *In case* $\gamma = \{3, 5, 10, 12\}$, *we write* γ *as* $\gamma = \{3, 3 + 2, 3 + 2 + 5, 3 + 2 + 5 + 2\}$.

From case (b) of Lemma 27, we have the following assertion.

Theorem 30 *Every set of positive integers is the global degree set of some connected bipartite graph.*

The following algorithm Global-Bipartite is based on Theorem 41. Therefore the algorithm is a sligtly modified version of Distributed-Bipartite. Global-bipartite constructs a bipartite graph having prescribed global degree set.

Input. p: the number of elements in the prescribed degree set γ;
$\gamma = \{g_1, g_2, \ldots, g_p\}$: the prescribed degree set for $B(V_1, V_2, E)$.

Output. $M(B)$: the incidence matrix of the constructed bipartite graph (V_1, V_2, B).
n_1: the number of lines of M, that is the size of the vertex set V_1;
n_2: the number of columns of M, that is the size of the vertex set V_2.

Work variables. i, j: cycle variables.

The pseudocodes of this paper are written using the conventions described in the textbook written by Cormen, Leiserson, Rivest, and Stein [7].

```
GLOBAL-BIPARTITE(p, γ)
01 α = 0                                   // lines 01–03: computation of α
02 for i = 1 to p
03     α = α + g_i
04 ν = α^2
08 n_1 = ν/p
09 n_2 = ν/p
10 for i = 1 to ν                          // lines 10–12: initialization of M
11     for j = 1 to ν
12         M_ij = 0
13 i = 1
14 j = 1
15 x = n_1
16 y = n_2
17 while j < ν
18         while x > 0 and y > 0
19                 M_ij = 1
20                 x = x − 1
21                 y = y − 1
22                 j = j + 1
23                 if x == 0
24                    x = n_1
25                    i = i + 1
26                 if y == 0
27                    y = n_2
28 return μ, M
```

Theorem 31 *Let* $\sum_{i=1}^{p} g_i = s$. *Then the running time of* GLOBAL-BIPARTITE *is* $\Theta(s^2/p^2)$ *in all cases.*

Proof. See the proof of Theorem 43. □

Let m and n be positive integers. A *signed bipartite graph* $B = (U, V, E)$ with $U = \{u_1, u_2, \ldots, u_{n_1}\}$ and $V = \{v_1, v_2, \ldots, v_{n_2}\}$ is a bipartite graph in which to each edge is assigned a positive or negative sign. The *signed degree* of a u_i is defined as $g_{u_i} = g_i = g_i^+ - g_i^-$, where $1 \leq i \leq n_1$ and g_i^+ (g_i^-) is the number of positive (negative) edges incident to u_i, and the *signed degree* of a v_j is defined as $g_{v_j} = g_j = g_j^+ - g_j^-$, where $1 \leq j \leq n_2$ and $g_j^+ g_j^-$) is the number of positive (negative) edges incident to v_j. The *global degree set* γ of a signed bipartite graph is the set of its distinct signed degrees.

In 2006 Pirzada et al. proved the following properties of signed bipartite graphs.

Theorem 32 (Pirzada, Naikoo, Dar [46, 49]) *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a nonempty set set of positive (or negative) integers.Then* γ *is the signed global degree set of some signed bipartite graph, and the minimal order of such graphs is* $1 + \max_{1 \le i \le p} |g_i|$.

Proof. See [49]. □

Theorem 33 (Pirzada, Naikoo, Dar [46, 49]) *Let* p *be a positive integer and* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a set of negative integers. Then* γ *is the signed global degree set of some signed bipartite graph, and the minimal order of such graphs is* $|\min(\gamma)|$.

Proof. See [46, 49]. □

As the following assertion shows, the requirement of the identical sign of the elements of the score set can be removed.

Theorem 34 (Pirzada, Naikoo, Dar [46, 49]) *Let* γ *be a set of integers. Then* γ *is the signed global degree set of some signed bipartite graph.*

Proof. See [46, 49]. □

In 2008 Pirzada et al. published the followig result.

Theorem 35 (Pirzada, Naikoo, Dar [46, 49]) *Let* p *be a positive integer and* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a set of positive integers. Then* $\sum_{i=1}^{1} g_i, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i$ *is the signed global degree set of some signed bipartite graph, and the minimal order of such graphs is* $|\min(\gamma)|$.

Proof. See [49]. □

3 Bipartite graphs with prescribed distributed degree set

Let p be a positive integer and $B = (V_1, V_2, E)$ a bipartite graph, where $\delta(V_1) = \{a_1, a_2, \ldots, a_p\}$, and $\delta(V_2) = \{b_1, b_2, \ldots, b_p\}$. Then the pair (δ_1, δ_2) is called the *distributed degree set* of B.

Given a pair (δ_1, δ_2) of finite, nonempty sets of positive integers, let $\mu(\delta_1 \cup \delta_2) = \min\{|B| : B \in \mathcal{B}$, where $\mathcal{B}$ is the family of all bipartite graphs B with

$\delta(B) = (\delta_1 \cup \delta_2)$. Manoussakis and Patil [33] have shown for a given pair (δ_1, δ_2) of finite, nonempty sets of positive integers of same cardinality the existence of a bipartite graph $B(V_1, V_2)$ such that $\delta(V_1) = \delta_1$ and $\delta(V_2) = \delta_2$ and obtained the minimum orders of different types of such graphs.

Theorem 36 (Manoussakis, Patil [33]) *Let* $\delta_1 = \{a_1, a_2, \ldots, a_p\}$ *and* $\delta_2 = \{b_1, b_2, \ldots, b_p\}$ *be nonempty sets of positive integers with* $a_1 < a_2 < \cdots < a_p$, *and* $b_1 < b_2 < \cdots < b_p$. *Then there exists a bipartite graph* $B = (V_1, V_2, E)$ *with distributed score set* (δ_1, δ_2). *Furthermore,* B *is connected if and only if the minimum order* $\mu(\delta_1 \cup \delta_2) = a_p + b_p$, *where* $|V_1| = a_p$, *and* $|V_2| = b_p$.

Proof. See [33]. The proof of 36 begins with the interesting remark that $B = \bigcup(\overline{K}_{a_i} + \overline{K}_{b_i})$ satisfies the required property. Then comes an inductive proof which is not correct. For example on page 387 in 6th and 5th lines from below if $n = 3$, then $3 \leq m < n$ is meaningless. On the next page in the third line $a_1 u$ is also an error. □

Manoussakis and Patil published also the following corollaries of Theorem 36 (the proofs can be seen in the same paper [33]).

Corollary 37 *Let* $\delta_1 = \{a_1, a_2, \ldots, a_p\}$, *and* $\delta_2 = \{b_1, b_2, \ldots, b_p\}$ *be nonempty sets of different positive integers such that* $a_1 < a_2 < \cdots < a_p$, *and* $b_1 < b_2 < \cdots < b_p\}$. *Then there exists a connected, bipartite graph* $B(V_1, V_2, E)$ *of order* $a_p + b_p$ *such that* $\delta(V_1) = \delta_1$ *and* $\delta(V_2) = \delta_2$, *where* $|V_1| = a_p$ *and* $|V_2| = b_p$.

Corollary 38 *Let* $\delta = \{a_1, a_2, \ldots, a_p\}$ *be nonempty set of positive integers with* $a_1 < a_2 < \cdots < a_p$. *Then there exists a connected, bipartite graph* B *with bipartition* (V_1, V_2) *such that the global degree sets* $\delta(V_1)$ *and* $\delta(V_2)$ *are different, and the global degree set* $\delta(B)$ *is* δ. *Moreover, the minimum order of* B *with* $\delta(B) = \delta$ *is*

$$\mu(\delta) = \begin{cases} a_{p/2} + a_p & \text{if } p \text{ is even,} \\ a_{\lceil p/2 \rceil} + a_p & \text{if } p \text{ is odd.} \end{cases}$$

Corollary 39 *Let* $\delta = \{a_1, a_2, \ldots, a_p\}$ *be a nonempty set of positive integers with* $1a_1 < a_2 < \cdots < a_p$. *Then there exists a bipartite graph* $B = (V_1, V_2, E)$ *such that* $\delta(B) = \delta$. *Furthermore,* $B(V_1, V_2, E)$ *is a connected, bipartite graph such that* $\delta(V_1) = \delta(V_2)$ *if and only if its minimum order is* $2a_p$ *so that* $|V_1| = |V_2| = a_p$.

Corollary 40 *Let* $\delta_1 = \{a_1, a_2, \ldots, a_p\}$ *be a nonempty set of positive integers with* $a_1 < a_2 < \cdots < a_p$. *Then there exists a connected bipartite graph* $B(V_1, V_2, E)$ *of order* $2a_p$, *such that* $\delta(V_1) = \delta(V_2) = \delta$, *where* $|V_1| = |V_2| = a_p$.

It is worth to mention, that this corollary is not true: e.g. if $\delta_1 = \{-5, -3\}$, then "order $2a_n$" is meaningless, therefore in the lemma we substituted "the set of integers" with "a set of positive integers".

Manoussakis and Patil finished their paper so: "For arbitrary sets δ_1 and δ_2 of positive integers with **different** cardinalities, the problem of determining a bipartite graph that holds the property in Theorem 36 is open."

Our next theorem shows, that the existence of a bipartite graph having prescribed distributed degree set does not require the condition $|\delta_1| = |\delta_2|$.

Theorem 41 *Let* $\delta_1 = \{a_1, a_2, \ldots, a_p\}$ *and* $\delta_2 = \{b_1, b_2, \ldots, b_q\}$ *be nonempty sets of nonnegative integers. Then there exists a bipartite graph* $B = (V_1, V_2, E)$ *such that* $\delta(V_1) = \delta_1$ *and* $\delta(V_2) = \delta_2$.

Proof. If $a_1 = 0$, then we delete a_1 from δ_1 resulting $\delta_1' = \{c_1, c_2, \ldots, c_{p-1}\}$. If $b_1 = 0$, then we delete b_1 from δ_2 resulting $\delta_2' = \{d_1, d_2, \ldots, d_{q-1}\}$.

Let $\alpha = \sum_{i=1}^{p} a_i$, $\beta = \sum_{j=1}^{q} b_j$, $\mu = \alpha\beta$, $V_1 = \{u_1, u_2, \ldots, u_\mu\}$ and $V_2 = \{v_1, v_2, \ldots, v_\mu\}$. Let the multiplication factor of V_1 be $m_1 = \mu/p$, the multiplication factor of V_2 be $m_2 = \mu/q$, the degree sequence of V_1 be $\sigma_1 = [c_1^{n_1}]$

Now let consider the bipartite graph $B(V_1, V_2, E)$, where $|V_1| = p\beta$, and $|V_2| = q\alpha$, the degree sequence of V_1 is $\sigma_1 = (a_1^\beta, a_2^\beta, \ldots, a_p^\beta\} = (e_1, e_2, \ldots, , e_\mu)$, and the degree sequence of V_2 is $\sigma_2 = (b_1^\alpha, b_2^\alpha, \ldots, b_p^\alpha) = (f_1, f_2, \ldots, f_\mu)$. Let $V_1 = \{u_1, u_2, \ldots, u_\mu\}$ and $V_2 = \{v_1, v_2, \ldots, v_\mu\}$.

