The minimum maximal k-partial-matching problem

Ignacio Garcia-Vargas · Raouf Senhadji-Navarro

Abstract In this paper, we introduce a new problem related to bipartite graphs called minimum maximal k-partial-matching (MMKPM) which has been modelled by using a relaxation of the concept of matching in a graph. The MMKPM problem can be viewed as a generalization of the classical HITTING SET and SET COVER problems. This property has been used to prove that the MMKPM problem is NP-Complete. An integer linear programming formulation and a greedy algorithm have been proposed. The problem can be applied to the design process of *finite state machines with input multiplexing* for simplifying the complexity of multiplexers.

Keywords Bipartite graph · Partial-matching · NP-completeness · Hitting set · Set Cover · Finite state machine

1 Introduction

Bipartite graphs are widely used in practical applications as a model of the interaction between two different types of objects [1]. Classical NP-hard problems as HITTING SET (HS) or SET COVER (SC) can be modelled with bipartite graphs.

The HS problem can be simply described as follows. Given a collection $C = \{S_1, S_2, \ldots, S_n\}$ of nonempty subsets of a finite set S, the goal is to find the smallest subset $H \subseteq S$ that hits every set of C, i.e., $S_i \cap H \neq \emptyset$ for all $S_i \in C$. This problem can be modelled with a bipartite graph whose right vertices represent the elements of S, whose left vertices represent the subsets $S_i \in C$, and whose edges represent

the membership relation between elements and subsets. In this graph, the aim of the HS problem is to find the smallest subset of right vertices that cover all left vertices. Similarly, the SC problem can be modelled by interchanging the role of the left and right vertices. It is known that the SC problem is equivalent to the HS problem [2].

In this paper, we present a new problem related to bipartite graphs called MINIMUM MAXIMAL K- PARTIAL- MATCHING which can be informally described as follows. Given a bipartite graph, the aim is to form k sets by selecting for each left vertex all of its adjacent right vertices up to k, and adding each of them to one different set. The goal of the problem is to minimize the sum of cardinalities of these sets. This problem has been described by using a relaxation of the concept of matching in a graph.

In the case of $k = 1$, the purpose of the MMKPM problem is to find the subset of the right-vertices with minimum cardinality that cover all left-vertices. Clearly, this problem is equivalent to the HS and SC problems when they are modelled with bipartite graphs. Therefore, the MMKPM problem can be viewed as a generalization of the HS and SC problems.

The original motivations for considering MMKPM problem arose in the context of digital design of *Finite State Machines with Input Multiplexing* (FSMIM) [7]. The complexity of the multiplexers of a FSMIM can be reduced by an appropriate assignment of inputs to multiplexers. The problem of finding the best assignment can be mapped onto the MMKPM problem.

The remainder of this paper is organized as follows. We provide the formal definition of the presented problem in Sect. 2. Its NP-completeness is proved in Sect. 3. An Integer Linear Programming (ILP) formulation and a greedy algorithm are presented in Sects. 4 and 5, respectively. We provide experimental results obtained by the proposed formulation and algorithm in Sect. 6. Finally, a practical application is shown in Sect. 7.

2 Definitions and problem statement

Let $G = (U \cup V, E)$ be a bipartite graph with edge set $E \subseteq U \times V$. The degree of a vertex $s \in U \cup V$ is denoted by $d_G(s)$, and the set of the edges in E which are incident with s is denoted by $E(s)$. The minimum and maximum degree of the vertices in U are denoted by $\delta_U(G) = \min\{d_G(u)|u \in U\}$ and $\Delta_U(G) = \max\{d_G(u)|u \in U\}$, respectively. Finally, $V(E')$ and $U(E')$ denote the set of vertices in V and U, respectively, which are incident with any edge of of $E' \subseteq E$.

A matching in G is a set of edges $M \subseteq E$ such that no two edges in M share a common endpoint. Different relaxations of this definition have been proposed for bipartite graphs [4].

In this paper, we define a *partial-matching* in a bipartite graph G as a set of edges, $E' \subseteq E$, such that no two edges in E' share a common endpoint in U. Note that this definition allows the edges in E' to share common endpoints in V. Let us say that a partial-matching is *perfect* if each vertex in U is incident with exactly one edge in the partial-matching. Note that a perfect partial-matching contains the largest possible number of edges.

