A Comparison of Algorithms for the Winner Determination Problem

CS136 Final Project

George Lok Arvind Narayanan

Harvard College

george.j.lok@gmail.com arvindn55@gmail.com

$$\max_{x_1,\ldots,x_k} \sum_{k=1}^{K} b_k x_k$$
$$\text{s.t.} \sum_{k=1}^{K} a_{kj} x_k \leq 1, \forall j \in G$$
$$x_k \in \{0,1\}, \forall k \in \{1,\ldots,K\}$$

Figure 1. The WDP as an IP. where b_k represents the value of bid k, x_k is an indicator for allocating to bid k, and a_{kj} is an indicator for item j being part of bid k.

1. Introduction

In auctions, the winner determination problem (WDP) is the problem of selecting the set of winning bids to maximize total bid value. In a single-item auction, the winner determination problem is easy since all the auctioneer has to do is select the largest bid. However, combinatorial auctions allow bidders to set bids that contain multiple items, which sets constraints on what set of bids can be accepted by the auctioneer.

WDP is NP-hard, which can easily be shown by reducing the weighted-independent set problem to WDP. Furthermore, approximating WDP to a constant factor is also NP-hard, meaning there is no polynomial-time approximation algorithm with a constant error factor for WDP.

2. Algorithms

2.1 Integer Program

2.1.1 Formulation

The Winner Determination problem is naturally described as an integer program (IP) that determines the best membership of bids that maximizes revenue with the constraint that bids cannot share items (see figure 1).

Such an IP is difficult to solve and is an NP-hard problem. However, in practical scenarios, we can relax the requirement that the variables be integers in order to get an upper bound on the optimization problem.

Branch-and-bound methods can use this upper bound to avoid computing possibilities that are provably worse than other possibilities. In real usage, this allows some large instances of the WDP to be solved within a factor of the optimal allocation in a reasonable amount of time.

2.1.2 IBM ILOG CPLEX Optimization Studio

IBM ILOG CPLEX Optimization Studio is an optimization software package that can be used to solve integer programs. For our particular project, CPLEX serves as our IP solver. We used the CPLEX python API and set custom parameters to solve our integer programs.

2.2 Greedy Algorithm

As a point of comparison against IBM ILOG CPLEX, we implemented a greedy algorithm for the WDP as described by Sandholm [2].

Bids are ordered in terms of descending $b(S)/\sqrt{|S|}$, and are added in that order to the allocation (bids that share atoms with bids already in the allocation are excluded). This algorithm runs in linear time in terms of the number of bids placed, and so is a good benchmark for comparing runtime and correctness with the IBM ILOG CPLEX.

2.3 Two-Atom Bids

The special case of max two-atom bids can be implemented as a polynomial algorithm. Specifically, we implement a maximum weighted matching algorithm, using goods as nodes and bids as weighted edges in the graph representation of our problem.

3. Methodology

3.1 Bid Generation

To generate bids in the OR* language, we use the Combinatorial Auction Test Suite (CATS) [1]. CATS allows the user to specify the number of goods and bids to generated from a selection of distributions, including fixed bid-length distributions.

