



HAL
open science

Effective heuristics for matchings in hypergraphs

Fanny Dufossé, Kamer Kaya, Ioannis Panagiotas, Bora Uçar

► **To cite this version:**

Fanny Dufossé, Kamer Kaya, Ioannis Panagiotas, Bora Uçar. Effective heuristics for matchings in hypergraphs. SEA2 2019 - International Symposium on Experimental Algorithms - Special Event, Jun 2019, Kalamata, Greece. pp.248-264, 10.1007/978-3-030-34029-2_17. hal-02417475

HAL Id: hal-02417475

<https://inria.hal.science/hal-02417475>

Submitted on 18 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Effective heuristics for matchings in hypergraphs

Fanny Dufossé¹, Kamer Kaya²[0000-0001-8678-5467],
Ioannis Panagiotas³[0000-0002-1081-2411], and Bora Uçar^{3,4}[0000-0002-4960-3545]

¹ Inria Grenoble Rhône-Alpes, fanny.dufosse@inria.fr

² Sabanci University, Istanbul, Turkey, kaya@sabanciuniv.edu

³ ENS Lyon, France, ioannis.panagiotas@ens-lyon.fr

⁴ CNRS and LIP (UMR5668, CNRS - ENS Lyon - UCB Lyon 1 - INRIA), Lyon,
France, bora.ucar@ens-lyon.fr

Abstract. The problem of finding a maximum cardinality matching in a d -partite, d -uniform hypergraph is an important problem in combinatorial optimization and has been theoretically analyzed. We first generalize some graph matching heuristics for this problem. We then propose a novel heuristic based on tensor scaling to extend the matching via judicious hyperedge selections. Experiments on random, synthetic and real-life hypergraphs show that this new heuristic is highly practical and superior to the others on finding a matching with large cardinality.

Keywords: d -dimensional matching · Tensor scaling · Matching in hypergraphs · Karp-Sipser heuristic.

1 Introduction

A hypergraph $H = (V, E)$ consists of a finite set V and a collection E of subsets of V . The set V is called vertices, and the collection E is called hyperedges. A hypergraph is called d -partite and d -uniform, if $V = \bigcup_{i=1}^d V_i$ with disjoint V_i s and every hyperedge contains a single vertex from each V_i . A matching in a hypergraph is a set of disjoint hyperedges. In this paper, we investigate effective heuristics for finding large matchings in d -partite, d -uniform hypergraphs.

Finding a maximum cardinality matching in a d -partite, d -uniform hypergraph for $d \geq 3$ is NP-Complete; the 3-partite case is called the MAX-3-DM problem [27]. This problem has been studied mostly in the context of local search algorithms [24], and the best known algorithm is due to Cygan [8] who provides $((d+1+\varepsilon)/3)$ -approximation, building on previous work [9, 21]. It is NP-Hard to approximate MAX-3-DM within 98/97 [3]. Similar bounds exist for higher dimensions: the hardness of approximation for $d = 4, 5$ and 6 are shown to be $54/53 - \varepsilon$, $30/29 - \varepsilon$, and $23/22 - \varepsilon$, respectively [22].

Finding a maximum cardinality matching in a d -partite, d -uniform hypergraph is a special case of the d -SET-PACKING problem [23]. It has been shown that d -SET-PACKING is hard to approximate within a factor of $\mathcal{O}(d/\log d)$ [23]. The maximum/perfect set packing problem has many applications, including combinatorial auctions [20] and personnel scheduling [18]. Such a matching

can also be used in the coarsening phase of multilevel hypergraph partitioning tools [6], when the input is d -uniform and d -partite, such as those used in modeling and partitioning tensors [28].

Our contributions in this paper are as follows. We propose five heuristics. The first two are adaptations of the well-known greedy [15] and Karp-Sipser [26] heuristics widely used for finding matchings in bipartite graphs. We use **Greedy^g** and **Karp-Sipser^g** to refer to these heuristics, and **Greedy** and **Karp-Sipser** for the proposed generalizations. **Greedy** traverses the hyperedge list in random order and adds a hyperedge to the matching whenever possible. **Karp-Sipser** introduces certain rules to **Greedy** to improve the cardinality. The third heuristic is inspired by a recent scaling-based approach proposed for the maximum cardinality matching problem on graphs [11–13]. The fourth heuristic is a modification of the third one that allows for faster execution time. The last one finds a matching for a reduced, $(d - 1)$ -dimensional problem and exploits it for d dimensions. This heuristic uses an exact algorithm for the bipartite matching problem. We perform experiments to evaluate the performance of these heuristics on special classes of random hypergraphs and real-life data.

Another way to tackle the problem at hand is to create the line graph G for a given hypergraph H . The line graph is created by identifying each hyperedge of H with a vertex in G , and by connecting two vertices of G with an edge, iff the corresponding hyperedges share a common vertex in H . Then, successful heuristics for computing large independent sets in graphs, e.g., KaMIS [29], can be used to compute large matchings in hypergraphs. This approach, although promising quality-wise, could be impractical. This is so, since building G from H requires quadratic run time and storage (in terms of the number of hyperedges) in the worst case. While this can be acceptable in some instances, in others it is not. We have such instances in the experiments.

The rest of the paper is organized as follows. Section 2 introduces the notation and summarizes the background material. The proposed heuristics are summarized in Section 3. Section 4 presents the experimental results and Section 5 concludes the paper.

2 Background and notation

Tensors are multidimensional arrays, generalizing matrices to higher orders. Let \mathbf{T} be a d -dimensional tensor whose size is $n_1 \times \dots \times n_d$. The elements of \mathbf{T} are shown with $\mathbf{T}_{i_1, \dots, i_d}$, where $i_j \in \{1, \dots, n_j\}$. A marginal is a $(d - 1)$ -dimensional section of a d -dimensional tensor, obtained by fixing one of its indices. A d -dimensional tensor where the entries in each of its marginals sum to one is called d -stochastic. In a d -stochastic tensor, all dimensions necessarily have the same size n . A d -stochastic tensor where each marginal contains exactly one nonzero entry (equal to one) is called a permutation tensor. Franklin and Lorenz [16] show that if a nonnegative tensor \mathbf{T} has the same zero-pattern as a d -stochastic tensor \mathbf{B} , then one can find a set of d vectors $x^{(1)}, x^{(2)}, \dots, x^{(d)}$ such that $\mathbf{T}_{i_1, \dots, i_d} \cdot x_{i_1}^{(1)} \dots x_{i_d}^{(d)} = \mathbf{B}_{i_1, \dots, i_d}$ for all $i_1, \dots, i_d \in \{1, \dots, n\}$. In fact, a multidimensional

version of the algorithm for doubly-stochastic scaling (of matrices) by Sinkhorn and Knopp [32] can be used to obtain these d vectors.