We construct the edge set of B as follows. We connect in cyclical order u_1 with the next e_1 vertices in V_2 (that is with $v_1, v_2, \ldots, v_{e_1}$), then connect u_2 with the next e_2 vertices in V_2 (that is with $v_{e_1+1}, v_{e_1+2}, \ldots, v_{e_1+e_2}$) and so on. □

It is a simple observation, that if $0 \in \delta_1 \cup \delta_2$, then the graphs with distributed degree set (δ_1, δ_2) are never connected.

The following example shows that the absence of zero in the prescribed distributed degree set is not sufficient to guarantee the existence of a corresponding connected bipartite graph.

Example 42 *Let* $\delta_1 = \{1\}$ *and* $\delta_2 = \{1, 2\}$. *In all construction we have to connect vertices whose degree is 1, and so this pair of vertices will not be connected with the remaining part of the constructed graph.*

The following program constructs a bipartite graph having a prescribed distributed degree set.

Input. p: the number of elements in the prescribed degree set δ_1 of V_1;
q: the number of elements in the prescribed degree set δ_2 of V_2;

$\delta_1 = \{a_1, a_2, \ldots, a_p\}$: prescribed degree set for V_1
$\delta_2 = \{b_1, b_2, \ldots, b_q\}$: prescribed degree set for V_2.

Output. $\mu = \alpha\beta$: the number of rows and columns of M, that is the common length of the degree sequence of V_1 and V_2;
M: the incidence matrix of the constructed bipartite graph $B = (V_1, V_2, E)$.

Work variables. i, j: cycle variables;
α: the sum of the elements of δ_1;
β: the sum of the elements of δ_2;
ν: the common length of the degree sequence of V_1 and V_2, and so the common number of rows and columns in the incidence matrix M;
$n_1 = \nu/p$: the multiplication factor of the degree sequence of V_1;
$n_2 = \nu/q$: the multiplication factor of the degree sequence of V_2.

DISTRIBUTED-BIPARTITE$(p, q, \delta_1, \delta_2)$

```
01 α = 0                                        // lines 01–03: computation of α
02 for i = 1 to p
03     α = α + a_i
04 β = 0                                        // lines 04–06: computation of β
05 for i = 1 to q
06     β = β + b_i
07 ν = αβ                          // lines 07–09: computation of β, n_1 and n_2
08 n_1 = ν/p
09 n_2 = ν/q
10 for i = 1 to ν                               // lines 10–12: initialization of M
11     for j = 1 to ν
12         M_ij = 0
13 i = 1                           // lines 13–16: initialization of i, j, x, and y
14 j = 1
15 x = n_1
16 y = n_2
17 while j ≤ ν                          // lines 17–27: connecting of the vertices
18         while x > 0 and y > 0
19             M_ij = 1
20             x = x − 1
21             y = y − 1
22             j = j + 1
23             if x == 0
24                 x = n_1
25                 i = i + 1
```

```
26                  if y == 0
27                     y = n2
28 return μ, M
```

Theorem 43 *The running time of* DISTRIBUTED-BIPARTITE *is in all cases* $\Theta(\nu)$.

Proof. The deciding parts of the running time are required by lines 10–12 and by lines 17–27 and both parts requires $\Theta(\nu)$ time. □

Comparing with the algorithm proposed by Manoussakis and Patil disadvantage of DISTRIBUTED-BIPARTITE is that it constructs usually larger solution.

4 Tripartite graphs with prescribed global degrees

A graph $T(V, E)$ is said to be tripartite graph (or trigraph or 3-partite graph) if its vertex set V can be partitioned into three disjoint sets V_1, V_2, and V_3 with $V = V_1 \cup V_2 \cup V_3$ such that if $uv \in E$, $u \in V_i$ and $v \in V_j$, then $i \neq j$. A tripartite graph is *complete* if there is edge from each $v \in V_i$ to every $v \in V_j$ with $i \neq, 1 \leq i,\ j \leq 3$. If $|V_1| = n_1$, $|V_2| = n_2$ and $|V_3| = n_3$, then the complete bipartite graph is denoted by K_{n_1,n_2,n_3}.

The set of distinct degrees of T is called its *global degree set* and is denoted by $\gamma(T)$ (or simply γ).

In 2007 Pirzada, Naikoo and Dar proved the following assertions.

Theorem 44 (Pirzada, Naikoo, Dar [48]) *Let* $\gamma\{g_1, g_2, \ldots, g_p\}$ *be a nonempty set of nonnegative integers. Then there exists a tripartite graph* $T = (V_1, V_2, V_3, E)$ *with global degree set* γ.

Proof. See [48] □

Theorem 45 (Pirzada, Naikoo, Dar [48]). *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a nonempty set of nonnegative integers with* $g_2 g_3 \cdots g_p > 0$. *Then there exists a tripartite graph whose global degree set is* $\gamma = \{g_1, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i\}$.

Proof. To prove the existence of such tripartite graphs, we consider two cases, (a) when $g_1 = 0$ and (b) $g_1 > 0$.

Case (a) Let $g_1 = 0$. We consider three subcases.

(i) Suppose $n = 1$. We choose the null tripartite graph $G(V_1, V_2, V_3, E)$ with $|V_1| = |V_2| = |V_3| = 1$. Here the degrees of the vertices $v_1 \in V_1$, $v_2 \in V_2$

and $v_3 \in V_3$ are $d_{v_1} = d_{v_2} = d_{v_3} = 0 = g_1$. So the global degree set of $G(V_1, V_2, V_3, E)$ is $\gamma = \{g_1\}$.

(ii) We now take $n = 2$. Consider a tripartite graph $G(V_1, V_2, V_3, E)$ with $|V_1| = 1, |V_2| = |V_3| = d_2$. Suppose there is an edge from each vertex of V_2 to every vertex of V_3. We observe that the degrees of the vertices of $G(V_1, V_2, V_3, E)$ are as under.

For $v_1 \in V_1$, $d_{v_1} = 0 = g_1$; for all $v_2 \in V_2$, $d_{v_2} = |V_3| = g_2 = g_1 + g_2$; and for all $v_3 \in V_3$, $d_{v_3} = |V_2| = g_2 = g_1 + g_2$.

Thus the degree set of $G(V_1, V_2, V_3, E)$ is $D = \{g_1, g_1 + g_2\}$.

(iii) Now let $p \geq 3$. We construct a tripartite graph $G(V_1, V_2, V_3, E)$ whose

$$V_1 = \bigcup_{i=1}^{n} X_i, V_2 = \bigcup_{i=1}^{n-2} Y_i \text{ and } V_3 = Z_1 \bigcup (\bigcup_{i=2}^{n-1} Z_i) \bigcup (\bigcup_{i=2}^{p-1} Z'_i),$$

where for all $i \neq j$, $X_i \cap X_j = \phi$, $Y_i \cap Y_j = \phi$, $Z_i \cap Z_j = \phi$, $Z_i \cap Z'_j = \phi$, $Z'_i \cap Z'_j = \phi$;

for all $i, 2 \leq i \leq p$, $|X_1| = 1, |X_i| = |Z_{i-1}| = d_i$;

for all $i, 1 \leq i \leq p-2$, $|Y_i| = |Z'_{i+1}| = \sum_{r=2}^{i+1} g_r = g_2 + g_3 + \cdots + g_{i+1}$.

We choose an edge from each vertex of X_i to every vertex of Z_j whenever $i \geq j$; an edge from each vertex of Y_i to every vertex of Z_{i+1} for all i, $1 \leq i \leq p-2$; and an edge from each vertex of Y_i to every vertex of Z'_{i+1} for all i, $1 \leq i \leq p-2$.

The degrees of the vertices of the tripartite graph $G(V_1, V_2, V_3, E)$ constructed above are as follows.

For $x_1 \in X_1$, $d_{x_1} = 0 = g_1$;

and for $2 \leq i \leq n$, for all $x_i \in X_i$,

$$\begin{aligned} d_{x_i} &= \sum_{j=1}^{i-1} |Z_j| = \sum_{j=1}^{i-1} |g_{j+1}| \\ &= g_2 + g_3 + \cdots + g_i = g_1 + g_2 + \cdots + g_i; \end{aligned}$$

for $1 \leq i \leq n-2$, for all $y_i \in Y_i$,

$$\begin{aligned} d(y_i) &= |Z_{i+1}| + |Z'_{i+1}| \\ &= g_{i+2} + g_2 + g_3 + \cdots + g_{i+1} \\ &= g_1 + g_2 + g_3 + \cdots + g_{i+2}; \end{aligned}$$

for all $z_1 \in Z_1$,

$$\begin{aligned} d(z_1) &= \sum_{i=2}^{n} |X_i| = \sum_{i=2}^{n} g_i \\ &= g_2 + g_3 + \cdots + g_n \\ &= g_1 + g_2 + g_3 + \cdots + g_n; \end{aligned}$$

for $2 \leq i \leq n-1$, for all $z_i \in Z_i$

$$\begin{aligned} d(z_i) &= \sum_{j=i+1}^{n} |X_j| + |Y_{i-1}| = \sum_{j=i+1}^{n} g_j + (g_2 + \cdots + g_i \\ &= g_{i+1} + g_{i+2} + \cdots + g_n + g_2 + g_3 + \cdots + g_i \\ &= g_1 + g_2 + \cdots + g_n; \end{aligned}$$

for $2 \leq i \leq n-1$, for all $z_i' \in Z_i'$

$$\begin{aligned} d(z_i') &= |Y_{i-1}'| = g_2 + g_3 + \cdots + g_i \\ &= g_1 + g_2 + g_3 + \cdots + g_i. \end{aligned}$$

Thus we see that the degree set of $G(V_1, V_2, V_3)$ is

$$D = \{g_1, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i\}.$$

Case (b) Assume $g_1 > 0$. We have two subcases.

(i) Let $p = 1$. We consider the tripartite graph $G(V_1, V_2, V_3, E)$ with $V_1 = X_1$, $V_2 = Y_1$, $V_3 = Z_1 \bigcup Z_2$ and $Z_1 \bigcap Z_2 = \phi$, $|X_1| = |X_2| = |X_3| = 1$. In this graph, let there be an edge from each vertex of X_1 to every vertex of Z_1, and from each each vertex of Y_1 to every vertex of Z_2. Then the degrees of the vertices of $G(V_1, V_2)$ are given as $d(v_1) = |V_2| = g_1$ and $d(v_2) = |V_1| = g_1$, for all $v_1 \in V_1, v_2 \in V_2$. Thus the degree set of $G(V_1, V_2, V_3, E)$ is $\gamma = g_1$.