Let us define a *k-partial-matching* in a bipartite graph G as a collection $P = \{P_1, P_2, \ldots, P_k\}$ of k disjoint partial-matchings, i.e., $P_i \cap P_j = \emptyset$ for all $i \neq j$.

Let us define a *maximal k-partial-matching* in a bipartite graph G as a k-partial-matching that contains the maximum number of edges, i.e., a k-partial-matching $P = \{P_1, P_2, \ldots, P_k\}$ such that $|\cup_{i=1}^{k} P_i|$ is maximum. It is easy to see that all the partial-matchings $P_1, P_2, \ldots, P_k$ are perfect iff $k \leq \delta_U(G)$.

The MMKPM problem can be described as follows. Given a bipartite graph $G = (U \cup V, E)$ and a positive integer $k \leq \Delta_U(G)$, the objective is to find a maximal k-partial-matching $P = \{P_1, P_2, \ldots, P_k\}$ in G such that $\sum_{i=1}^{k} |V(P_i)|$ is minimum.

3 NP-completeness

The MMKPM problem can be described as the following decision problem:

MINIMUM MAXIMAL k-PARTIAL-MATCHING
INSTANCE: Graph bipartite $G = (U \cup V, E)$, positive integer $k \leq \Delta_U(G)$, positive integer B.
QUESTION: Is there a maximal k-partial-matching $P = \{P_1, P_2, \ldots, P_k\}$ in G such that $\sum_{i=1}^{k} |V(P_i)| \leq B$?

Theorem 1 *The MMKPM problem is NP-complete*

Proof It is easy to see that MMKPM $\in$ NP since a nondeterministic algorithm needs only guess k subsets of edges $P = \{P_1, P_2, \ldots, P_k\}$ and check in polynomial time whether P is a maximal k-partial-matching and whether $\sum_{i=1}^{k} |V(P_i)| \leq B$.

The NP-completeness of MMKPM problem has been demonstrated by restriction to the HS problem allowing only instances with $k = 1$. The NP-completeness of the HS problem has been proved in [5]. The HS problem can be described as the following decision problem:

HITTING SET
INSTANCE: Collection $C = \{S_1, S_2, \ldots, S_n\}$ of nonempty subsets of a finite set S, positive integer $B \leq |S|$.
QUESTION: Is there a subset $H \subseteq S$ with $|H| \leq B$ such that H contains at least one element from each subset in C?

Let $I = ((S, C), B)$ be an instance of the HS problem where $S = \{s_1, s_2, \ldots, s_n\}$ and $C = \{S_1, S_2, \ldots, S_m\}$. We transform I into an instance of the MMKPM problem $I' = ((G, K), B)$ as follows. A bipartite graph $G = (U \cup V, E)$ is constructed by setting $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$, and by letting $(u_i, v_j) \in E$ iff $s_j \in S_i$. It is not hard to see that G can be constructed in polynomial time.

In the remaining of the proof, we restrict the instance I' to $k = 1$. Since all subset $S_i \in C$ are nonempty, $\delta_U(G) \geq 1$. So, any feasible solution of I' is a maximal 1-partial-matching $P = \{P_1\}$ where P_1 is a perfect partial-matching.

We claim that (S, C) has a hitting set $H \subseteq S$ with $|H| \leq B$ iff G has a maximal 1-partial-matching $P = \{P_1\}$ with $|V(P_1)| \leq B$. First, suppose that $H \subseteq S$ is a hitting set with $|H| \leq B$. Consider $V' = \{v_i \in V | s_i \in H\}$, since H contains at least one element from each $S_i \in C$, each vertex in U is adjacent to at least one vertex in V'.

So, a maximal 1-partial-matching $P = \{P_1\}$ can be trivially created by selecting for each vertex in U exactly one edge incident with any vertex in V'. It is easy to see that $|V(P_1)| \leq |V'| = |H| \leq B$.