A d -partite, d -uniform hypergraph $H = (V_1 \cup \dots \cup V_d, E)$ can be naturally represented by a d -dimensional tensor. This is done by associating each tensor dimension with a vertex class. Let $|V_i| = n_i$, and the tensor $\mathbf{T} \in \{0, 1\}^{n_1 \times \dots \times n_d}$ have a nonzero element $\mathbf{T}_{v_1, \dots, v_d}$ iff (v_1, \dots, v_d) is a hyperedge of H . Then, \mathbf{T} is called the adjacency tensor of H . In H , if a vertex is a member of only a single hyperedge we call it a degree-1 vertex. Similarly, if it is a member of only two, we call it a degree-2 vertex.

In the k -out random hypergraph model, given V , each vertex $u \in V$ selects k hyperedges from the set $E_u = \{e : e \subseteq V, u \in e\}$ in a uniformly random fashion and the union of these hyperedges forms E . We are interested in the d -partite, d -uniform case, and hence $E_u = \{e : |e \cap V_i| = 1 \text{ for } 1 \leq i \leq d, u \in e\}$. This model generalizes random k -out bipartite graphs [34]. Devlin and Kahn [10] investigate fractional matchings in these hypergraphs, and mention in passing that k should be exponential in d to ensure that a perfect matching exists.

3 Heuristics for maximum d -dimensional matching

A matching which cannot be extended with more hyperedges is called *maximal*. In this work, we propose heuristics for finding maximal matchings on d -partite, d -uniform hypergraphs. For such hypergraphs, any maximal matching is a d -approximate matching. The bound is tight and can be verified for $d = 3$. Let H be a 3-partite $3 \times 3 \times 3$ hypergraph with the following hyperedges $e_1 = (1, 1, 1), e_2 = (2, 2, 2), e_3 = (3, 3, 3)$ and $e_4 = (1, 2, 3)$. The maximum matching is $\{e_1, e_2, e_3\}$, and the hyperedge $\{e_4\}$ alone forms a maximal matching.

3.1 A Greedy heuristic for Max- d -DM

There exist two variants of Greedy^g in the literature. The first one [15] randomly visits the edges and adds the current edge to the matching if both end points are available. The second one randomly visits the vertices [30], and matches the vertex with the first available neighbor, if any, visited in a random order. We adapt the first variant to our problem and call it Greedy. It traverses the hyperedges in random order and adds the current hyperedge to the matching whenever possible. Since any maximal matching is possible as its output, Greedy is a d -approximation heuristic. It obtains matchings of varying quality, depending upon the order in which the hyperedges are processed.

3.2 Karp-Sipser for Max- d -DM

A widely-used heuristic to obtain large matchings in graphs is Karp-Sipser^g [26]. On a graph, the heuristic iteratively adds a random edge to the matching and reduces the graph by removing its endpoints, as well as their edges. Whenever possible, Karp-Sipser^g does not apply a random selection but reduces the problem size, i.e., number of vertices in the graph by one via two rules:

- At any time during the heuristic, if a degree-1 vertex appears it is matched with its only neighbor.

- If a degree-2 vertex u appears with neighbors $\{v, w\}$ and no degree-1 vertex exists, u (and its edges) is removed from the current graph, and v and w are merged to create a new vertex vw whose set of neighbors is the union of those of v and w (except u). A maximum cardinality matching for the reduced graph can be extended to obtain one for the current graph by matching u with either v or w depending on vw 's match.

Both rules are optimal in the sense that they do not reduce the cardinality of a maximum matching in the current graph they are applied on. We now propose an adaptation of Karp-Sipser⁹ for d -partite, d -uniform hypergraphs, and call this heuristic Karp-Sipser. Similar to the original one, Karp-Sipser iteratively adds a random hyperedge to the matching, remove its d endpoints and their hyperedges. However, the random selection is not applied whenever hyperedges defined by the lemmas below appear.

Lemma 1. *During the heuristic, if a hyperedge e with at least $d - 1$ degree-1 endpoints appears, there exists a maximum cardinality matching in the current hypergraph containing e .*

Proof. Let H' be the current hypergraph at hand and $e = (u_1, \dots, u_d)$ be a hyperedge in H' whose first $d - 1$ endpoints are degree-1 vertices. Let M' be a maximum cardinality matching in H' . If $e \in M'$, we are done. Otherwise, assume that u_d is the endpoint matched by a hyperedge $e' \in M'$ (note that if u_d is not matched M' can be extended with e). Since u_i , $1 \leq i < d$, are not matched in M' , $M' \setminus \{e'\} \cup \{e\}$ defines a valid maximum cardinality matching for H' . \square

We note that it is not possible to relax the condition by using a hyperedge e with less than $d - 1$ endpoints of degree-1; in M' , two of e 's higher degree endpoints could be matched with two different hyperedges, in which case the substitution as done in the proof of the lemma is not valid.

Lemma 2. *During the heuristic, let $e = (u_1, \dots, u_d)$ and $e' = (u'_1, \dots, u'_d)$ be two hyperedges sharing at least one endpoint where for an index set $\mathcal{I} \subset \{1, \dots, d\}$ of cardinality $d - 1$, the vertices u_i, u'_i for all $i \in \mathcal{I}$ only touch e and/or e' . That is for each $i \in \mathcal{I}$, either $u_i = u'_i$ is a degree-2 vertex or $u_i \neq u'_i$ and they are both degree-1 vertices. For $j \notin \mathcal{I}$, u_j and u'_j are arbitrary vertices. Then, in the current hypergraph, there exists a maximum cardinality matching having either e or e' .*

Proof. Let H' be the current hypergraph at hand and $j \notin \mathcal{I}$ be the remaining part id. Let M' be a maximum cardinality matching in H' . If either $e \in M'$ or $e' \in M'$, we are done. Otherwise, u_i and u'_i for all $i \in \mathcal{I}$ are unmatched by M' . Furthermore, since M' is maximal, u_j must be matched by M' (otherwise, M' can be extended by e). Let $e'' \in M'$ be the hyperedge matching u_j . Then $M' \setminus \{e''\} \cup \{e\}$ defines a valid maximum cardinality matching for H' . \square

Whenever such hyperedges appear, the rules below are applied in the same order:

- **Rule-1:** At any time during the heuristic, if a hyperedge e with at least $d - 1$ degree-1 endpoints appears, instead of a random hyperedge, e is added to the matching and removed from the hypergraph.

- **Rule-2:** Otherwise, if two hyperedges e and e' as defined in Lemma 2 appear, they are removed from the current hypergraph with the endpoints u_i, u'_i for all $i \in \mathcal{I}$. Then, we consider u_j and u'_j . If u_j and u'_j are distinct, they are merged to create a new vertex $u_j u'_j$, whose hyperedge list is defined as the union of u_j 's and u'_j 's hyperedge lists. If u_j and u'_j are identical, we rename u_j as $u_j u'_j$. After obtaining a maximal matching on the reduced hypergraph, depending on the hyperedge matching $u_j u'_j$, either e or e' can be used to obtain a larger matching in the current hypergraph.