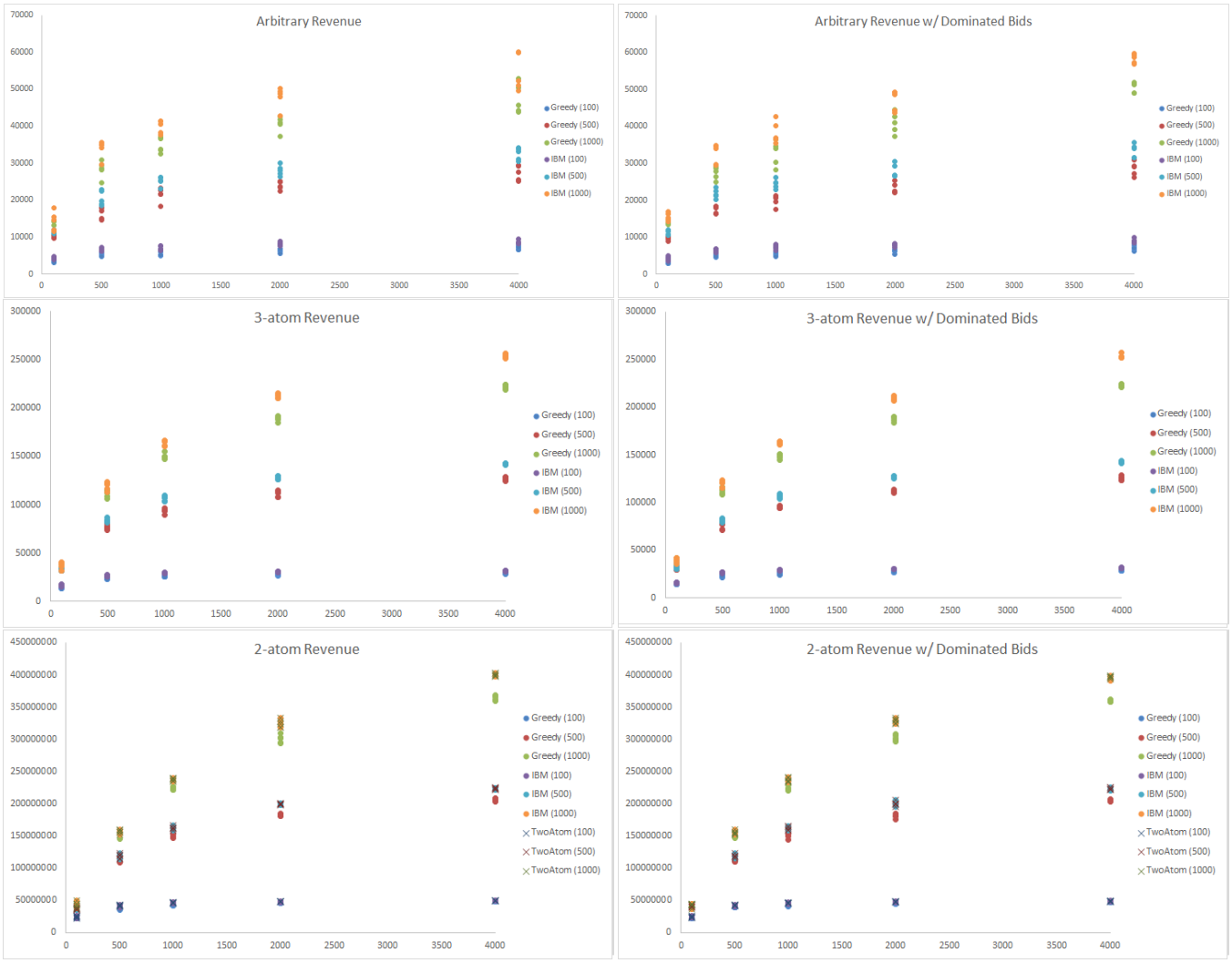

Figure 2. Comparisons of revenue among algorithms by distribution. Tests on the left side have no dominated bids. Each series represents a certain number of goods for a particular distribution. The x-axis represents the number of bids. The y-axis represents the amount of revenue.

We generate six types of bid distributions: arbitrary, two-atom, and three-atom, with and without dominated bids. For each type, we generated a total of 75 bid distributions, consisting of 5 instances of a combination of 100, 500, & 1000 goods and 100, 500, 1000, 2000, & 4000 bids.

We parsed the native CATS format files into a readable format for our algorithms. CATS also provides the ability to out CPLEX compatible LP files.

3.2 Environment

All tests were run in a virtual machine running Ubuntu 16.04 with 8 gb of memory and 2 cores with 4 threads running at 2.0 ghz.

3.3 Implementation

We implemented a python version of the greedy algorithm.

For solving the maximum weighted matching problem, we unfortunately realized that we attempted to implement a bipartite graph version of the algorithm. For our tests, we use the Joris VR python implementation of the Galil algorithm for maximum weighted matching [3].

For IBM CPLEX, we try to find solutions within 5% of the optimal solution. We set the probing strategy to be aggressive and emphasize finding feasible solutions. Finally, we force timeouts after 5 minutes in order to prevent long periods of computation.

Timing for all algorithms was done using python's time module.