Conversely, suppose that $P = \{P_1\}$ is a maximal 1-partial-matching in G with $|V(P_1)| \leq B$. Consider $H = \{s_i \in S | v_i \in V(P_1)\}$, since P_1 is a perfect partial-matching in G, for all $u_i \in U$ there exists $(u_i, v_j) \in E$ such that $v_j \in V(P_1)$. This implies that for all $S_i \in C$ there exists $s_j \in H$ such that $s_j \in S_i$. So, H is a hitting set such that $|H| = |V(P_1)| \leq B$. □

4 Integer Linear Programming formulation

In this section, we propose a 0/1 ILP formulation for the MMKPM problem. In order to represent a k-partial-matching $P = \{P_1, P_2, \ldots, P_k\}$ in a bipartite graph $G = (U \cup V, E)$, we define the sets of binary variables $x_{e,i} \in \{0, 1\}$ and $y_{v,i} \in \{0, 1\}$ as follows:

$$x_{e,i} = \begin{cases} 1 & \text{if } e \in P_i, \\ 0 & \text{otherwise.} \end{cases} \quad \forall e \in E,\ i = 1, \ldots, k$$

$$y_{v,i} = \begin{cases} 1 & \text{if } v \in V(P_i), \\ 0 & \text{otherwise.} \end{cases} \quad \forall v \in V,\ i = 1, \ldots, k$$

Then, the MMKPM problem can be formulated in the following way:

$$\text{minimize} \quad \sum_{i=1}^{k} \sum_{v \in V} y_{v,i} \tag{1}$$

$$\text{subject to} \quad \sum_{e \in E(u)} x_{e,i} \leq 1 \qquad \forall u \in U,\ i = 1, \ldots, k \tag{2}$$

$$\sum_{i=1}^{k} x_{e,i} \leq 1 \qquad \forall e \in E \tag{3}$$

$$\sum_{i=1}^{k} \sum_{e \in E(u)} x_{e,i} = \min k, d_G(u) \qquad \forall u \in U \tag{4}$$

$$x_{e,i} \leq y_{v,i} \qquad \forall e \equiv (u, v) \in E,\ i = 1, \ldots, k \tag{5}$$

$$y_{v,i} \leq \sum_{e \in E(v)} x_{e,i} \qquad \forall v \in V,\ i = 1, \ldots, k \tag{6}$$

In (1), the cardinality of $V(P_i)$ is calculated as $\sum_{v \in V} y_{v,i}$. The constraint (2) ensures that each P_i has at most one edge incident with each vertex in U, i.e., that P_i is a partial-matching. The solution is a k-partial-matching because the constraint (3) guarantees that an edge can not belong to different partial-matchings. Furthermore, the k-partial-matching of the solution is maximal due to the constraint (4) since it requires that the k-partial-matching contains the maximum number of edges incident with each vertex in U. Finally, the constraints (5) and (6) ensure that the values of the variables $x_{e,i}$ and $y_{v,i}$ are coherent. The constraint (5) guarantees that an edge $e \equiv (u, v)$ can belong to

the partial-matching P_i only if the vertex $v \in V(P_i)$, i.e., $x_{e,i}$ can be equal to 1 only if $y_{v,i} = 1$. On the other hand, the constraint (6) imposes that a vertex v can belong to $V(P_i)$ only if there exists an edge $e \equiv (u, v) \in P_i$.

5 Greedy algorithm

A natural greedy strategy to solve an instance of MMKPM is to construct the k-partial-matching by finding the largest set of edges sharing a common endpoint in V that can be added to some partial-matching and adding them to the corresponding partial matching. This procedure must be repeated until the k-partial-matching obtained is maximal.

Algorithm 1 shows an explicit description of the proposed greedy algorithm based on the above strategy. To avoid ambiguity, the subscripted notations U_G, V_G, and E_G are used to denote the vertex set U, the vertex set V, and the edge set E of a graph G, respectively. Given a bipartite graph G and a partial-matching $E' \subseteq E_G$, $\alpha_G(v, E') = \{(u, v) \in E_G(v) \mid u \notin U_G(E')\}$ is the largest set of edges incident to the vertex $v \in V_G$ such that $E' \cup \alpha_G(v, E')$ continues to be a partial-matching.