When Rule-2 is applied, the two hyperedges identified in Lemma 2 are removed from the hypergraph, and only the hyperedges containing u_j and/or u'_j have an update in their vertex list. Since the original hypergraph is d -partite and d -uniform, that update is just a renaming of a vertex in the concerned hyperedges (hence the resulting hypergraph is d -partite and d -uniform).

Although the extended rules usually lead to improved results in comparison to Greedy, Karp-Sipser still adheres to the d -approximation bound of maximal matchings. To see this, we use the toy example given as a worst-case for Greedy. For the example given at the beginning of Section 3, Karp-Sipser generates a maximum cardinality matching by applying Rule-1. However, if $e_5 = (2, 1, 3)$ and $e_6 = (3, 1, 3)$ are added to the example, neither of the two rules can be applied. As before, if e_4 is randomly selected, it forms a maximal matching.

3.3 Karp-Sipser-scaling for Max- d -DM

Karp-Sipser can be modified for better decisions in case neither of the two rules applies. In this variant, called Karp-Sipser-scaling, instead of a random selection, we first scale the adjacency tensor of H and obtain an approximate d -stochastic tensor \mathbf{T} . We then augment the matching by adding the hyperedge which corresponds to the largest value in \mathbf{T} . The modified heuristic is summarized in Algorithm 1.

Our inspiration comes from the $d = 2$ case and more specifically from the relation between scaling and matching. It is known due to Birkhoff [4] that the polytope of $n \times n$ doubly stochastic matrices is the convex hull of the $n \times n$ permutation matrices. A nonnegative matrix \mathbf{A} where all entries participate in some perfect matching can be scaled with two positive diagonal matrices \mathbf{R} and \mathbf{C} such that \mathbf{RAC} is doubly stochastic. Otherwise, provided that \mathbf{A} has a perfect matching, it can still be scaled to a doubly stochastic form asymptotically. In this case, the entries not participating in any perfect matching tend to zero in the scaled matrix. This fact is exploited to design randomized approximation algorithms for the maximum cardinality matching problem in graphs [12, 13]. By scaling the adjacency matrix in a preprocess and choosing edges with a probability corresponding to the scaled value of the associated matrix entry, the edges which are not included in a perfect matching become less likely to be chosen. The current algorithm differs from these approaches by selecting a single hyperedge at each step and applying scaling again before the next selection.

For $d \geq 3$, there is no equivalent of Birkhoff's theorem as demonstrated by the following lemma.

Algorithm 1: Karp-Sipser-scaling

Input: A d -partite, d -uniform $n_1 \times \dots \times n_d$ hypergraph $H = (V, E)$
Output: A maximal matching M of H

- 1: $M \leftarrow \emptyset$ ► Initially M is empty
- 2: $S \leftarrow \emptyset$ ► Stack for the merges for Rule-2
- 3: **while** H is not empty **do**
- 4: Remove the isolated vertices from H
- 5: **if** $\exists e = (u_1, \dots, u_d)$ as in Rule-1 **then**
- 6: $M \leftarrow M \cup \{e\}$ ► Add e to the matching
- 7: Apply the reduction for Rule-1 on H
- 8: **else if** $\exists e = (u_1, \dots, u_d), e' = (u'_1, \dots, u'_d)$ and \mathcal{I} as in Rule-2 **then**
- 9: Let j be the part index where $j \notin \mathcal{I}$
- 10: Apply the reduction for Rule-2 on H by introducing the vertex $u_j u'_j$
- 11: $E' = \{(v_1, \dots, u_j u'_j, \dots, v_d) : \text{for all } (v_1, \dots, u_j, \dots, v_d) \in E\}$
 ► memorize the hyperedges of u_j
- 12: $S.\text{push}(e, e', u_j u'_j, E')$ ► Store the current merge
- 13: **else**
- 14: $\mathbf{T} \leftarrow \text{SCALE}(\text{adj}(H))$ ► Scale the adjacency tensor of H
- 15: $e \leftarrow \arg \max_{(u_1, \dots, u_d)} (\mathbf{T}_{u_1, \dots, u_d})$ ► Find the maximum entry in \mathbf{T}
- 16: $M \leftarrow M \cup \{e\}$ ► Add e to the matching
- 17: Remove all hyperedges of u_1, \dots, u_d from E
- 18: $V \leftarrow V \setminus \{u_1, \dots, u_d\}$
- 19: **while** $S \neq \emptyset$ **do**
- 20: $(e, e', u_j u'_j, E') \leftarrow S.\text{pop}()$ ► Get the most recent merge
- 21: **if** $u_j u'_j$ is not matched by M **then**
- 22: $M \leftarrow M \cup \{e\}$
- 23: **else**
- 24: Let $e'' \in M$ be the hyperedge matching $u_j u'_j$
- 25: **if** $e'' \in E'$ **then**
- 26: Replace $u_j u'_j$ in e'' with u'_j
- 27: $M \leftarrow M \cup \{e'\}$
- 28: **else**
- 29: Replace $u_j u'_j$ in e'' with u_j
- 30: $M \leftarrow M \cup \{e\}$

Lemma 3. *For $d \geq 3$, there exist extreme points in the set of d -stochastic tensors which are not permutation tensors.*

The proof can be found in the accompanying technical report [14], where we give extreme points that are not permutation tensors. Due to the lemma above, we do not have the theoretical foundation to imply that hyperedges corresponding to the large entries in the scaled tensor must necessarily participate in a perfect matching. Nonetheless, the entries not in any perfect matching tend to become zero (not guaranteed for all though). For the worst case example of Karp-Sipser described above, the scaling indeed helps the entries corresponding to e_4, e_5 and e_6 to become zero. Additionally even if the heuristic selects an entry in the non-zero pattern of an extreme point without a perfect matching, we do not necessarily reduce our chances of obtaining a good matching (see the discussion following the proof of Lemma 3 in the technical report).

On a d -partite, d -uniform hypergraph $H = (V, E)$, the Sinkhorn-Knopp algorithm used for scaling operates in iterations, each of which requires $\mathcal{O}(|E| \times d)$

time. In practice, only a few iterations (e.g., 10–20) can be performed. Since we can match at most $|V|/d$ hyperedges, the overall run time of scaling is $\mathcal{O}(|V| \times |E|)$. A straightforward implementation of the second rule can take quadratic time in the case of a large number of repetitive merges with a given vertex. In practice, more of a linear time behavior should be observed.

3.4 Hypergraph matching via pseudo scaling

In Algorithm 1, applying scaling at every step can be very costly. Here we propose an alternative idea inspired by the specifics of the Sinkhorn-Knopp algorithm to reduce the overall cost.