Now, let $p = 1$ and $g_1 > 1$. Consider the tripartite graph $G(V_1, V_2, V_3, E)$ with $|V_1| = 1$, $|V_2| = g_1 - 1$, $|V_3| = g_1$, and let there be an edge from each vertex of V_1 to every vertex of V_3, and from each vertex of V_2 to every vertex of V_3. The degrees of the vertices of this graph are as follows.

For $v_1 \in V_1$, we have $d(v_1) = |V_3| = g_1$; for all $v_2 \in V_2$, we have $d(v_2) = |V_3| = g_1$; and for all $v_3 \in V_3$, we have $d(v_3) = |V_1| + |V_2| = 1 + g_1 - 1 = g_1$. Therefore the degree set of $G(V_1, V_2, V_3, E)$ is $\gamma = g_1$.

(ii) Let $p \geq 2$. Consider the tripartite graph $G(V_1, V_2, V_3, E)$ whose

$$V_1 = \bigcup_{i=1}^{n} X_i, V_2 = \bigcup_{i=1}^{n-1} Y_i, \text{ and } V_3 = Z_1 \bigcup (\bigcup_{i=2}^{p} Z_i) \bigcup (\bigcup_{i=2}^{p} Z'_i),$$

where for all $i \neq j$, $X_i \cap X_j = \phi$, $Y_i \cap Y_j = \phi$, $Z_i \cap Z_j = \phi$, $Z_i \cap Z'_j = \phi$, $Z'_i \cap Z'_j = \phi$ for all i, $1 \leq i \leq p$, $|X_i| = |Z_i| = g_i$. Also for all i, $1 \leq i \leq p-1$, we have $|Y_i| = |Z_{i+1}| = g_1 + g_2 + \cdots + g_i$.

Take an edge from each vertex of X_i to every vertex of Z_j whenever $i \geq j$; an edge from each vertex of Y_i to every vertex of Z_{i+1} for all i, $2 \leq i \leq n-1$, and an edge from each vertex of Y_i to every vertex of Z'_{i+1} for all i, $1 \leq i \leq p-1$. The following are the degrees of the vertices of the tripartite graph $G(V_1, V_2, V_3, E)$ constructed above.

For $1 \leq i \leq p$, for all $x_i \in X_i$,

$$d(x_i) = \sum_{j=1}^{i} |Z_j| = \sum_{j=1}^{i} \nu_j = \nu_1 + \nu_2 + \cdots + \nu_i.$$

For $1 \leq i \leq n-1$, for all $y_i \in Y_i$,

$$d(y_i) = |Z_{i+1}| + |Z'_{i+1}| = g_{i+1} + (g_1 + g_2 + \cdots + g_{i+1}) = g_1 + g_2 + \cdots + g_{i+1}.$$

For all $z_1 \in Z_1$, we have

$$d(z_1) = \sum_{i=1}^{p} |X_i| = \sum_{i=1}^{p} g_i = g_1 + g_2 + \cdots + g_p.$$

For $2 \leq i \leq p$, for all $z_i \in Z_i$,

$$\begin{aligned} d(z_i) &= \sum_{j=1}^{p} |X_j| + |Y_{i-1}| \\ &= \big(\sum_{j=1}^{p} d_j \big) + (g_1 + g_2 + \cdots + g_{i-1}) \\ &= (g_i + g_{i+1} + \cdots + g_p) + (g_1 + g_2 + \cdots + g_{i-1}) \\ &= g_1 + g_2 + \cdots + g_p. \end{aligned}$$

For $2 \leq i \leq p$, for all $z'_i \in Z'_i$,

$$d(z'_i) = |Y_{i-1}| = g_1 + g_2 + \cdots + g_{i-1}.$$

Therefore the degree set of the tripartite graph $G(V_1, V_2, V_3, E)$ constructed above is $\gamma = \{g_1, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i\}$. □

Corollary 46 (Pirzada, Naikoo, Dar [48]). *Every nonempty set of positive integers, except* $\{1\}$*, is the global degree set of some connected tripartite graph.*

Proof. Here we give a new, correct proof. In case $g_1 > 0$ in the proof of Theorem 44, the construction gives a connected tripartite graph.

If the global degree set of a tripartite graph is $\gamma = \{1\}$, then let u and v be two connected vertices. If we connect one of these vertices with any other vertex, then the degree of this vertex will be at least 2. □

The following algorithm Global-Tripartite is based on Theorem 44. It constructs a tripartite graph having prescribed global degree set.

Input. p: the number of elements in the prescribed degree set γ;
$\gamma = \{g_1, g_2, \ldots, g_p\}$: the prescribed degree set for $B(V_1, V_2, E)$.

Output. $M(B)$: the incidence matrix of the constructed tripartite graph (V_1, V_2, V_3, E);
n_1: the number of lines of M, that is the size of the vertex set V_1;
n_2: the number of columns of M, that is the size of the vertex set V_2.

Work variables. i, j: cycle variables;
$n_1' = \bigcup_i = 1^p |X_i|$;
$n_1' = \bigcup_i = 1^p |X_i|$;
$n_2' = \bigcup_{i=1}^{p-1} |X_i'|$;
$n_2'' = \bigcup_{i=3}^{p-1} |X_i''|$.

Global-Tripartite$(p, q, r, \delta_1, \delta_2, \delta_3)$

```
01 if a_1 == 0                                  // lines 01–06: the case a_1 = 0
02    if p == 1                          // lines 02–06: the subcase a_1 = 0 and p = 1
03       n_1 = 1
04       n_2 = 1
05       M_{n_1,n_2} = 1
06       return M
07    if p == 2                           // lines 7–16: the subcase a_1 = 0 and p = 2
08       n_1 = a_2 + 1
09       n_2 = a_2
10       for i = 1 to n_1                          // lines 10–12: initialization of M
11           for j = 1 to n_2
12               M_ij = 0
13       for i = 2 to n_1
```

```
14          for j = 1 to n_2
15              M_ij = 1
16      return M
17  if p == 3                    // lines 17–33: the subcase a_1 = 0 and p = 3
18      n'_1 = Σ_{i=2}^{p} a_i       // lines 18–23: computation of n_1 and n_2
19      n''_1 = Σ_{i=3}^{p} Σ_{j=2}^{i} -1 a_i
20      n_1 = n'_1 + n''_1
21      n'_2 = Σ_{i=1}^{n-1} a_i
22      n''_2 = Σ_{i=3}^{n} Σ_{j=2}^{i-1} a_j
23      n_2 = n'_2 + n''_2
24      for i = 1 to n_1                    // lines 24–26: initialization of M
25          for j = 1 to n_2
26              M_ij = 0
27      for i = 2 to n_1            // lines 27–32: drawing of the edges of M
28          for j = 1 to i
29              M_ij = 1
30      for i = n'_1 + 1 to n_1
31          for j = n'_2 to n_2
32              M_ij = 1
33      return n_1, n_2, M                  // line 33: return of the result
```

Theorem 47 *The running time of* Global-Bipartite *is* $\Theta(1)$ *in best case and* $\Theta(n_1 n_2)$ *in worst case.*

Different authors proved that signed tripartite graphs have similar properties, then tripartite graphs.

Theorem 48 (Pirzada, Dar [38]) *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a nonempty set of positive integers. Then there exists a connected signed tripartite graph with signed global degree set* $\left\{ \sum_{i=1}^{1} g_i, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i \right\}$.

Proof. See [38]. □

The next result follows from Theorem 48 by interchanging positive edges with negative ones.

Corollary 49 (Pirzada, Dar [38]) *Every set of negative integers is the global degree set of some connected signed tripartite graph.*

Proof. See [38] □

Pirzada and Dar proved the following stronger assertion too.

Theorem 50 (Pirzada, Dar [38]) *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a nonempty set of positive integers. Then there exists a connected signed tripartite graph with signed global degree set* γ.

Proof. See [38]. □

5 Tripartite graphs with prescribed distributed degree set

As the following theorem shows, the existence of a corresponding tripartite graph do not require the condition $|\gamma_1| = |\gamma_2|, |\gamma_3|$.

Theorem 51 *Let* $\delta_1 = \{a_1, a_2, \ldots, a_{n_1}\}$, $\delta_2 = \{b_1, b_2, \ldots, b_{n_2}\}$, *and* $\delta_3 = \{c_1, c_2, \ldots, c_{n_3}\}$ *be nonempty sets of positive integers. positive integers with* $a_1 < a_2 < \cdots < a_{n_1}$, $b_1 < b_2 < \cdots < b_{n_2}$ *and* $c_1 < c_2 < \cdots < c_{n_3}$. *Then there exists a tripartite graph* $B = (V_1, V_2, E)$ *such that* $\delta(V_1) = \delta_1$, *and* $\delta_2(V_2) = \delta_2$ *and* $\delta_3(V_3) = \delta_3$.

Proof. Let $A = \sum_{i=1}^{n_1} a_i$ and $B = \sum_{i=1}^{n} b_i$ and $C = \sum_{i=1}^{n_3} c_i$. □

The following programDISTRIBUTED-TRIPARTITE constructs a tripartite graph having a prescribed distributed degree set.

Input. p: the number of elements in the prescribed degree set for V_1;
q: the number of elements in the prescribed degree set for V_2;
r: the number of elements in the prescribed degree set for V_3;
$\delta_1 = \{a_1, a_2, \ldots, a_p\}$: prescribed degree set for V_1;
$\delta_2 = \{b_1, b_2, \ldots, b_q\}$: prescribed degree set for V_2;
$\delta_3 = \{c_1, c_2, \ldots, c_r\}$: prescribed degree set for V_3.

Output. n_1: the size of vertex set V_1;
n_2: the size of vertex set V_2;
n_2: the size of vertex set V_3;
M_1: the incidence matrix of the constructed bipartite graph with distributed degree set δ_1, δ_2;
M_2: the incidence matrix of the constructed bipartite graph with distributed degree set δ_1, δ_3;
M_3: the incidence matrix of the constructed bipartite graph with distributed degree set δ_2, δ_3;
M: the incidence matrix of the constructed tripartite graph $B = (V_1, V_2, V_3, E)$.

Work variables. i, j: cycle variables;
α: the sum of the elements of δ_1;

β: the sum of the elements of δ_2;
τ: the sum of the elements of δ_3;
$\sigma_1\alpha$: the least common multiple of α and β;
$n_1 = \sigma/\alpha$: the number of rows of M, that is the size of V_1;
$n_2 = \sigma\beta$: the number of columns of M, that is the size of V_2;;
$\phi = (p_1, p_2, \ldots, p_\beta)$: the degree vector of V_1;
$\rho = (r_1, r_2, \ldots, r_\alpha$: the degree vector of V_2.