The algorithm starts with k empty partial-matchings $P = \{P_1, P_2, \ldots, P_k\}$ and, in each iteration, simply looks for the pair (v, P_i) that gives the largest set $\alpha_G(v, P_i)$. The obtained set of edges is added to P_i and deleted from G in order to ensure that the final P is a k-partial-matching. The algorithm terminates when $|E_G|$ is equal to κ, which is the number of edges that remain in G when the k-partial-matching P is maximal.

Algorithm 1 Greedy algorithm for MMKPM

Input: graph bipartite G, positive integer k
Output: maximal k-partial-matching $P = \{P_1, P_2, \ldots, P_k\}$
1: $P_i \leftarrow \emptyset$ for all $P_i \in P$
2: $\kappa \leftarrow |E_G| - \sum_{u \in U_G} \min k, d_G(u)$
3: **while** $|E_G| > \kappa$ **do**
4: $\quad (v, P_i) \leftarrow \underset{(v,P_i) \in V \times P}{\arg\max} |\alpha_G(v, P_i)|$
5: $\quad P_i \leftarrow P_i \cup \alpha(v, P_i)$
6: $\quad G \leftarrow G - \alpha_G(v, P_i)$
7: **end while**

6 Experimental results

Both the greedy algorithm and the ILP formulation have been coded using SAGE [8]. The ILP formulation has been solved using the solver Gurobi 4.6 [9] as the back end of SAGE. The experiments have been executed in an Intel(R) Core (TM) i7 CPU 860 at 2.80Ghz with 8GB of RAM running Linux 64 bits (kernel 3.0.15).

The test data set consists of 384 MMKPM instances obtained from 96 randomly generated graphs for which $|V|$ takes the values 20 and 50; $|U|$ takes the values $0.5 \cdot |V|$ and $1.5 \cdot |V|$; $\Delta_U(G)$ takes the values $0.4 \cdot |V|$ and $0.7 \cdot |V|$; and finally, $|E|$ takes the

values $0.4 \cdot |U| \cdot \Delta_U(G)$ and $0.7 \cdot |U| \cdot \Delta_U(G)$. For each set of values, six different graphs have been generated without any isolated vertex. Each graph has been used to generate different instances of MMKPM for which k takes the four equidistant values between 2 and $\Delta_U(G)$, both included. The experiments have been executed with a time limit of 2400 seconds.

Table 1 summarizes the results obtained by the ILP formulation and the greedy algorithm. The table shows different statistics calculated over the six random instances of each problem size (determined by $|U|$, $|V|$, $|E|$, $\Delta_U(G)$, and k). Hereinafter, we refer to the problem instances solved to optimality by the ILP formulation as "solved

Table 1 Experimental results obtained by ILP formulation and greedy algorithm

Grafo					ILP				Greedy		
$\|V\|$	$\|U\|$	$\|E\|$	$\Delta_U(G)$	k	Solved	Gap	R-Gap	Time	O-Gap	Gap	Time
20	10	32	8	2	6	–	0.83	0.05	2.80	–	0.01
20	10	32	8	4	6	–	0.00	0.06	0.98	–	0.02
20	10	32	8	6	6	–	0.00	0.08	0.00	–	0.02
20	10	32	8	8	6	–	0.00	0.10	0.00	–	0.03
20	10	56	8	2	6	–	3.70	0.06	6.94	–	0.01
20	10	56	8	4	6	–	6.52	0.25	8.27	–	0.02
20	10	56	8	6	6	–	4.66	0.64	4.45	–	0.03
20	10	56	8	8	6	–	2.38	0.59	2.34	–	0.03
20	10	56	14	2	6	–	4.76	0.06	0.00	–	0.01
20	10	56	14	6	6	–	0.00	0.12	2.83	–	0.02
20	10	56	14	10	6	–	0.88	0.32	0.00	–	0.04
20	10	56	14	14	6	–	0.00	0.25	0.00	–	0.05
20	10	98	14	2	6	–	18.58	0.11	0.00	–	0.01
20	10	98	14	6	6	–	18.44	2.49	5.73	–	0.03
20	10	98	14	10	6	–	14.09	24.93	5.85	–	0.05
20	10	98	14	14	6	–	17.78	123.62	2.00	–	0.07
20	30	96	8	2	6	–	6.19	0.16	9.90	–	0.02
20	30	96	8	4	6	–	9.10	1.69	12.86	–	0.03
20	30	96	8	6	6	–	3.17	3.18	7.26	–	0.03
20	30	96	8	8	6	–	1.59	1.36	2.31	–	0.04
20	30	168	8	2	6	–	12.16	0.45	10.10	–	0.02
20	30	168	8	4	6	–	14.86	6.00	16.37	–	0.03
20	30	168	8	6	6	–	20.23	380.28	19.41	–	0.05
20	30	168	8	8	1	9.60	23.08	1448.73	16.13	27.16	0.06
20	30	168	14	2	6	–	12.64	0.71	10.00	–	0.01
20	30	168	14	6	6	–	17.17	172.53	17.13	–	0.05
20	30	168	14	10	6	–	13.65	270.73	9.67	–	0.07
20	30	168	14	14	6	–	7.65	24.54	0.76	–	0.08
20	30	294	14	2	6	–	22.97	0.78	10.25	–	0.01