The Sinkhorn-Knopp algorithm scales a d -dimensional tensor \mathbf{T} in a series of iterations by updating the set of vectors $x^{(1)}, \dots, x^{(d)}$ where initially all values in all vectors are equal to 1. During an iteration, the coefficient vector $x^{(j)}$ for a given dimension j is updated by using

$$x_{i_j}^{(j)} = \frac{x_{i_j}^{(j)}}{\sum_{i_1, \dots, i_{j-1}, i_{j+1}, \dots, i_d} \left(\mathbf{T}_{i_1, \dots, i_j, \dots, i_d} \prod_{k=1}^d x_{i_k}^{(k)} \right)}, \text{ for all } i_j \in \{1, \dots, n_j\}. \quad (1)$$

These updates are done in a sequential order and for simplicity we assume that they happen in the dimension order: $1, \dots, d$. Each vector entry $x_{i_j}^{(j)}$ corresponds to a vertex in the hypergraph. Let λ_{i_j} denote the degree of the vertex i_j from j th part. For the first iteration of (1), each $x_{i_1}^{(1)}$ is set to $\frac{1}{\lambda_{i_1}}$ since all values in the vectors are one. The pseudo scaling approach applies d parallel executions of updates (1) and sets each $x_{i_j}^{(j)} = \frac{1}{\lambda_{i_j}}$ for all $j \in \{1, \dots, d\}$ and $i_j \in \{1, \dots, n_j\}$. That is, each vertex gets a value inversely proportional to its degree. This avoids 10–20 iterations of Sinkhorn-Knopp and the $\mathcal{O}(|E|)$ cost for each. However, as the name of the approach implies, this scaling is not exact.

With this approach each hyperedge $\{i_1, \dots, i_d\}$ is associated with a value $\frac{1}{\prod_{j=1}^d \lambda_{i_j}}$. The selection procedure is the same as that of Algorithm 1, i.e., the hyperedge with the maximum value is added to the matching set. We refer to this algorithm as **Karp-Sipser-mindegree**, as it selects a hyperedge based on a function of the degrees of the vertices. With a straightforward implementation, finding this hyperedge takes $\mathcal{O}(|E|)$ time.

3.5 Reduction to bipartite graph matching

A perfect matching in a d -partite, d -uniform hypergraph H remains perfect when projected on a $(d-1)$ -partite, $(d-1)$ -uniform hypergraph obtained by removing one of H 's vertex parts. Matchability in $(d-1)$ -partite sub-hypergraphs has been investigated [1] to provide an equivalent of Hall's Theorem for d -partite hypergraphs. These observations lead us to propose a heuristic called **Bipartite-reduction**. This heuristic tackles the d -partite, d -uniform case by recursively asking for matchings in $(d-1)$ -partite, $(d-1)$ -uniform hypergraphs and so on, until $d=2$.

Let us start with $d=3$. Let $G = (V_G, E_G)$ be the bipartite graph where the vertex set $V_G = V_1 \cup V_2$ is obtained by deleting V_3 from a 3-partite, 3-regular

hypergraph $H = (V, E)$. The edge $(u, v) \in E_G$ iff there exists a hyperedge $(u, v, z) \in E$. One can assign weights to the edges during this step, e.g., $w(u, v) = |\{z : (u, v, z) \in E\}|$. A maximum weighted matching algorithm can be used to obtain a matching M_G on G . A second bipartite graph $G' = (V_{G'}, E_{G'})$ is then created with $V_{G'} = (V_1 \times V_2) \cup V_3$ and $E_{G'} = \{(uv, z) : (u, v) \in M_G, (u, v, z) \in H\}$. Any matching in G' corresponds a valid matching in H . Furthermore, if the weight function defined above is used with a maximum weighted matching algorithm, the number of edges surviving for G' is maximized.

For d -dimensional matching, a similar process is followed. First, an ordering i_1, i_2, \dots, i_d of the dimensions is defined. At the j th bipartite reduction step, the matching is found between the dimension cluster $i_1 i_2 \dots i_j$ and the dimension i_{j+1} by similarly solving a bipartite matching problem, where the edge $(u_1 \dots u_j, v)$ exists in the bipartite graph iff the vertices u_1, \dots, u_j were matched previously, and there exists a hyperedge $(u_1, \dots, u_j, v, z_{j+2}, \dots, z_d)$ in H . Unlike the previous heuristics, **Bipartite-reduction** does not have any approximation guarantee, as stated in the following lemma (the proof is in the accompanying technical report [14], where we describe a family of hypergraphs for which **Bipartite-reduction** yields $\frac{5}{n}$ -approximation for $n \geq 5$).

Lemma 4. *If algorithms for the maximum cardinality or the maximum weighted matching (with the suggested edge weights) problems are used, then **Bipartite-reduction** has a worst-case approximation ratio of $\Omega(n)$.*

3.6 Performing local search

A local search heuristic is proposed by Hurkens and Schrijver [24]. It starts from a feasible maximal matching M and performs a series of swaps until it is no longer possible. In a swap, k hyperedges of M are replaced with at least $k + 1$ new hyperedges from $E \setminus M$ so that the cardinality of M increases by at least one. These k hyperedges from M can be replaced with at most $d \times k$ new edges. Hence, these hyperedges can be found by a polynomial algorithm enumerating all the possibilities. The approximation guarantee improves with higher k values. Local search algorithms are limited in practice due to their high time complexity. The algorithm might have to examine all $\binom{|M|}{k}$ subsets of M to find a feasible swap at each step. The algorithm by Cygan [8] which achieves $\left(\frac{d+1+\epsilon}{3}\right)$ -approximation is based on a different swap scheme but is also not suited for large hypergraphs.

4 Experiments

To understand the relative performance of the proposed heuristics, we conducted a wide variety of experiments with both synthetic and real-life data. The experiments were performed on a computer equipped with intel Core i7-7600 CPU and 16GB RAM. For $d = 3$, we also implemented a local search heuristic [24], called **Local-Search**, which replaces one hyperedge from a maximal matching M with at least two hyperedges from $E \setminus M$ to increase the cardinality of M . We did not consider local search schemes for higher dimensions or with better approximation ratios as they are computationally too expensive. For each hypergraph, we perform ten runs of **Greedy** and **Karp-Sipser** with different random decisions and

take the maximum cardinality obtained. Since Karp-Sipser-scaling or Karp-Sipser-mindegree do not pick hyperedges randomly, we run them only once. We perform 20 steps of the scaling procedure in Karp-Sipser-scaling. We refer to quality of a matching M in a hypergraph H as the ratio of M 's cardinality to the size of the smallest vertex partition of H .

4.1 Experiments on random hypergraphs

We perform experiments on two classes of d -partite, d -uniform random hypergraphs where each part has n vertices. The first class contains random k -out hypergraphs, and the second one contains sparse random hypergraphs.