DISTRIBUTED-TRIPARTITE$(p, q, r, \delta_1, \delta_2, \delta_3)$

```
01 DISTRIBUTED-BIPARTITE(p, q, δ1, δ2)   // lines 01–03: computation of M1
02 N = M
03 μ2 = μ
04 DISTRIBUTED-BIPARTITE(p, r, δ1, δ3)   // lines 04–06: computation of M2
05 P = M
06 μ2 = μ
07 DISTRIBUTED-BIPARTITE(q, r, δ2, δ3)   // lines 07–09: computation of M3
08 Q = M
09 μ3 = μ
10 for i = 1 to μ1 + μ2 + μ3              // lines 10–11: initialization of M
11     Mij = 0
12 for i = 1 to μ1                        // lines 12–20: computation of M
13     for j = 1 to μ1
14         Mij = Nij
15 for i = 1 to μ2
16     for j = 1 to μ2
17     Mμ1+i,μ1+j = Pij
18 for i = 1 to μ3
19     for j = 1 to μ3
20     Mμ1+μ2+i,μ1+μ2+j = Qij
21 Σ = μ1 + μ2 + μ3                       // lines 21–21: computation of Σ
22 return Σ, M                           // lines 22–22: return of the results
```

Theorem 52 *The running time of* DISTRIBUTED-TRIIPARTITE *is in all cases cases* $\Theta(\Sigma^2)$.

Proof. The deciding part of the running time is required by lines 10–12. □

In 2007 Pirzada and Dar proved the following result on the global degree set of signed tripartite graphs.

Theorem 53 (Pirzada, Dar [38]) *Let* $\gamma = \{g_1, g_2, \ldots, g_p\}$ *be a nonempty set of positive integers. Then there exists a connested signed tripasrtite graph* G *whose global degre set is* $\sum_{i=1}^{1} g_i, \sum_{i=1}^{2} g_i, \ldots, \sum_{i=1}^{p} g_i$.

Proof. See [38]. □

Corollary 54 (Pirzada, Dar [38]) *Every set of negative integers is the global degree set of a connected tripartite signed graph.*

Proof. See [38]. □

Theorem 55 (Pirzada, Dar [38]) *Every set of integers is the global signed degree set of some connected signed tripartite graph.*

Proof. See [38]. □

6 Simple, bipartite and tripartite digraphs with prescribed global score sets

For directed graphs there are similar results as for undirected graphs.

6.1 Simple digraphs with prescribed score sets

The following papers contain the known results on the simple digraphs having a prescribed score set: [3, 12, 18, 19, 20, 23, 22, 35, 37, 40, 44, 50, 51, 63].

A directed graph is called *asymmetric* or *oriented*, if whenever a vertex $uv \in E$, then $vu \notin E$. A complete asymmetric graph is called *tournament*. The outdegrees of the vertices of a tournament are called *scores*, and the sequence of the scores is called *score sequence*, the set of scores is called *score set*.

Reid in 1978 proved the following sufficient conditions for a tournament T to have a prescribed score set.

Theorem 56 (Reid [50])

1. *Every singleton and doubleton set of positive integers is the score set of a tournament.*
2. *Let* $a \geq 1$, $d \geq 2$ *and* $n \geq 0$ *be integers and* $\gamma = \{a, ad, ad^2, \ldots, ad^n\}$. *Then there exists a tournament* T *whose score set is* γ.
3. *Let* $a \geq 1$, $d \geq 1$ *and* $n \geq 0$ *be integers and* $\gamma = \{a, a+d, a+2d, \ldots, a+nd\}$. *Then there exists a tournament* T *whose score set is* γ.

Proof. See [50] □

Since a single vertex is also a tournament, therefore $S = 0$ is also the score set of a tournament. If $a \geq 1$ and T is the union of of T_1, consisting of a single vertex and T_2 is such $(2a+1)$-regular tournament, that the elements of T_2 win against the players in T_1, then the score set of T is $\{0, a\}$,that is the first part of Theorem 56 is true not only for positive, but also for nonnegative elements.

In the same paper [50] Reid formulated the conjecture, that any set of nonnegative integers is a score set of some tournament.

In 1986 Hager [12] continued the researches of Reid proving that any set of 3, 4 or 5 nonnegative elements are also the score sets of some tournament.

Finally in 1989 Yao [63] proved the conjecture of Reid.

Theorem 57 (Yao [63]) *Any set of nonnegative integers is the global degree set of some tournament.*

Proof. See [63]. □

Let $n \geq 1$ a positive integer and $\mu_o(\gamma)$ be the minimal order of oriented graphs having score set $\gamma = \{g_1, g_2, \ldots, g_n\}$. In 1976 Chartrand, Lesniak and Roberts proved the following assertions.

Lemma 58 (Chartrand, Lesniak and Roberts [4]). *If a is a nonnegative integer, then $\mu_o(\{a\}) = 2a + 1$.*

Lemma 59 (Chartrand, Lesniak and Roberts [4]). *If γ is a finite, nonempty set of nonnegative integers and p is an integer such that $p \geq \mu_o(\gamma)$, then there exists an asymmetric digraph D of order p such that $\gamma(D) = \gamma$.*

As a simple consequence of Lemma 59, we have the following result.

Lemma 60 (Chartrand, Lesniak and Roberts [4]). *If γ is a finite, nonempty set of nonnegative integers and p is an integer such that $p \geq \mu_0(\gamma)$, then there exists an asymmetric digraph D of order p such that $\mathcal{D}(D) = \gamma$.*

Corollary 61 (Chartrand, Lesniak and Roberts [4]). *If p is a positive integer and $\gamma = \{a_1, a_2, \ldots, a_p\}$ is a set of nonnegative integers with $a_1 < a_2 < \cdots < a_p$ and $a_1 = 0$, then $\mu_0(\gamma) = a_p + 1$.*

Lemma 62 (Chartrand, Lesniak and Roberts [4]). *If $n \geq 2$ and $1 \leq a_1 < \cdots < a_n$, then $\mu_0(a_1, a_2, \ldots, a_n) \geq 2a_1 + t$, where $t > 1$ is the least integer for which $(n + t - 2)a_1 + \binom{t}{2} \geq \sum_{i=1}^{p} a_i$.*

The main result of Chartrand, Lesniak and Roberts is the following theorem.

Theorem 63 (Chartrand, Lesniak and Roberts [4]). *Let* $p \geq 2$ *be an integer* $\gamma = (a_1, a_2, \ldots, a_p)$ *be a sequence of positive integers, and let* t *be the least integer exceeding one for which* $(p+t-2)a_1 + \binom{t}{2} \geq \sum_{i=1}^{n} a_i$. *Then*

$$\mu_0(a_1, a_2, \ldots, a_p) = \begin{cases} a_p + 1 \text{ if } a_p \geq \mu_0(a_1, a_2, \ldots, a_{p-1}), \\ 2a_1 + 1 \text{ if } a_p < \mu_0(a_1, a_2, \ldots, a_{p-1}). \end{cases}$$

Proof. The proofs of Lemma 60, Corollary 61, Lemma 62, Lemma **??** are in [4]. □

In 1983 Harary and Harzheim [17] investigated the degree sets of infinite connected graphs.

Im 2006 Pirzada and Naikoo proved the following assertion on the score sets of k-partite tournaments.

Theorem 64 (Pirzada, Naikoo [43]) *Let* $k \geq 1$, d_1, d_2, $\ldots$, d_k *be nonnegative integers with* $d_2 d_3 \, ldots d_k > 0$. *Then there exists a tripartite tournament with global score set* $\{\sum_{i=1}^{1} d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{k} d_i\}$ *except for* $p = 1$, $d_1 = 0$, *and* $p = 2$, $d_1 = 0$, $d_2 = 2$.

Proof. See [43]. □

Theorem 65 (Pirzada, Naikoo [43]) *Let* d_1, d_2, $\ldots$, d_p *be nonnegative integers with* $d_2, d_3, \ldots, d_p > 0$. *Then for every* $p \geq k \geq 2$ *then there exists a* k*-partite tournament with global score set* $\{\sum_{i=1}^{1} d_i, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{k} d_i\}$.

Proof. See [43]. □

In 2006 Dziechcińska-Halamoda, Majcher, Michael, and Skupień [8] studied the properties of sets of pairs of scores in oriented graphs.

In 2006 Pirzada, Naikoo and Chishti proved the following conditions which are sufficient for an oriented graph to have special degree sets.

Theorem 66 (Pirzada, Naikoo, Chishti [45]) *If* γ *contains one, two or three positive integers, then there exists an oriented graph whose global degree set is* γ.

Proof. See [45]. □

It is also a sufficient condition, if γ contains an arithmetical or geometrical sequence.

In 2008 Pirzada and Naikoo gave the following sufficient conditions for an oriented graph G to have the global degree set γ.

Theorem 67 (Pirzada, Naikoo [44]). *Let Let* $a > 0$, $d > 1$ *and* $n \geq 0$ *be integers and* $A = \{a, ad, ad^2, \ldots, ad^n\}$. *Then there exists an oriented graph with degree set* A *except for* $a = 1$, $d = 2$, $n > 0$ *and for* $a = 1$, $d = 3$, $n > 0$.

Theorem 68 (Pirzada, Naikoo [44]) *Let If* $n \geq 1$ *and* $a_1, a_2, \ldots, a_n$ *are nonnegative integers with* $a_1 < a_2 < \cdots < a_n$, *then there exists an asymmetric graph with* $a_n + 1$ *vertices and global degree set* $a'_1, a'_2, \ldots, a'_n$, *where*

$$a'_i = \begin{cases} a_i & \text{for } i = 1, \\ a_{i-1} + a_i + 1 & \text{for } i > 1. \end{cases}$$

In 2014 Khan [27] proved, that the problem of construction of a tournament having prescribed imbalance set is weakly NP-complete.

6.2 Bipartite digraphs with prescribed score sets

A bipartite tournament is a complete asymmetric bipartite graph. Let $\delta_1 = \{a_1, a_2, \ldots, a_p = a\}$ and $\delta_2 = \{b_1, b_2, \ldots, b_q = b\}$ be finite, nonempty, increasingly ordered sets, containing nonnegative integers, whose elements are nonnegative integers with $a_1 + b_1 > 0$.

In 1983 Wayland proved the following assertion.

Theorem 69 (Wayland [60]). *There exists a bipartite tournament* $T = (V_1, V_2, E)$ *with distributed score set* (δ_1, δ_2), *if and only if*

$$\sum_{i=1}^{p} s_i + (t - p + 1)q + \sum_{j=1}^{q} + b_j + 1 - q(b_q + 1)$$

is positive.

Proof. See [60]. □

Corollary 70 *If* $s > m + 1$, *then there exists a bipartite tournament with distributed score set* (δ_1, δ_2).

Proof. See [60]. □

Also in 1983 Petrović published the following assertion.

Theorem 71 (Petrović [36]) *The set of nonnegative integers* $\delta_1 = \{a\}$ *and* $\delta_2 = \{b_1, b_2, \ldots, b_n\}$ *form a distributed score set for some bipartite tournament if and only if one of the following conditions are satisfied:*

a) $b_1 + b + 2 + \ldots + b_n = b(n - a - 1)b_n$;

b) $b_1 + b + 2 + \ldots + b_n > (n - a + 1)b_n$;

c) $b_1 + b + 2 + \ldots + b_n = (n - a + 1)b_n + d$, $1 \leq d \leq n - a - 1$.