Table 1 Continued

Grafo					ILP				Greedy		
$\|V\|$	$\|U\|$	$\|E\|$	$\Delta_U(G)$	k	Solved	Gap	R-Gap	Time	O-Gap	Gap	Time
20	30	294	14	6	6	–	20.01	232.51	12.97	–	0.07
20	30	294	14	10	0	15.60	–	–	–	30.05	0.11
20	30	294	14	14	0	19.12	–	–	–	29.77	0.13
50	25	200	20	2	6	–	11.66	0.62	11.87	–	0.02
50	25	200	20	8	6	–	2.11	114.74	11.40	–	0.19
50	25	200	20	14	6	–	0.00	5.71	2.62	–	0.30
50	25	200	20	20	6	–	0.00	2.18	0.00	–	0.42
50	25	350	20	2	6	–	16.65	1.10	11.95	–	0.02
50	25	350	20	8	0	13.27	–	–	–	21.81	0.22
50	25	350	20	14	0	17.48	–	–	–	28.47	0.50
50	25	350	20	20	0	17.83	–	–	–	27.43	0.66
50	25	350	35	2	6	–	19.59	1.24	13.22	–	0.02
50	25	350	35	13	0	13.50	–	–	–	25.99	0.47
50	25	350	35	24	1	4.37	0.00	766.37	10.71	11.34	0.69
50	25	350	35	35	5	2.31	0.39	101.41	0.77	2.31	0.93
50	25	612	35	2	6	–	26.16	5.52	2.38	–	0.02
50	25	612	35	13	0	26.93	–	–	–	32.64	0.53
50	25	612	35	24	0	37.20	–	–	–	38.82	1.30
50	25	612	35	35	0	44.27	–	–	–	29.64	1.56
50	75	600	20	2	6	–	19.76	11.22	11.95	–	0.05
50	75	600	20	8	0	27.63	–	–	–	44.02	0.49
50	75	600	20	14	0	20.75	–	–	–	33.32	0.65
50	75	600	20	20	0	10.40	–	–	–	17.10	0.79
50	75	1050	20	2	6	–	27.05	32.26	10.99	–	0.06
50	75	1050	20	8	0	31.57	–	–	–	41.67	0.70
50	75	1050	20	14	0	55.13	–	–	–	56.06	1.38
50	75	1050	20	20	0	77.42	–	–	–	54.39	1.58
50	75	1050	35	2	6	–	28.67	59.34	9.56	–	0.06
50	75	1050	35	13	0	58.68	–	–	–	55.87	1.35
50	75	1050	35	24	0	80.00	–	–	–	40.43	1.66
50	75	1050	35	35	0	81.33	–	–	–	23.05	2.10
50	75	1837	35	2	6	–	35.57	111.99	7.64	–	0.06
50	75	1837	35	13	0	60.00	–	–	–	48.87	1.76
50	75	1837	35	24	0	–	–	–	–	–	3.75
50	75	1837	35	35	0	–	–	–	–	–	3.78
35.0	35.0	438.6	19.2	10.6	0.64	34.57	10.82	49.76	6.80	34.17	0.46