Random k -out, d -partite, d -uniform hypergraphs

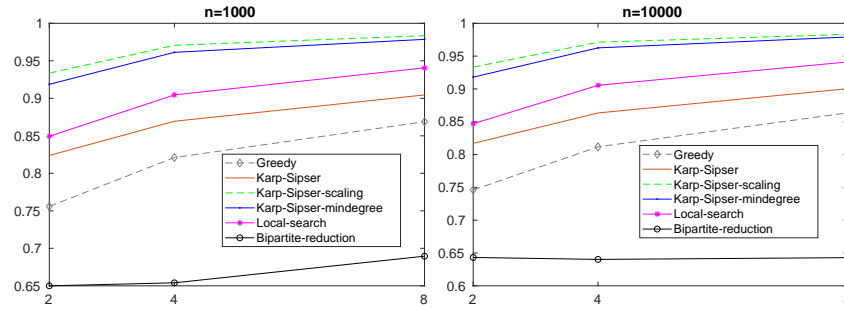
Here, we consider random k -out, d -partite, d -uniform hypergraphs described in Section 2. Hence (ignoring the duplicate ones), these hypergraphs have around $d \times k \times n$ hyperedges. These k -out, d -partite, d -uniform hypergraphs have been recently analyzed in the matching context by Devlin and Kahn [10]. They state in passing that k should be exponential in d for a perfect matching to exist with high probability. The bipartite graph variant of the same problem has been extensively studied in the literature [17, 25, 34]; a perfect matching almost always exists in a random 2-out bipartite graph [34].

		k					k				
		d	d^{d-3}	d^{d-2}	d^{d-1}			d	d^{d-3}	d^{d-2}	d^{d-1}
$n = 10$	2	-	0.87	1.00	$n = 30$	2	-	0.84	1.00		
	3	0.80	1.00	1.00		3	0.88	1.00	1.00		
	4	1.00	1.00	1.00		4	0.99	1.00	1.00		
	5	1.00	1.00	1.00		5	*	1.00	1.00		
$n = 20$	2	-	0.88	1.00	$n = 50$	2	-	0.87	1.00		
	3	0.85	1.00	1.00		3	0.84	1.00	1.00		
	4	1.00	1.00	1.00		4	*	1.00	1.00		
	5	1.00	1.00	1.00		5	*	*	*		

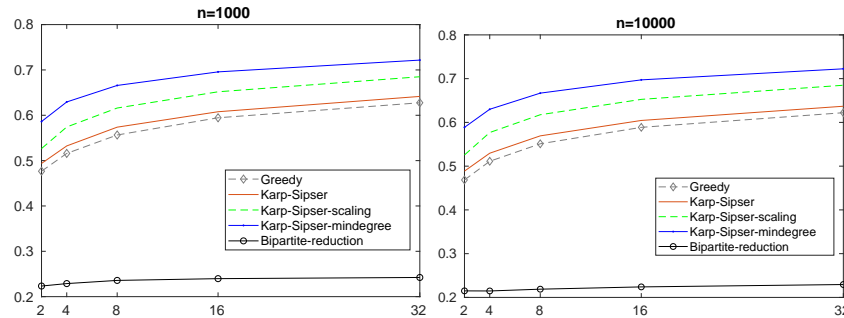
Table 1: The average maximum matching cardinalities of five random instances over n on random k -out, d -partite, d -uniform hypergraphs for different k , d , and n . No runs for $k = d^{d-3}$ and $d = 2$; the problems marked with * were not solved within 24 hours.

We first investigate the existence of perfect matchings in random k -out, d -partite, d -uniform hypergraphs. For this purpose, we implemented the linear program of d -dimensional matching in CPLEX and found the maximum cardinality of a matching in these hypergraphs with $k \in \{d^{d-3}, d^{d-2}, d^{d-1}\}$ for $d \in \{2, \dots, 5\}$ and $n \in \{10, 20, 30, 50\}$. For each (k, d, n) triple, we created five hypergraphs and computed their maximum cardinality matchings. For $k = d^{d-3}$, we encountered several hypergraphs with no perfect matching, especially for $d = 3$. The hypergraphs with $k = d^{d-2}$ were also lacking a perfect matching for $d = 2$. However, all the hypergraphs we created with $k = d^{d-1}$ had at least one. Based on these results, we experimentally confirm Devlin and Kahn's statement. We also conjecture that d^{d-1} -out random hypergraphs have perfect matchings almost surely. The average maximum matching cardinalities we obtained in this experiment

are given in Table 1. In this table, we do not have results for $k = d^{d-3}$ for $d = 2$, and the cases marked with * were not solved within 24 hours.



(a) $d = 3$, $n = 1000$ (left) and $n = 10000$ (right)



(b) $d = 9$, $n = 1000$ (left) and $n = 10000$ (right)

Fig. 1: The performance of the heuristics on k -out, d -partite, d -uniform hypergraphs with n vertices at each part. The y -axis is the ratio of matching cardinality to n whereas the x -axis is k . No Local-Search for $d = 9$.

We now compare the performance of the proposed heuristics on random k -out, d -partite, d -uniform hypergraphs with $d \in \{3, 9\}$ and $n \in \{1000, 10000\}$. We tested with k values equal to powers of two for $k \leq d \log d$. The results are summarized in Figure 1. For each (k, d, n) triplet, we create ten random instances and present the average performance of the heuristics on them. Further figures for $d = 6$ can be found in the accompanying technical report [14]. The x -axis in each figure denotes k , and the y -axis reports the matching cardinality over n . As seen, Karp-Sipser-scaling and Karp-Sipser-mindegree have the best performance, comfortably beating the other alternatives. For $d = 3$ Karp-Sipser-scaling dominates Karp-Sipser-mindegree, but when $d > 3$ we see that Karp-Sipser-mindegree has the best performance. Karp-Sipser performs better than Greedy. However, their performances get closer as d increases. This is due to the fact that the conditions for Rule-1 and Rule-2 hold less often for larger d . Bipartite-reduction has worse performance than the others, and the gap in the performance grows as d

increases. This happens, since at each step, we impose more and more conditions on the edges involved and there is no chance to recover from bad decisions.