Proof. See [36]. □

Corollary 72 (Petrović [36], Wayland [60]). *Any nonempty set of nonnegative integers except* $\{0\}$ *is the global degree set of some bipartite tournament.*

Proof. See Petrović [36], Wayland [60]. □

6.3 Tripartite digraphs with prescribed global score sets

Let k be a positive integer and $D = (V_1, V_2, \ldots, V_k, E)$ be a k-partite oriented graph. In 2006 Pirzada, and Naikoo [42]—using an unusual definition of score sets—published sufficient conditions of the existence of 3-partite oriented graphs having special singleton sets, arithmetical and geometrical series as their prescribed global score set.

In 2007 Pirzada et al. [41] gave further sufficient conditions for the existence of oriented tripartite graphs having prescribed global score set.

Acknowledgement. The authors thank the useful remarks of the unknown referee.

References

[1] T. S. Ahuja, A. Tripathi, On the order of a graph with a given degree set. *J. Comb. Math. Comb. Comput.*, **57** (2006) 157–162. ⇒74

[2] G. Chartrand, H. Gavlas, F. Harary, M. Schultz, On signed degrees in signed graphs, *Czechoslovak Math. J.*, **44,** 4 (1994) 677–690. ⇒79

[3] G. Chartrand, R. J. Gould, S. F. Kapoor, Graphs with prescribed degree sets and girth, *Periodica Math. Hung.*, **12,** 4 (1981) 261–266. ⇒78, 99

[4] G. Chartrand, L. Lesniak, J. Roberts, Degree sets for digraphs, *Periodica Math. Hung.*, **7**, 1 (1976) 77–85. ⇒100, 101

[5] G. Chartrand, L. Lesniak, P. Zhang, *Graphs & Digraphs*, CRC Press, Boca Raton, 2011. ⇒72, 77

[6] A. A. Chernyak, Minimal graphs with a given degree set and girth (Russian), *Vestsi Akad. Navuk BSSR Ser. Fiz.-Mat. Navuk,* **1988,** 2 21–25, 123. ⇒78

[7] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein. *Introduction to Algorithms* (third edition), The MIT Press/McGraw Hill, Cambridge/New York, 2009. ⇒85

[8] Z. Dziechcińska-Halamoda, Z. Majcher, J. Michael, Z. Skupień, Extremum degree sets of irregular oriented graphs and pseudodigraphs, *Discussiones Math. Graph Theory,*, **26,** 2 (2006) 317–333. ⇒101

[9] J. A. Ellis, M. Mate-Montero, H. Müller, Serial and parallel algorithms for $(k, 2)$-partite graphs, *J. Parallel Dist. Comp.*, **22** (1994) 129–137. ⇒81
[10] P. Erdős, H. Sachs, Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl. *Wiss. Z. Martin-Luther-Univ. Halle–Wittenburg, Math.-Natur. Reihe*, **12** (1963) 251–258. ⇒77
[11] J. L. Gross, J. Yellen, P. Zhang. *Handbook of Graph Theory* (second editionI, CRC Press, Boca Raton, FL, 2014. ⇒72
[12] M. Hager. On score sets for tournaments, *Discrete Math.*, **58** (1986) 25–34. ⇒ 99, 100
[13] S. L. Hakimi, On the realizability of a set of integers as degrees of the vertices of a simple graph. *J. SIAM Appl. Math.* **10** (1962) 496–506. ⇒79
[14] S. L. Hakimi, On the degrees of the vertices of a graph, *F. Franklin Institute,* **279,** (4) (1965) 290–308. ⇒
[15] F. Harary, On the notion of balance of a signed graph, *Michigan Math. J.* **2,** 2 (1953), 143–146. ⇒78, 79
[16] F. Harary, The number of linear, directed, rooted and connected graphs, *Trans. Amer. Math. Soc*, **78,** 2 (1955) 445–463. ⇒79
[17] F. Harary, E. Harzheim, The degree sets of connected infinite graphs. *Fund. Math.*, **118,** 3 (1983) 233–236. ⇒101
[18] A. Iványi, Reconstruction of score sets, *Acta Univ. Sapientiae, Informatica*, **6,** 2 (2014) 210–229. ⇒99
[19] A. Iványi, J. Elek, Reconstruction of tournaments using the set of outdegrees (in Russian), *Heuristic Algorithms and Distributed Computations*, **1,** 4 (2014) 46–70. ⇒99
[20] A. Iványi, J. Elek, Degree sets of tournaments, *Studia Univ. Babeş-Bolyai, Informatica*, **59** (2014) 150–164. ⇒99
[21] A. Iványi, L. Lucz, T. Matuszka, G. Gombos, Score sets in multitournaments, I. Mathematical results, *Annales Univ. Sci. Budapest., Sectio Comp.*, **40** (2013) 307–320. ⇒99
[22] A. Iványi, B. M. Phong. On the unicity of the score sets of multitournaments, in: *Fifth Conference on Mathematics and Computer Science* (Debrecen, June 9–12, 2004), University of Debrecen, 2006, 10 pages. ⇒99
[23] A. Iványi, S. Pirzada, N. A. Shah, Imbalances of bipartite multitournaments, *Annales Univ. Sci. Budapest., Sectio Comp.*, **37** (2012) 215–238. ⇒99
[24] S. F. Kapoor, L. Lesniak, Degree sets for triangle-free graphs. In *Second Int. Conf. Comb. Math.* (New York, 1978), pp. 320–330, Ann. New York Acad. Sci., 319, New York Acad. Sci., New York, 1979. ⇒80
[25] S. F. Kapoor, A. D. Polimeni, C. E. Wall, Degree sets for graphs, *Fund. Math.*, **95,** 3 (1977) 189–194. ⇒73, 80
[26] F. Kárteszi, Ciclici come risoluzionidi un certoproblema di minimo, *Bol. Un. Mat. Ital.*, **15** (1960) 522–528, or *Mat. Lapok*, **11** (1960) 323–329 (in Hungarian). ⇒ 77
[27] M. A. Khan, Equal sum sequences and imbalance sets of tournaments, *arXiv*, arXiv:1402.2456v1 [math.CO] 11 Feb 2014. ⇒102

[28] S. Koukichi, H. Katsuhiro, Some remarks on degree sets for graphs. *Rep. Fac. Sci. Kagoshima Univ.* No. 32 (1999), 9–14. ⇒73

[29] P. Kumar, M. N. J. Sarma, S. Sawlani, On directed tree realization of degree sets, in: ed. by S. K. Ghost, T. Tokuyama, *WALCOM 2013*, Lecture Notes in Computer Sciemce, **7748**, 2013, 274–285. ⇒80

[30] Y. Manoussakis, H. P. Patil, Bipartite graphs and their degree sets, *Electron. Notes on Discrete Math.,* (Proceedings of the R. C. Bose Centenary Symposium on Discrete Mathematics and Applications,) **15** (2003) 125–125. ⇒75

[31] Y. Manoussakis, H. P. Patil, V. Sankar, Further results on degree sets for graphs, *Mano I. J. M. S.,* **1,** 2 (2001) 1–6. ⇒75

[32] Y. Manoussakis, H. P. Patil, V. Sankar, Further results on degree sets for graphs, *AKCE J. Graphs Combin.*, **1,** 2 (2004) 77–82. ⇒75

[33] Y. Manoussakis, H. P. Patil, On degree sets and the minimum orders in bipartite graphs, *Discussiones Math. Graph Theory,* **34,** 2 (2014) 383–390. ⇒81, 88

[34] C. M. Mynhardt, Degree sets of degree uniform graphs, *Graphs Comb.*, **1** (1985) 183–190. ⇒78

[35] S. Osawa, Y. Sabata, Degree sequuences related to degree sets, *Kokyuroki*, **1744** (2011) 151–158. ⇒99

[36] V. Petrović. On bipartite score sets, *Zbornik radova Prirodno-matematičkog Fakulteta Universitetr u Novom Sadu, Ser. Mat.*, **13** (1983) 297–303. ⇒ 102, 103

[37] S. Pirzada, *An Introduction to Graph Theory*, Universities Press, Hyderabad, India, 2012. ⇒73, 77, 99

[38] S. Pirzada, F. A. Dar, Signed degree sets in signed tripartite graphs, *Matematicki Vesnik*, **59,** 3 (2007) 121–124. ⇒96, 97, 99

[39] S. Pirzada, F. A. Dar, A. Iványi, Existence of bipartite and tripartite graphs with prescribed degree sets, *Heuristic Alg. Dist. Comp.*, **1,** 1 (2015) 62–72. ⇒ 81

[40] S. Pirzada, A. Iványi, M. A. Khan. Score sets and kings, in ed. A. Iványi, *Algorithms of Informatics, Vol. 3*, mondAt, Vác, 2013, 1337–1389. ⇒99

[41] S. Pirzada, Merajuddin, T. A. Naikoo, Score sets in oriented 3-partite graphs, *Analysis Theory Appl.*, **4** (2007) 363–374. ⇒103

[42] S. Pirzada, T. A. Naikoo, Score sets in oriented k-partite graphs, *AKCE J. Graphs Combin.*, **3,** 2 (2006) 135–145. ⇒103

[43] S. Pirzada, T. A. Naikoo, Score sets in k-partite tournaments, *J. Appl. Math. Comp.* **22,** 1–2 (2006) 237–245. ⇒101

[44] S. Pirzada, T. A. Naikoo, Score sets in oriented graphs, *Appl. Anal. Discrete Math.*, **2,** 1 (2008) 107–113. ⇒99, 102

[45] S. Pirzada, T. A. Naikoo, T. A. Chishti, Score sets in oriented bipartite graphs, *Novi Sad J. Math*, **36,** 1 (2006) 35–45. ⇒101

[46] S. Pirzada, T. A. Naikoo, F. A. Dar, Signed degree sets in signed bipartite graphs, *arXiv*, arXiv/math0609129v1 [math.CO], 5 September 2006, 5 pages. ⇒87

[47] S. Pirzada, T. A. Naikoo, F. A. Dar, Signed degree sets in signed graphs, *Czechoslovak Math. J.*, **57,** 3 (2007) 843–848. ⇒79, 80