instances"; on the other hand, we refer to the instances that could not be solved by the ILP formulation within the imposed time limit as "non-solved instances". In the case of ILP formulation, "Solved" represents the number of solved instances; "Gap", the

average final percentage optimality gap for non-solved instances; "R-Gap", the average percentage gap between the root relaxation and the incumbent for solved instances; and "Time", the average amount of time in seconds spent by solved instances. In the case of the greedy algorithm, "Gap" represents the average percentage gap between the solution of the greedy algorithm and the best bound found by the ILP formulation for non-solved instances; "O-Gap", the average percentage gap between the solution of the greedy algorithm and the optimal objective function value found by the ILP formulation for solved instances; and "Time", the average amount of time in seconds spent by the greedy algorithm. The last row shows the average of the different measures for all experimental results.

The ILP formulation found an optimal solution in 247 instances. It can be observed that the number of solved instances tends to decrease with k, specially for hight values of $\Delta_U(G)$. This shows that k has a significant influence on the complexity of the problem, as expected. The "R-Gap" is less than 5 % in the 40 % of the cases and equal to 0 in the 33 %. The greedy algorithm found an optimal solution in 103 instances. As expected, the greedy algorithm is faster than the ILP approach in all experiments. However, the "Gap" in the ILP formulation is less than in the greedy algorithm in the 70 % of the non-solved instances (the 12 instances related to the last two problem sizes have been excluded because the solver could not calculate any incumbent within the imposed time limit). The average percentage difference between the gaps obtained by the ILP formulation and greedy algorithm in the consider cases was 36 %. The average "O-Gap" is equal to 6.80 % which corresponds to an approximation ratio of 1.07. In the case of non-solved instances, the average "Gap" of the greedy algorithm is 34.17 % which correspond to an approximation ratio of at most 1.52. The approximation ratio is lower than 2.28 in all experiments.

7 Practical application

A FSMIM is an special type of *Finite State Machine* (FSM) oriented to the memory-based implementation of FSMs [3]. A FSM is a behavioural model composed of inputs and outputs, where the outputs depend on the current and previous inputs. A FSM is defined as a set of states where the present state determines the input-output relationship at each instant of time. The next state and outputs of the FSM are determined by the present state and inputs. The implementation of FSMs is an important area of research on Digital Design.

The aim of FSMIMs is to take advantage of the fact that, usually, a state is sensitive only to a subset of inputs (note that the number of these inputs can be different for each state). A multiplexer is a digital device with n input lines (hereinafter called channels) that allows to select one channel to connect it to the output line. The complexity of multiplexers grows with the number of channels. In a FSMIM, a set of multiplexers selects the appropriate subset of inputs for the current state. The largest subset of inputs determines the number of multiplexers. Therefore, there can exist states that are sensitive to a smaller number of inputs than the number of multiplexers. For these states, the FSMIM ignores the inputs selected by the unnecessary multiplexers. One of the most critical issues in the FSMIM design process is to simplify the complexity of

$$
\begin{aligned}
s_1 &= \{i_{10}\} \\
s_2 &= \{i_{10}, i_{11}, i_5, i_7\} \\
s_3 &= \{i_6\} \\
s_4 &= \{i_8, i_{10}, i_{11}, i_5, i_7\} \\
s_5 &= \{i_1, i_2, i_3, i_4, i_{10}\} \\
s_6 &= \{i_{11}, i_5, i_7\} \\
s_7 &= \{i_{10}, i_6\} \\
s_8 &= \{i_9, i_{10}, i_5\} \\
s_9 &= \{i_1, i_2, i_3, i_4, i_7, i_8, i_{10}\} \\
s_{10} &= \{i_8, i_{11}, i_5, i_6, i_7\} \\
s_{11} &= \{i_1, i_2, i_3, i_4, i_5, i_{10}\} \\
s_{12} &= \{i_{11}\}
\end{aligned}
$$