Sparse random d -partite, d -uniform hypergraphs

Here, we consider a random d -partite, d -uniform hypergraph H_i that has $i \times n$ random hyperedges. The parameters used for these experiments are $i \in \{1, 3, 5, 7\}$, $n \in \{4000, 8000\}$, and $d \in \{3, 9\}$. Each H_i is created by choosing the vertices of a hyperedge uniformly at random for each dimension. We do not allow duplicate hyperedges. Another random hypergraph H_{i+M} is then obtained by planting a perfect matching to H_i . We again generate ten random instances for each parameter setting. We do not present results for **Bipartite-reduction** as it was always worse than the others, as before. The average quality of different heuristics on these instances is shown in Figure 2 (the accompanying report [14] contains further results). The experiments confirm that **Karp-Sipser** performs consistently better than **Greedy**. Furthermore, **Karp-Sipser-scaling** performs significantly better than **Karp-Sipser**. **Karp-Sipser-scaling** works even better than the local search heuristic, and it is the only heuristic that is capable of finding planted perfect matchings for a significant number of the runs. In particular when $d > 3$, it finds a perfect matching on H_{i+M} s in all cases shown. For $d = 3$, it finds a perfect matching only when $i = 1$ and attains a near perfect matching when $i = 3$. Interestingly **Karp-Sipser-minDegree** outperforms **Karp-Sipser-scaling** on H_i s but is dominated on H_{i+M} s, where it is the second best performing heuristic.

i	H_i : Random Hypergraph								H_{i+M} : Random Hypergraph with Perfect Matching											
	Greedy		Local Search		Karp-Sipser		Karp-Sipser-scaling		Karp-Sipser-minDegree		Greedy		Local Search		Karp-Sipser		Karp-Sipser-scaling		Karp-Sipser-minDegree	
	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000
1	0.43	0.42	0.47	0.47	0.49	0.48	0.49	0.48	0.49	0.48	0.75	0.75	0.93	0.93	1.00	1.00	1.00	1.00	1.00	1.00
3	0.63	0.63	0.71	0.71	0.73	0.72	0.76	0.76	0.78	0.77	0.72	0.71	0.82	0.81	0.81	0.81	0.99	0.99	0.92	0.92
5	0.70	0.70	0.80	0.80	0.78	0.78	0.86	0.86	0.88	0.88	0.75	0.74	0.84	0.84	0.82	0.82	0.94	0.94	0.92	0.92
7	0.75	0.75	0.84	0.84	0.81	0.81	0.94	0.94	0.93	0.93	0.77	0.77	0.87	0.87	0.83	0.83	0.96	0.96	0.94	0.94

(a) $d = 3$, without (left) and with (right) the planted matching

i	H_i : Random Hypergraph								H_{i+M} : Random Hypergraph with Perfect Matching							
	Greedy		Karp-Sipser		Karp-Sipser-scaling		Karp-Sipser-minDegree		Greedy		Karp-Sipser		Karp-Sipser-scaling		Karp-Sipser-minDegree	
	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000	4000	8000
1	0.25	0.24	0.27	0.27	0.27	0.27	0.30	0.30	0.56	0.55	0.80	0.79	1.00	1.00	1.00	1.00
3	0.34	0.33	0.36	0.36	0.36	0.36	0.43	0.43	0.40	0.40	0.44	0.44	1.00	1.00	0.99	1.00
5	0.38	0.37	0.40	0.40	0.41	0.41	0.48	0.48	0.41	0.40	0.43	0.43	1.00	1.00	0.99	0.99
7	0.40	0.40	0.42	0.42	0.44	0.44	0.51	0.51	0.42	0.42	0.44	0.44	1.00	1.00	0.97	0.96

(b) $d = 9$, without (left) and with (right) the planted matching

Fig. 2: Performance comparisons on d -partite, d -uniform hypergraphs with $n = \{4000, 8000\}$. H_i contains $i \times n$ random hyperedges, and H_{i+M} contains an additional perfect matching.

4.2 Evaluating algorithmic choices

Here, we evaluate the use of scaling and the importance of Rule-1 and Rule-2.

Scaling vs no-scaling

To evaluate and emphasize the contribution of scaling better, we compare the

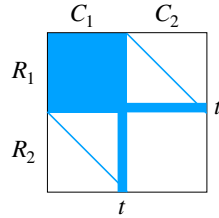


Fig. 3: \mathbf{A}_{KS} : A challenging instance for Karp-Sipser^g .

t	Greedy	Local Search	Karp-Sipser	Karp-Sipser-scaling	Karp-Sipser-minDegree
2	0.53	0.99	0.53	1.00	1.00
4	0.53	0.99	0.53	1.00	1.00
8	0.54	0.99	0.55	1.00	1.00
16	0.55	0.99	0.56	1.00	1.00
32	0.59	0.99	0.59	1.00	1.00

Table 2: Performance of the proposed heuristics on 3-partite, 3-uniform hypergraphs corresponding to \mathbf{T}_{KS} with $n = 300$ vertices in each part.

performance of the heuristics on a particular family of d -partite, d -uniform hypergraphs where their bipartite counterparts have been used before as challenging instances for the original Karp-Sipser^g heuristic [12].

Let \mathbf{A}_{KS} be an $n \times n$ matrix. Let R_1 and C_1 be \mathbf{A}_{KS} 's first $n/2$ rows and columns, respectively, and R_2 and C_2 be the remaining $n/2$ rows and columns, respectively. Let the block $R_1 \times C_1$ be full and the block $R_2 \times C_2$ be empty. A perfect bipartite graph matching is hidden inside the blocks $R_1 \times C_2$ and $R_2 \times C_1$ by introducing a non-zero diagonal to each. In addition, a parameter t connects the last t rows of R_1 with all the columns in C_2 . Similarly, the last t columns in C_1 are connected to all the rows in R_2 . An instance from this family of matrices is depicted in Figure 3. Karp-Sipser^g is impacted negatively when $t \geq 2$ whereas Greedy^g struggles even with $t = 0$ because random edge selections will almost always be from the dense $R_1 \times C_1$ block. To adapt this scheme to hypergraphs/tensors, we generate a 3-dimensional tensor \mathbf{T}_{KS} such that the nonzero pattern of each marginal of the 3rd dimension is identical to that of \mathbf{A}_{KS} . Table 2 shows the performance of the heuristics (i.e., matching cardinality normalized with n) for 3-dimensional tensors with $n = 300$ and $t \in \{2, 4, 8, 16, 32\}$.

The use of scaling indeed reduces the influence of the misleading hyperedges in the dense block $R_1 \times C_1$, and the proposed $\text{Karp-Sipser-scaling}$ heuristic always finds the perfect matching as does $\text{Karp-Sipser-mindegree}$. However, Greedy and Karp-Sipser perform significantly worse. Furthermore, Local-Search returns 0.99-approximation in every case because it ends up in a local optima.

Rule-1 vs Rule-2

We finish the discussion on the synthetic data by focusing on Karp-Sipser . Recall from Section 3.2 that Karp-Sipser has two rules. In the bipartite case, a variant of Karp-Sipser^g in which Rule-2 is not applied received more attention than the original version, because it is simpler to implement and easier to analyze. This simpler variant has been shown to obtain good results both theoretically [26] and experimentally [12]. Recent work [2] shows that both rules are needed to obtain perfect matchings in random cubic graphs.