[48] S. Pirzada, T. A. Naikoo, F. A. Dar, Degree sets in bipartite and 3-partite graphs, *Oriental J. Math. Sciences*, **1,** 1 (2007) 47–53. ⇒81, 91, 95
[49] S. Pirzada, T. A. Naikoo, F. A. Dar, A note on signed degree sets in signed bipartite graphs, *Appl. Anal. Discrete Math.*, **2,** 1 (2008) 114–117. ⇒87
[50] K. B. Reid. Score sets for tournaments, *Congressus Numer.*, **21** (1978) 607–618. ⇒99, 100
[51] K. B. Reid. Tournaments: Scores, kings, generalizations and special topics, *Congressus Numer.*, **115** (1996) 171–211. ⇒99
[52] T. A. Sipka, The orders of graphs with prescribed degree sets, *J. Graph Theory*, **4,** 3 (1980) 301–307. ⇒74
[53] A. Tripathi, S. Vijay, On the least size of a graph with a given degree set, *Discrete Appl. Math.*, **154** (2006) 2530–2536. ⇒75, 76
[54] A. Tripathi, S. Vijay, A short proof of a theorem on degree sets of graphs, *Discrete Appl. Math.*, **155** (2007) 670–671. ⇒73
[55] R. I. Tyshkevich, A. A. Chernyak, Decomposition of graphs, *Cybernetics Syst. Anal.* **21,** (1985) 231–242. In Russian: *Kibernetika*, 2 (1985) 65–74. ⇒73
[56] R. I. Tyshkevich, A. A. Chernyak, Zh. A. Chernyak, Decomposition of graphs, I, *Cybernetics Syst. Anal.*, **23,** 6 (1987), 734–745. In Russian: *Kibernetika*, 6 (1987) 12–19. ⇒73
[57] R. I. Tyshkevich, A. A. Chernyak, Zh. A. Chernyak, Decomposition of graphs, II, *Cybernetics Syst. Anal.*, **24,** 2 (1988), 137–152. In Russian: *Kibernetika*, 2 (1988) 1–12. ⇒73
[58] R. I. Tyshkevich, A. A. Chernyak, Zh. A. Chernyak, Decomposition of graphs, III, *Cybernetics Syst. Anal.*, **24,** 5 (1988), 539–550. In Russian: *Kibernetika*, 5 (1988) 1–8. ⇒73
[59] L. Volkmann, Some remarks on degree sets of multigraphs, *J. Combin. Math. Combin. Comput.*, **77** (2011) 45–49. ⇒76, 77
[60] K. Wayland, Bipartite score sets, *Canadian Math. Bull.*, **26** (1983) 273–279. ⇒102, 103
[61] P. K. Wong, Cages—a survey, *J. Graph Theory*, **6,** 1 (1982) 1–22. ⇒78
[62] Y. H. Yan, K. W. Lih, D. Kuo, G. J. Chang, Signed degree sequences in signed graphs, *J. Graph Theory*, **26**, 1 (1977) 111–117. ⇒79
[63] T. X. Yao. On Reid conjecture of score sets for tournaments. *Chinese Science Bull.*, **34** (1989) 804–808. ⇒99, 100

Received: March 8, 2015 • Revised: May 1, 2015

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **7,** 1 (2015) 107–120

DOI: 10.1515/ausi-2015-0014

Empirical study of the greedy heuristic as applied to the link selection problem

Pál PUSZTAI
Széchenyi István University
email: pusztai@sze.hu

Tamás HAJBA
Széchenyi István University
email: hajbat@sze.hu

Abstract. Behind the link selection problem there is a practical problem that aims to check efficiently the vehicles on a road network. The checking process is to be realized with license plate reading cameras for checking the valid vignette of vehicles using that part of the network. However this problem should be defined generally and the methods of obtaining a solution can be applied to a wider range of problems independent of the original problem. This paper defines the link selection problem with directed graph, it shows the NP-hard complexity and it gives a heuristic and binary integer programming models to solve the problem. These two kinds of approaches allow us to examine and qualify the heuristic. The computational results of the methods are compared with different sizes of problems.

1 Introduction

The problem of link selection as an effective traffic check was introduced in [6]. The input data of a real life situation were produced with a traffic assignment model [5], and the problem was solved with an algorithm that is based on the greedy heuristic of set cover [1, 2, 3]. It was focused on the efficiency of the

Computing Classification System 1998: G.2.3
Mathematics Subject Classification 2010: 90B20, 90C10, 90C59
Key words and phrases: link selection, greedy algorithm, binary integer programming, traffic check

monitoring when the checking process has no influence on the flow of traffic, i.e. it does not stop nor slow down the vehicles. Present work is also related to this case, the loads of the network are given.

2 The link selection problem

The link selection problem has got two similar but slightly different optimization tasks. We define the problem and its heuristic more generally than it was described in [6].

Let us suppose that G is a directed graph and P is a finite, nonempty set that contains acyclic paths in G. Every path has got a positive integer number called weight.

Task 1: For a given ratio $r \in (0, 1]$ let us select minimal number of edges from G so that $x/y \geq r$ satisfied, where x is the sum of the weights of the paths that contain at least one selected edge, and y is the sum of the weights of all paths.

Task 2: For a given integer $k > 0$ let us select k edges from G so that the sum of the weights of the paths that contain at least one selected edge is maximum.

The first task is a special set cover problem in case of $r = 1$. The speciality comes from the data. Let us consider the weights of the paths as the same number of vehicles that are using the related paths. All vehicles are considered as the set to cover and the vehicles that use an edge are considered as a subset. The second task is a max k-cover problem with the same subsets of vehicles. The set cover problem and the max k-cover problem are NP-hard [2].

The link selection problem is a special case of them, where the weights of the paths of P correspond to the set to cover, the edges correspond to the subsets, and the paths make a kind of relationship between the subsets.

3 The complexity of the link selection problem

It is shown that the 3-SAT problem can be reduced to the Task 1 and Task 2 problem as well, thus the link selection problem is NP-hard. For an arbitrary 3-SAT problem we give a proper Task 1 and Task 2 link selection problem such that the solution of the 3-SAT problem can be obtained from the solution of the link selection problem.

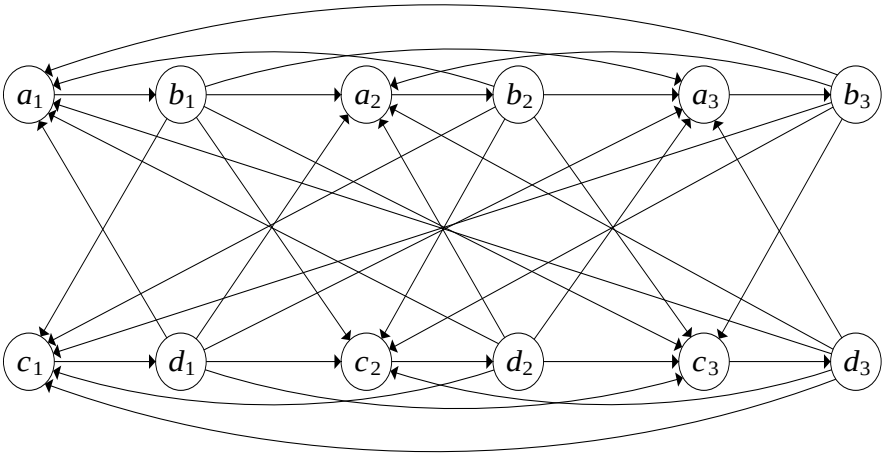

Figure 1: The G graph constructed for a three variable 3-SAT problem

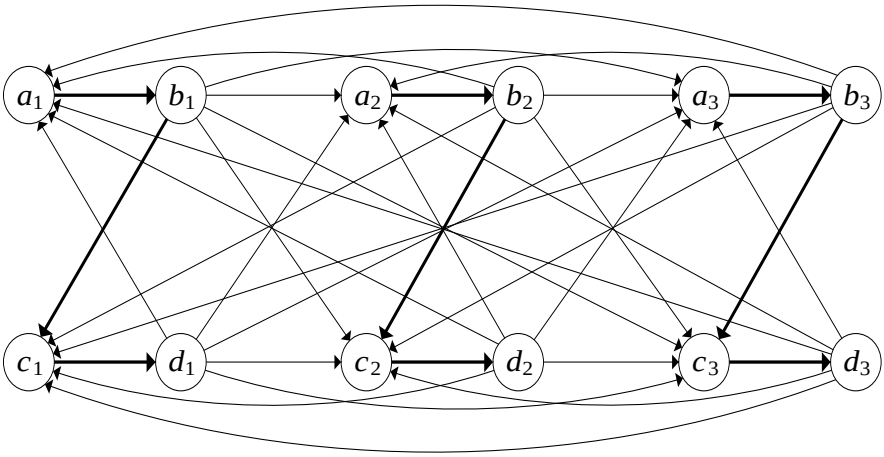

Figure 2: The 3 edge paths related to the three variables

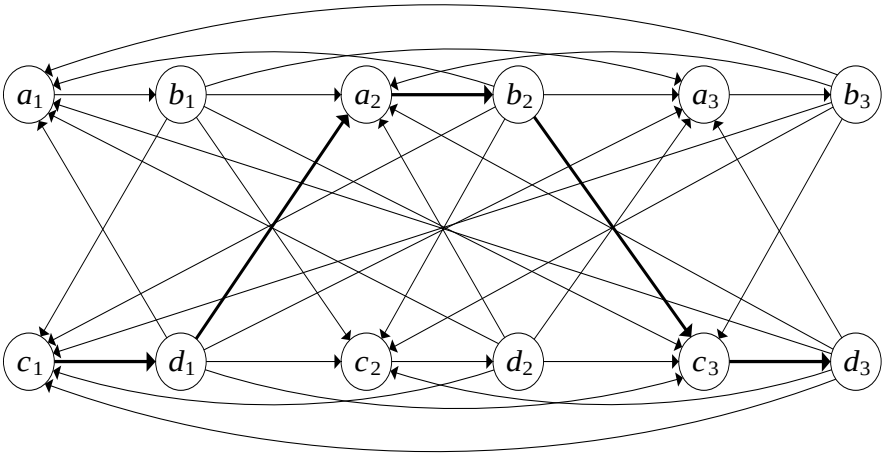

Figure 3: The 5 edge path related to the $\overline{x_1} \vee x_2 \vee \overline{x_3}$ clause

Construction: Let us suppose that there is a given 3-SAT problem with n variables and m clauses. Let $x_1, x_2, \cdots, x_n$ be the variables of the 3-SAT problem. Construct the $G = (V, E)$ directed graph as follows: for every x_i variable add 4 vertices a_i, b_i, c_i, d_i into V (so V will contain $4n$ vertices); for every i ($i = 1, \cdots, n$) add an edge from a_i to b_i (this edge corresponds to x_i) and add an edge from c_i to d_i (this corresponds to $\overline{x_i}$) into E; in addition, for every i, j, $i \neq j$ add (b_i, a_j) and (d_i, c_j) edges into E, and for every (i, j) add (b_i, c_j) and (d_i, a_j) edges as well. Note that in this G graph there is exactly one edge from every a_i and c_i vertices (see Fig. 1).

Now we make P, a set of directed paths in G. For every x_i variable add the $a_i - b_i - c_i - d_i$ 3 edges path into P (see Fig. 2). In addition, for every clause of the 3-SAT problem relate a 5 edges path as follows: let $(X \vee Y \vee Z)$ be an arbitrary clause of the 3-SAT problem; add into P the path in which the 1st edge corresponds to X, the 3rd corresponds to Y, the 5th corresponds to Z, and the 2nd and 4th ones are the edges between the proper vertices. For example for the $\overline{x_1} \vee x_2 \vee \overline{x_3}$ clause the $c_1 - d_1 - a_2 - b_2 - c_3 - d_3$ path is related (see Fig. 3). After this P will contain $n + m$ directed paths.