(a)

$$
\begin{aligned}
M_1 &= \{i_4, i_7\} \\
M_2 &= \{i_9, i_8\} \\
M_3 &= \{i_2, i_6\} \\
M_4 &= \{i_{10}\} \\
M_5 &= \{i_1, i_{11}\} \\
M_6 &= \{i_5, i_4\} \\
M_7 &= \{i_3\}
\end{aligned}
$$

(b)

Fig. 1 An example of application of the MMKPM problem for simplifying the multiplexers of a FSMIM: **a** inputs of each state, and **b** connection of inputs to multiplexers

the multiplexers by reducing its number of channels [3,7]. The problem of minimizing the number of channels of all multiplexers can be modelled as the MMKPM problem as follows:

Let U and V represent the set of states and inputs of a FSMIM, respectively. A bipartite graph $G = (U \cup V, E)$ is created such that $(u, v) \in E$ iff the state $u \in U$ is sensitive to the input $v \in V$. The number of multiplexers is given by $k = \Delta_U(G)$. Each maximal k-partial-matching $P = \{P_1, P_2, \ldots, P_k\}$ in G represents a set of multiplexers $M = \{M_1, M_2, \ldots, M_k\}$ of the FSMIM: the set of inputs connected to M_i is given by $V(P_i)$, each edge $(u, v) \in P_i$ implies that v is the input selected by M_i for the state u, and the total number of channels of M is given by $\sum_{i=1}^{k} |V(P_i)|$. All appropriate inputs for each state can be selected if and only if P is maximal and $k = \Delta_U(G)$. So, the set of multiplexers with the least number of channels is given by the minimum maximal k-partial-matching in G.

The MMKPM problem has been applied to a set of FSM benchmarks [6] by solving the ILP formulation with Gurobi 4.6 [9]. As an example, we present the results obtained by the test case *sand*. This FSM has 32 states and 11 inputs. However, the states sensitive to the same subset of inputs are represented by the same vertex in U. This reduces the significant set of states to 12. Figure 1 shows the subset of inputs related to each state (Fig. 1a) and the connection of inputs to multiplexers established by the solution (Fig. 1b). The largest subset of inputs has 7 inputs (this determines the value of k). The total number of channels of the solution is equal to 12. As can be observed, the optimal solution connects the input i_4 to multiplexer M_1 and M_6. Therefore, no solution can connect each input to exactly one multiplexer. The solution can be interpreted by observing, e.g., the state s_{10}: the multiplexers M_2, M_5, M_6, M_3, and M_1 select the inputs i_8, i_{11}, i_5, i_6, and i_7, respectively, when s_{10} is the current state. The FSMIM is designed in such way that the channels selected by the multiplexers M_4 and M_7 have no influence on the state s_{10}.

References

1. Asratian, A.S., Denley, T.M.J., Häggkvist, R.: Bipartite Graphs and Their Applications. Cambridge University Press, New York (1998)
2. Ausiello, G., D'Atri, A., Protasi, M.: Structure preserving reductions among convex optimization problems. J. Comput. Syst. Sci. **21**(1), 136–153 (1980)
3. Garcia-Vargas, I., Senhadji-Navarro, R., Jimenez-Moreno, G., Civit-Balcells, A., Guerra-Gutierrez, P.: Rom-based finite state machine implementation in low cost fpgas. In: IEEE International Symposium on Industrial Electronics, 2007. ISIE 2007, pp. 2342–2347 (2007)
4. Harvey, N.J.A., Ladner, R.E., Lovász, L., Tamir, T.: Semi-matchings for bipartite graphs and load balancing. J. Algorithms **59**(1), 53–78 (2006)
5. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations., pp. 85–103. Plenum Press, New York (1972)
6. McElvain, K.: IWLS'93 benchmark set: Version 4.0 (1993)
7. Senhadji-Navarro, R., Garcia-Vargas, I., Jimenez-Moreno, G., Civit-Ballcels, A.: Rom-based fsm implementation using input multiplexing in fpga devices. Electron. Lett. **40**(20), 1249–1251 (2004)
8. Stein, W.: Sage: Open Source Mathematical Software. The Sage Group (2008). http://www.sagemath.org
9. Gurobi Optimization, Inc.: Gurobi Optimizer Reference Manual. http://www.gurobi.com