We present a family of hypergraphs to demonstrate that using Rule-2 can lead to better performance than using Rule-1 only. We use Karp-Sipser_{R_1} to refer to Karp-Sipser without Rule-2. As before, we describe first the bipartite case. Let \mathbf{A}_{RF} be a $n \times n$ matrix with $(\mathbf{A}_{RF})_{i,j} = 1$ for $1 \leq i \leq j \leq n$, and $(\mathbf{A}_{RF})_{2,1} = (\mathbf{A}_{RF})_{n,n-1} = 1$. That is, \mathbf{A}_{RF} is composed of an upper triangular

matrix and two additional subdiagonal nonzeros. The first two columns and the last two rows have two nonzeros. Assume without loss of generality that the first two rows are merged by applying Rule-2 on the first column (which is discarded). Then in the reduced matrix, the first column (corresponding to the second column in the original matrix) will have one nonzero. Rule-1 can now be applied whereupon the first column in the reduced matrix will have degree one. The process continues similarly until the reduced matrix is a 2×2 dense block, where applying Rule-2 followed by Rule-1 yields a perfect matching. If only Rule-1 reductions are allowed, initially no reduction can be applied and randomly chosen edges will be matched, which negatively affects the quality of the returned matching.

For higher dimensions we proceed as follows. Let \mathbf{T}_{RF} be a d -dimensional $n \times \dots \times n$ tensor. We set $(\mathbf{T}_{RF})_{i,j,\dots,j} = 1$ for $1 \leq i \leq j \leq n$ and $(\mathbf{T}_{RF})_{1,2,\dots,2} = (\mathbf{T}_{RF})_{n,n-1,\dots,n-1} = 1$. By similar reasoning, we see that Karp-Sipser with both reduction rules will obtain a perfect matching, whereas Karp-Sipser $_{R_1}$ will struggle. We give some results in Table 3 that show the difference between the two. We test for $n \in \{1000, 2000, 4000\}$ and $d \in \{2, 3, 6\}$, and show the quality of Karp-Sipser $_{R_1}$ and the number of times that Rule-1 is applied over n . We present the best result over 10 runs.

As seen in Table 3, Karp-Sipser $_{R_1}$ obtains matchings that are about 13–25% worse than Karp-Sipser. Furthermore, the larger the number of Rule-1 applications is, the higher the quality is.

n	d					
	2		3		6	
	quality	$\frac{r}{n}$	quality	$\frac{r}{n}$	quality	$\frac{r}{n}$
1000	0.83	0.45	0.85	0.47	0.80	0.31
2000	0.86	0.53	0.87	0.56	0.80	0.30
4000	0.82	0.42	0.75	0.17	0.84	0.45

Table 3: Quality of matching and the number r of the applications of Rule-1 over n in Karp-Sipser $_{R_1}$, for hypergraphs corresponding to \mathbf{T}_{RF} . Karp-Sipser obtains perfect matchings.

4.3 Experiments with real-life tensor data

We also evaluate the performance of the proposed heuristics on some real-life tensors selected from FROSTT library [33]. The descriptions of the tensors are given in Table 4. For **nips** and **Uber**, a dimension of size 17 and 24 is dropped respectively, as they restrict the size of the maximum cardinality matching. As described before, a d -partite, d -uniform hypergraph is obtained from a d -dimensional tensor by associating a vertex for each dimension index, and a hyperedge for each nonzero. Unlike the previous experiments, the parts of the hypergraphs obtained from real-life tensors in Table 4 do not have an equal number of vertices. In this case, the scaling algorithm works along the same lines. Let $n_i = |V_i|$ be the cardinality at the i th dimension, and $n_{max} = \max_{1 \leq i \leq d} n_i$ be the maximum one. By slightly modifying Sinkhorn-Knopp, for each iteration of

Karp-Sipser-scaling, we scale the tensor such that the marginals in dimension i sum up to n_{max}/n_i instead of one. The results in Table 4 resemble those from previous sections; **Karp-Sipser-scaling** has the best performance and is slightly superior to **Karp-Sipser-mindegree**. **Greedy** and **Karp-Sipser** are close to each other and when it is feasible, **Local-Search** is better than them. In addition we see that in these instances **Bipartite-reduction** exhibits a good performance: its performance is at least as good as **Karp-Sipser-scaling** for the first three instances, but about 10% worse for the last one.

Tensor	d	Dimensions	nnz	Greedy	Local-Search	Karp-Sipser	Karp-Sipser-minDegree	Karp-Sipser-scaling	Bipartite-Reduction
Uber	3	$183 \times 1140 \times 1717$	1,117,629	183	183	183	183	183	183
nips [19]	3	$2,482 \times 2,862 \times 14,036$	3,101,609	1,847	1,991	1,839	2005	2,007	2,007
Nell-2 [5]	3	$12,092 \times 9,184 \times 28,818$	76,879,419	3,913	4,987	3,935	5,100	5,154	5,175
Enron [31]	4	$6,066 \times 5,699 \times 244,268 \times 1,176$	54,202,099	875	-	875	988	1,001	898

Table 4: The performance of the proposed heuristics on the hypergraphs corresponding to real-life tensors. No **Local-Search** for four dimensional tensor **Enron**.

4.4 Experiments with an independent set solver

We compare **Karp-Sipser-scaling** and **Karp-Sipser-mindegree** with the idea of reducing **MAX- d -DM** to the problem of finding an independent set in the line graph of the given hypergraph. We show that this transformation can lead good results, but is restricted because line graphs can require too much space.

We use **KaMIS** [29] to find independent sets in graphs. **KaMIS** uses a plethora of reductions and a genetic algorithm in order to return high cardinality independent sets. We use the default settings of **KaMIS** (where execution time is limited to 600 seconds) and generate the line graphs with efficient sparse matrix-matrix multiplication routines. We run **KaMIS**, **Greedy**, **Karp-Sipser-scaling**, and **Karp-Sipser-mindegree** on a few hypergraphs from previous tests. The results are summarized in Table 5. The run time of **Greedy** was less than one second in all instances. **KaMIS** operates in rounds, and we give the quality and the run time of the first round and the final output. We note that **KaMIS** considers the time-limit only after the first round has been completed. As can be seen, while the quality of **KaMIS** is always good and in most cases superior to **Karp-Sipser-scaling** and **Karp-Sipser-mindegree**, it is also significantly slower (its principle is to deliver high quality results). We also observe that the pseudo scaling of **Karp-Sipser-mindegree** indeed helps to reduce the run time compared to **Karp-Sipser-scaling**.

The line graphs of the real-life instances from Table 4 are too large to be handled. We estimated (using known techniques [7]) the number of edges in these graphs to range from 1.5×10^{10} to 4.7×10^{13} . The memory needed ranges from 126GB to 380TB if edges are stored twice (assuming 4 bytes per edge).