Lemma 1 *Let it be given an arbitrary 3-SAT problem and let* G *and* P *be the graph and the set of paths constructed before. The 3-SAT problem is satisfiable if and only if there exists a set of edges* $C \subseteq E, |C| = n$ *such that every path of* P *contains at least one edge from* C *(shortly* C *covers* P*).*

Proof. $\Rightarrow$ If the 3-SAT problem is satisfiable then we give a set of edges $C \subseteq E, |C| = n$ that covers P. If an x_i variable of the 3-SAT problem is true then add the (a_i, b_i) edge into C, otherwise (if x_i is false) add the (c_i, d_i) edge into C (thus $|C| = n$ is satisfied). Let us notice that C contains exactly one edge from every 3 edge paths and at least one edge from every 5 edge paths (as every clause has got at least one true literal and the edge related to this literal is in C), thus C covers P.

$\Leftarrow$ Let us suppose that there exists $C \subseteq E, |C| = n$ that covers P. As P has got all 3 edges $a_i - b_i - c_i - d_i$ $(i = 1, \cdots, n)$ paths and these paths are edge disjunct (they have no common edges), so every 3 edges path has got exactly one edge in C. On the other hand all paths of P start from an a_i or c_i vertex, furthermore only one edge goes to every b_i vertex, so if there is a (b_i, c_i) edge in C then replacing it with the (a_i, b_i) edge we get such a set of n edges that still covers P. Therefore it can be supposed that every edge in C is (a_i, b_i) or (c_i, d_i) type, and for every i there is exactly one edge from these (a_i, b_i) and (c_i, d_i) edges in C. Let x_i be true if the (a_i, b_i) edge is in C, otherwise (if the (c_i, d_i) edge is in C) let it be false. For these variables the 3-SAT problem will be satisfiable because every path of the 5 edge paths of the clauses contains at least one edge from C (due to the assumption) that is (a_i, b_i) or (c_i, d_i) type edge, thus the literal related to this edge (and the clause itself) will be true.

□

Theorem 2 *Task 1 of the link selection problem is NP-hard.*

Proof. Let it be given an arbitrary 3-SAT problem. Let G and P be the graph and the set of paths constructed before, let the weights of all paths equal to 1, and let $r = 1$ (i.e. we want to cover all paths of P). Due to the lemma if the solution of this link selection problem contains n edges then the 3-SAT problem is satisfiable, otherwise it is not. □

Theorem 3 *Task 2 of the link selection problem is NP-hard.*

Proof. Let it be given an arbitrary 3-SAT problem. Let G and P be the graph and the set of paths constructed before, let the weights of all paths equal to 1, and let $k = n$ (i.e. we want to cover with n edges as much as possible paths of P). Due to the lemma if the solution of this link selection problem covers all paths of P then the 3-SAT problem is satisfiable, otherwise it is not. □

4 The greedy heuristic

A greedy algorithm can be used to solve the link selection problem.

GREEDY($G, P, task, r, k$)

1. $L \leftarrow \{\}$
2. **if** $task = 1$
3. $\quad y \leftarrow$ sum up the weights of paths of P
4. **repeat**
5. $\quad$ For every edge e of G sum up the weights of paths of P that contain e
6. $\quad$ Select the edge e that has got the maximum weight
7. $\quad L \leftarrow L \cup \{e\}$
8. $\quad P \leftarrow P \setminus \{$paths that contain edge $e\}$
9. $\quad$ /* Stopping criteria */
10. $\quad$ **if** $task = 1$ /* reaching r ratio */
11. $\quad\quad u \leftarrow$ sum up the weights of paths of P
12. $\quad\quad x \leftarrow y - u$
13. $\quad\quad stop \leftarrow x/y \geq r$
14. $\quad$ **else** /* selecting k number of edges */
15. $\quad\quad stop \leftarrow |L| = k$
16. **until** $stop$
17. **return** L

Describing the complexity of the algorithm let $G = (V, E)$ graph be given and $n = |V|$. Let us suppose that $|P| = O(n^2)$. In this case we can handle all the paths between all different origin-destination pairs. If G represents a road network, the degree of vertices is limited with a small constant, thus $|E| = O(n)$. The lengths of the (acyclic) paths of P are $O(n)$. The algorithm repeats a greedy selection (line 6) until the stopping criteria becomes true. This iteration (line 4–16) runs $O(n)$ times. The most complex step of the iteration is in line 5. Based on the previous assumptions this step can be done in $O(n^3)$ time, thus we get $O(n^4)$ complexity in both cases ($task = 1, task = 2$).

Unfortunately this polynomial time algorithm does not guarantee the optimal solution. The greedy algorithm is an $\mathcal{H}(d)$-approximation algorithm for the set cover problem, where $\mathcal{H}(d) = \sum_{i=1}^{d} \frac{1}{i}$, and d is the size of the largest subset [3, 4]. It means that the solution of the greedy algorithm (the number of the selected subsets) is at most $\mathcal{H}(d)$ times larger than the optimum (the number of subsets selected by the optimal solution).

The ratio of the greedy algorithm is $1-(1-\frac{1}{k})^k \geq 1-\frac{1}{e} \approx 0.632$ for the max k-cover problem [2], which means that the number of elements covered by the greedy algorithm divided by the number of elements covered by the optimal solution is at least 0.632.

These general approximation ratios are valid for our algorithm too, namely it is $\mathcal{H}(d)$ approximation algorithm for Task 1 with $r = 1$, and it is 0.632 approximation algorithm for Task 2.

Because of the speciality of the link selection problem, the greedy algorithm gives much better solution than it is guaranteed by these ratios. In section 6 it will be shown that the greedy heuristic produces close to optimal solution for the link selection problem.

5 Binary integer programming models

The link selection problem can be solved with binary integer programming models too.

List of symbols

Parameters

n	number of the edges of G
m	number of the paths of P
w_j	weight of the path j ($j = 1, 2, \ldots, m$)
n_j	number of the edges of path j ($j = 1, 2, \ldots, m$)
j_l	the index of the lth edge of path j ($j = 1, 2, \ldots, m; l = 1, 2, \ldots, n_j$)
r	the required checking rate for Task 1 ($r \in (0, 1]$)
k	the number of required edges for Task 2 ($k > 0$)

Binary variables

x_i	$= 1$, if edge i is selected (0, otherwise) ($i = 1, 2, \ldots, n$)
y_j	$= 1$, if at least one edge of path j is selected (0, otherwise) ($j = 1, 2, \ldots, m$)

The following models can be given according to the tasks. Both models contain $n + m$ binary variables and $m + 1$ equations. Equations (1) and (4) express that a path will be selected if an edge of that path is selected. Equation (2) ensures that it reaches the required monitoring rate. Equation (5) ensures that it selects the given number of edges.

Model 1: checking with a given ratio

$$y_j \leq \sum_{l=1}^{n_j} x_{jl} \qquad (j = 1, \ldots, m) \tag{1}$$

$$r \cdot \sum_{j=1}^{m} w_j \leq \sum_{j=1}^{m} w_j y_j \tag{2}$$

$$z = \sum_{i=1}^{n} x_i \longrightarrow \min \tag{3}$$

Model 2: checking with a given number of edges

$$y_j \leq \sum_{l=1}^{n_j} x_{jl} \qquad (j = 1, \ldots, m) \tag{4}$$

$$\sum_{i=1}^{n} x_i \leq k \tag{5}$$

$$z = \sum_{j=1}^{m} w_j y_j \longrightarrow \max \tag{6}$$

6 Computational outcomes

To compare the solution of our algorithm with the optimal solution we used three test networks with 10, 20 and 80 junctions (vertices) (see Fig. 4, where the thickness of the links (e-dges) corresponds to their total weights). The greedy heuristic and Model 2 were compared for all possible values of k.

The results are shown in the figures, where the values of the X axes are the number of selected links. The checked rates of the optimal solution are shown in Fig. 6, 8, 10 corresponding to the test networks. The checked rates are given in percentage of all traffic (the sum of the weights of all paths). The differences between the checked rate of the approximate and the optimal solution are given in percentage too and shown in Fig. 5, 7, 9.

We used all shortest paths between all different junctions of the networks. The weights of the paths were generated in two different ways. In the first case the weights were the demands of travel and in the second case the weights were calculated by the demands of travel multiplied by the lengths of the paths (that resulted larger weights and larger differences between them). In the first case the weights give the *traffic* (see Traffic data in the figures) and in the second

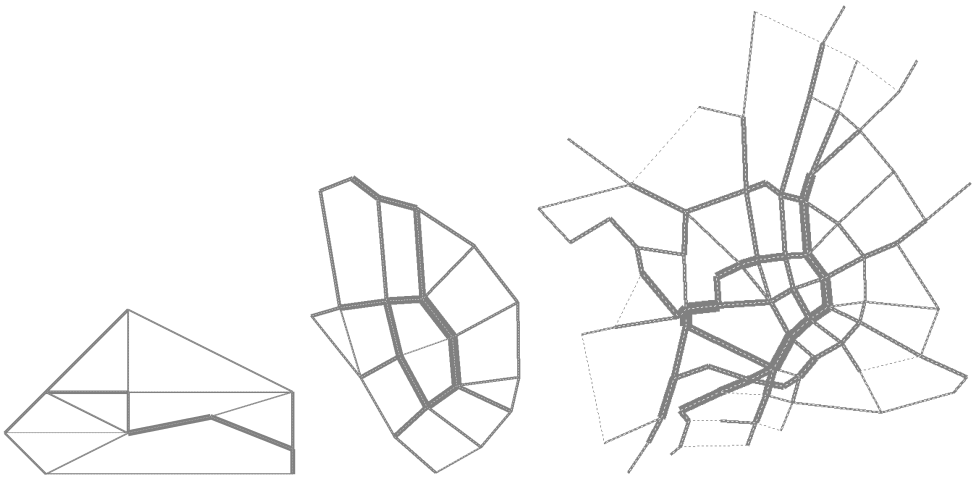

Figure 4: Test networks (10, 20 and 80 vertices) with loaded links

case the weights give the *traffic performance* (see Traf. Perf. data in figures). With using traffic performance we prefer to check those vehicles that travel longer distances on the networks (as it has been done in [6]).

In our tests the demands of travel were a random integer number in [1, 10]. The largest test network models a small part of the downtown of Budapest. It contains 80 junctions (vertices), 244 links (edges), 6320 (80*79) paths, i.e. 6564 binary variables. We used GAMS software and CPLEX solver on an average PC to solve Model 2. On our largest test network the solver required about 2 hours to compute the optimal solution for all 244 values of k, while the heuristic ran only 1 second.

	10 junctions		20 junctions		80 junctions	
Difference	Traffic	Traf.Perf.	Traffic	Traf.Perf.	Traffic	Traf.Perf.
Maximum	1.879	1.442	2.477	2.247	1.819	2.394
Average	1.037	0.671	0.587	0.596	0.397	0.250

Table 1: The maximum and the average differences between the solutions

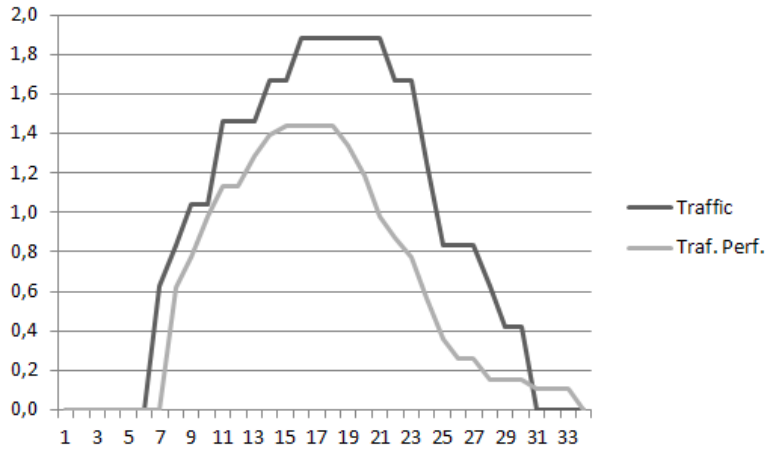

Figure 5: The difference of the solutions on a network with 10 junctions

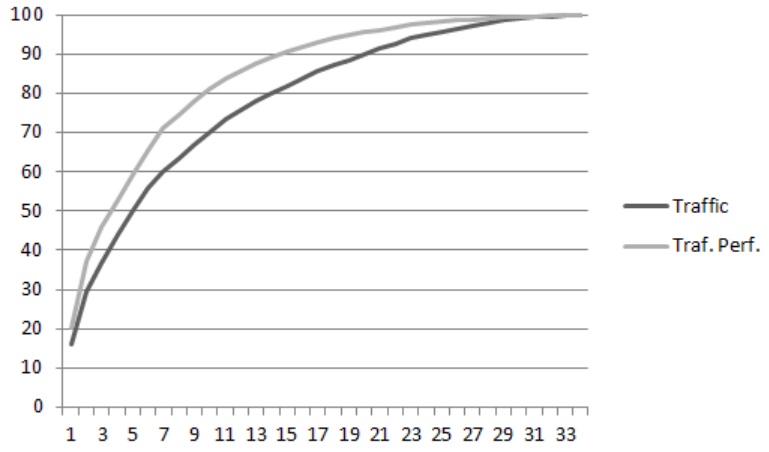

Figure 6: The optimal solution on a network with 10 junctions

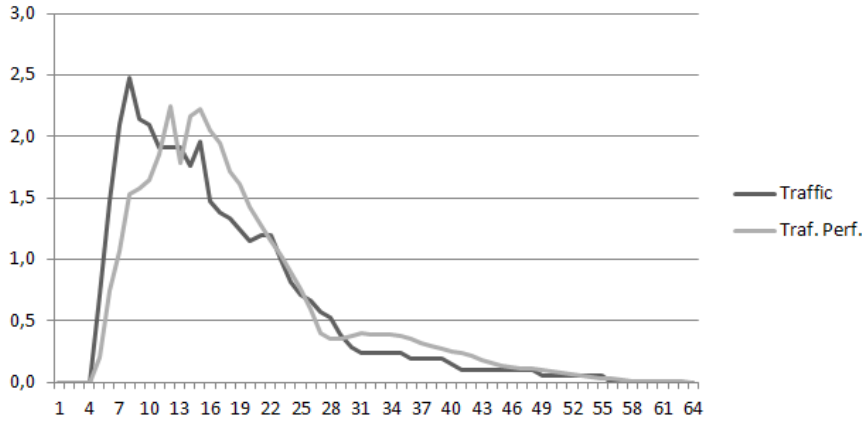

Figure 7: The difference of the solutions on a network with 20 junctions

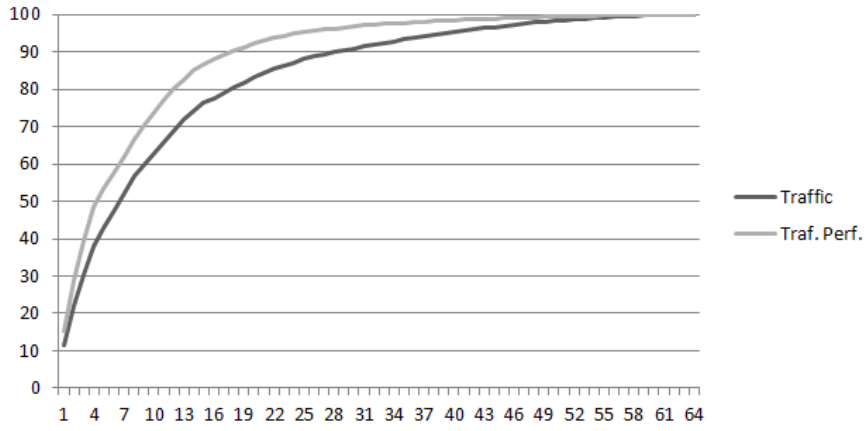

Figure 8: The optimal solution on a network with 20 junctions

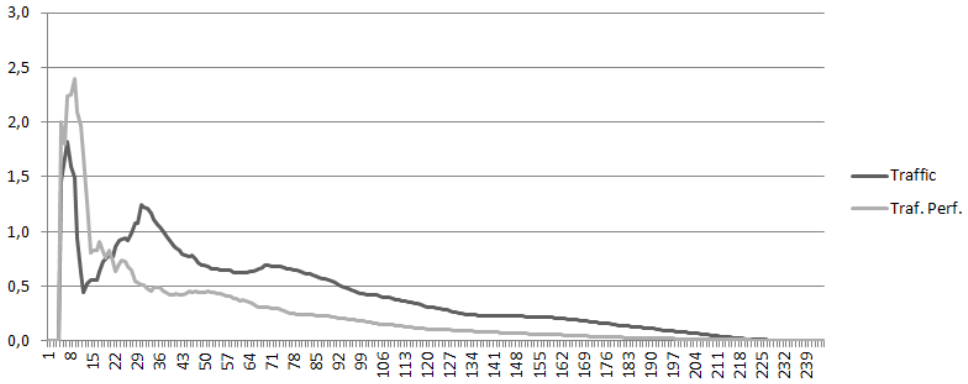

Figure 9: The difference of the solutions on a network with 80 junctions

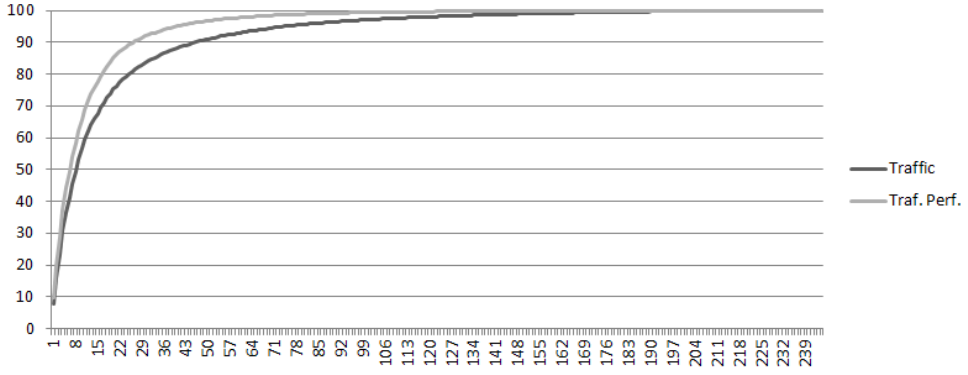

Figure 10: The optimal solution on a network with 80 junctions

The results show that:

- The maximum difference between the solution of the greedy algorithm and the optimal solution was below 2.5%, but the average difference was usually below 1% (see Tab. 1).
- Increasing the size of the problem or the weights of the paths, the functions of the optimal solution became steeper (see Fig. 6, 8, 10). It means that we need proportionally less number of links to reach the same checked ratio.
- Selecting one or all links obviously gives optimal solution, so it is expected that selecting some or almost all links also results good solution. The shapes of the functions of the maximum difference have proved this idea. Additionally it can be seen that the parts of the worst cases are not too long and they occurred earlier with increasing the size of the problem.

7 Conclusion

The link selection problem is an NP-hard optimization problem. The goal of the original real life problem was to check efficiently the vehicles on a road network. The checking process is realized in such way that it does not change the flow of traffic. A greedy heuristic was applied to solve a large size problem, but the qualification of this heuristic was demonstrated only on small (10 loaded links) networks [6].

The optimal solution of the link selection problem can be calculated with binary integer programming models as well. With this method the size of the investigated problems could be increased, but it is bounded by the capability of the applied solver contrary to the heuristic that is always usable.

In this study the greedy heuristic has been described generally for the link selection problem and it was tested on different sized problems. The results were compared with the optimal solutions that were calculated with binary integer programming models presented here. The comparison shows that the solution of this fast heuristic is much closer to the optimal solution than it is guaranteed by the general approximation ratio.

References

[1] V. Chvátal, A greedy heuristic for the set-covering problem, *Math. Oper. Res.,* **4,** 3, (1979) 233–235. ⇒107

[2] D. S. Hochbaum (Ed.), *Approximation Algorithms for NP-hard Problems,* PWS Publishing Company, Boston, Mass., 1997. ⇒107, 108, 113

[3] D. B. Johnson, Approximation algorithms for combinatorial problems, *J. Comp. System Sci.*, **9,** 3 (1974) 256–278. ⇒107, 112

[4] L. Lovász, On the ratio of optimal integral, and fractional covers *Discrete Math.,* **13,** 4 (1975) 383–390. ⇒112

[5] L. Marton, P. Pusztai, On modelling and computing traffic assignment, *Proc. EURO XVII. European Conf. Operational Research,* Budapest, Hungary, 2000, pp. 114. ⇒107

[6] P. Pusztai, An application of the greedy heuristic of set cover to traffic checks, *CEJOR Cent. Eur. J. Oper. Res.*, **16,** 4 (2008) 407–414 ⇒107, 108, 115, 119

Received: January 9, 2015 • Revised: May 1, 2015

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University

DE GRUYTER OPEN

Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: `http://www.acta.sapientia.ro`.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (`acta-inf@acta.sapientia.ro`) only, using the LaTeX style and sample file at the address `http://www.acta.sapientia.ro`. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (`http://www.acm.org/about/class/1998`) and AMS Mathematics Subject Classification codes (`http://www.ams.org/msc/`).

References should be listed alphabetically based on the Intructions for Authors given at the address `http://www.acta.sapientia.ro`.

Illustrations should be given in Encapsulated Postscript (eps) format.

One issue is offered each author free of charge. No reprints will be available.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Gloria Printing House
Director: Péter Nagy

ISSN 1844-6086
http://www.acta.sapientia.ro