5 Conclusion and future work

We have proposed heuristics for the **MAX- d -DM** problem by generalizing existing heuristics for the maximum cardinality matching in bipartite graphs. The experimental analyses on various hypergraphs/tensors show the effectiveness and

hypergraph	KaMIS						Greedy quality	Karp-Sipser- scaling		Karp-Sipser- mindegree	
	line graph gen. time	Round 1		Output		quality		quality	time	quality	time
		quality	time	quality	time						
8-out, $n = 1000$, $d = 3$	10	0.98	80	0.99	600	0.86	0.98	1	0.98	1	
8-out, $n = 10000$, $d = 3$	112	0.98	507	0.99	600	0.86	0.98	197	0.98	1	
8-out, $n = 1000$, $d = 9$	298	0.67	798	0.69	802	0.55	0.62	2	0.67	1	
$n = 8000$, $d = 3$, H_3	1	0.77	16	0.81	602	0.63	0.76	5	0.77	1	
$n = 8000$, $d = 3$, H_{3+M}	2	0.89	25	1.00	430	0.70	1.00	11	0.91	1	

Table 5: Run time (in seconds) and performance comparisons between KaMIS, Greedy, and Karp-Sipser-scaling. The time required to create the line graphs should be added to KaMIS’s overall time.

efficiency of the proposed heuristics. As future work, we plan to investigate the stated conjecture that d^{d-1} -out random hypergraphs have perfect matchings almost always, and analyze the theoretical guarantees of the proposed algorithms.

References

- Aharoni, R., Haxell, P.: Hall’s theorem for hypergraphs. *Journal of Graph Theory* **35**(2), 83–88 (2000)
- Anastos, M., Frieze, A.: Finding perfect matchings in random cubic graphs in linear time. arXiv preprint arXiv:1808.00825 (2018)
- Berman, P., Karpinski, M.: Improved approximation lower bounds on small occurrence optimization. *ECCC Report* (2003)
- Birkhoff, G.: Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucuman, Ser. A* **5**, 147–154 (1946)
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: *AAAI*. vol. 5, p. 3 (2010)
- Çatalyürek, Ü.V., Aykanat, C.: PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0. Bilkent University, Department of Computer Engineering, Ankara, 06533 Turkey. Available at <https://www.cc.gatech.edu/~umit/software.html> (1999)
- Cohen, E.: Structure prediction and computation of sparse matrix products. *Journal of Combinatorial Optimization* **2**(4), 307–332 (Dec 1998)
- Cygan, M.: Improved approximation for 3-dimensional matching via bounded path-width local search. In: *Foundations of Computer Science (FOCS)*, 2013 IEEE 54th Annual Symposium on. pp. 509–518. IEEE (2013)
- Cygan, M., Grandoni, F., Mastrolilli, M.: How to sell hyperedges: The hypermatching assignment problem. In: *Proc. of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. pp. 342–351. SIAM (2013)
- Devlin, P., Kahn, J.: Perfect fractional matchings in k -out hypergraphs. arXiv preprint arXiv:1703.03513 (2017)
- Dufossé, F., Kaya, K., Panagiotas, I., Uçar, B.: Scaling matrices and counting perfect matchings in graphs. *Tech. Rep. RR-9161*, Inria - Research Centre Grenoble – Rhône-Alpes (2018)
- Dufossé, F., Kaya, K., Uçar, B.: Two approximation algorithms for bipartite matching on multicore architectures. *J. Parallel Distr. Com.* **85**, 62–78 (2015)
- Dufossé, F., Kaya, K., Panagiotas, I., Uçar, B.: Approximation algorithms for maximum matchings in undirected graphs. In: *Proc. Seventh SIAM Workshop on Combinatorial Scientific Computing*. pp. 56–65. SIAM, Bergen, Norway (2018)

14. Dufossé, F., Kaya, K., Panagiotas, I., Uçar, B.: Effective heuristics for matchings in hypergraphs. Research Report RR-9224, Inria Grenoble Rhône-Alpes (Nov 2018), <https://hal.archives-ouvertes.fr/hal-01924180>
15. Dyer, M., Frieze, A.: Randomized greedy matching. *Random Structures & Algorithms* **2**(1), 29–45 (1991)
16. Franklin, J., Lorenz, J.: On the scaling of multidimensional matrices. *Linear Algebra and its applications* **114**, 717–735 (1989)
17. Frieze, A.M.: Maximum matchings in a class of random graphs. *J. Comb. Theory B* **40**(2), 196 – 212 (1986)
18. Froger, A., Guyon, O., Pinson, E.: A set packing approach for scheduling passenger train drivers: the French experience. In: RailTokyo2015. Tokyo, Japan (Mar 2015), <https://hal.archives-ouvertes.fr/hal-01138067>
19. Globerson, A., Chechik, G., Pereira, F., Tishby, N.: Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research* **8**, 2265–2295 (2007)
20. Gottlob, G., Greco, G.: Decomposing combinatorial auctions and set packing problems. *J. ACM* **60**(4), 24:1–24:39 (Sep 2013)
21. Halldórsson, M.M.: Approximating discrete collections via local improvements. In: SODA. vol. 95, pp. 160–169 (1995)
22. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k -dimensional matching. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 83–97. Springer (2003)
23. Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k -set packing. *Computational Complexity* **15**(1), 20–39 (2006)
24. Hurkens, C.A.J., Schrijver, A.: On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics* **2**(1), 68–72 (1989)
25. Karoński, M., Pittel, B.: Existence of a perfect matching in a random $(1+e^{-1})$ -out bipartite graph. *J. Comb. Theory B* **88**(1), 1–16 (2003)
26. Karp, R.M., Sipser, M.: Maximum matching in sparse random graphs. In: FOCS’81. pp. 364–375. Nashville, TN, USA (1981)
27. Karp, R.M.: Reducibility among combinatorial problems. In: *Complexity of computer computations*, pp. 85–103. Springer (1972)
28. Kaya, O., Uçar, B.: Scalable sparse tensor decompositions in distributed memory systems. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. pp. 77:1–77:11. SC ’15, ACM, Austin, Texas (2015)
29. Lamm, S., Sanders, P., Schulz, C., Strash, D., Werneck, R.F.: Finding Near-Optimal Independent Sets at Scale. In: *Proceedings of the 16th Meeting on Algorithm Engineering and Experimentation (ALENEX’16)* (2016)
30. Pothén, A., Fan, C.J.: Computing the block triangular form of a sparse matrix. *ACM T. Math. Software* **16**, 303–324 (1990)
31. Shetty, J., Adibi, J.: The enron email dataset database schema and brief statistical report. Information sciences institute technical report, University of Southern California **4** (2004)
32. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.* **21**, 343–348 (1967)
33. Smith, S., Choi, J.W., Li, J., Vuduc, R., Park, J., Liu, X., Karypis, G.: FROSTT: The formidable repository of open sparse tensors and tools (2017), <http://frostdt.io/>
34. Walkup, D.W.: Matchings in random regular bipartite digraphs. *Discrete Math.* **31**(1), 59–64 (1980)