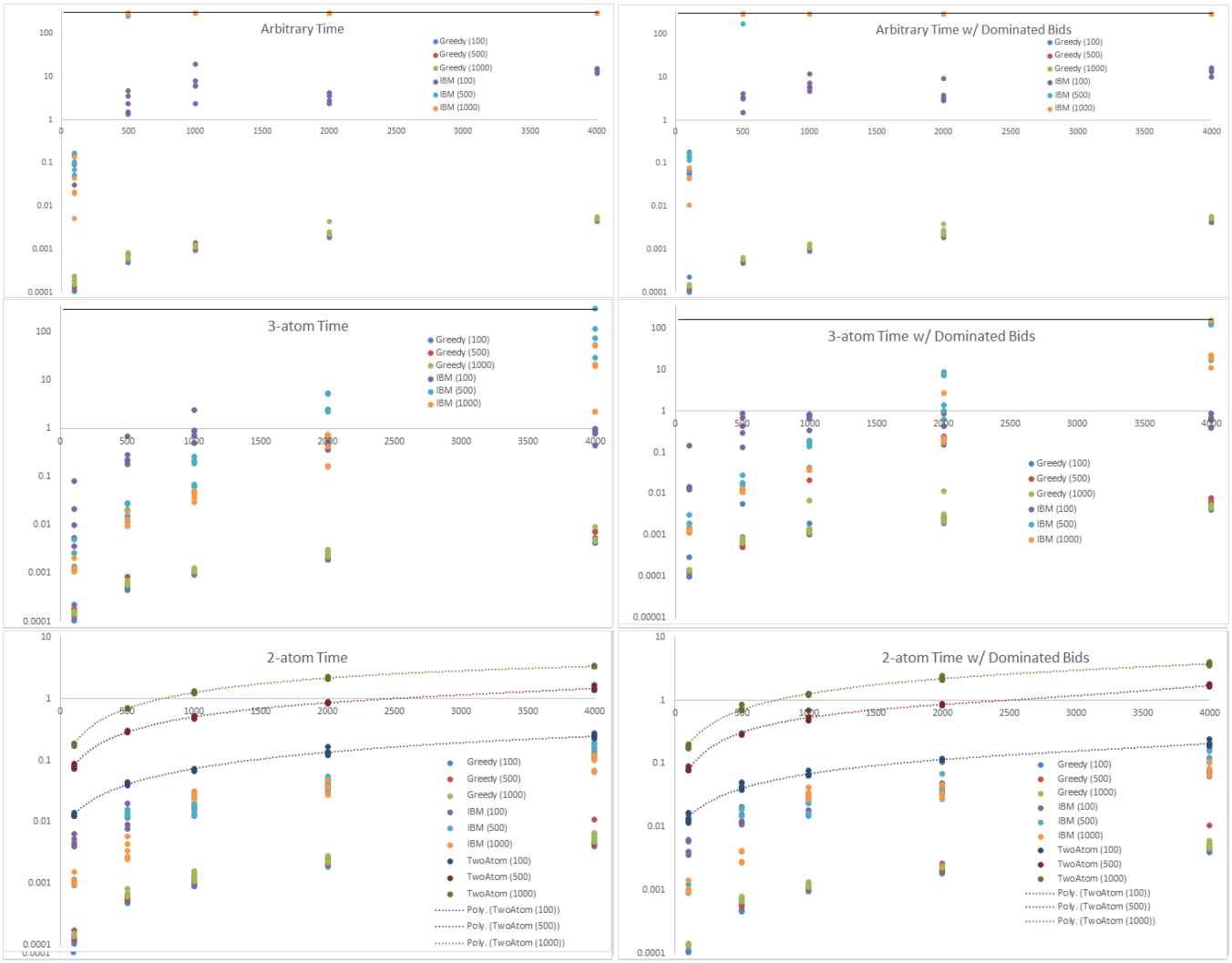

Figure 3. Comparisons of time among algorithms by distributions. Tests on the left side have no dominated bids. Each series represents a certain number of goods for a particular distribution. The x-axis represents the number of bids. The y-axis represents the number of seconds taken to compute the winner. TwoAtom results are plotted with trendlines for degree 3 polynomials.

4. Results

4.1 Arbitrary

4.1.1 Revenue

We used CPLEX as a baseline in order to evaluate Greedy.

Greedy performs ok, with the majority of time coming withing a 20% factor of the IBM revenue (see figure 4).

However, relative to its performances in the two-atom and three-atom distributions, Greedy does significantly less well.

Furthermore, CPLEX itself did not reach optimal performance within the 5 minute bound, which means that greedy potentially is significantly far from the optimal solution.

From these results, arbitrary distributions seem to be significantly harder to optimize compared to two-atom and three-atom distributions.

4.1.2 Time

Greedy does extremely well here, with computations finishing extremely quickly (see figure 3). In particular, it has roughly the same time performance as in other distributions.

CPLEX on the other hand, times out on all distributions except distributions with only 100 bids or 100 goods. This is unfortunate but expected, since the WDP problem is at least exponential for a deterministic algorithm.

4.2 Two Atom

4.2.1 Revenue

TwoAtom serves as a ground truth for our optimal revenue. As we see from figure 5, IBM as configured reaches within 5% of the optimal revenue. Our greedy algorithm also does

	Revenue (IBM)	Revenue (Greedy)	100*(Greedy/IBM)
100-100	4269.457798	3533.654538	0.827658852
100-500	6670.29128	5460.742156	0.818666221
100-1000	6996.1497	5481.048522	0.783437856
100-2000	8141.90536	6336.7991	0.77829437
100-4000	8609.1337	7062.053354	0.820297791
500-100	11550.13356	10297.16018	0.891518711
500-500*	20551.83076	16560.8585	0.805809404
500-1000*	25118.00325	21472.04355	0.854846754
500-2000*	27999.82205	23907.71996	0.853852568
500-4000*	32513.61171	27394.88201	0.842566561
1000-100	14276.81113	12849.99327	0.900060466
1000-500*	33886.94107	28357.07939	0.836814374
1000-1000*	39812.05349	34692.88185	0.871416539
1000-2000*	47792.90583	40612.46727	0.849759322
1000-4000*	54527.75367	47450.57491	0.870209604

	Revenue (IBM)	Revenue (Greedy)	100*(Greedy/IBM)
100-100	4299.217252	3481.97536	0.809909143
100-500	6370.40116	4958.92428	0.778432026
100-1000	7280.684078	5760.84734	0.791250833
100-2000	7890.12272	6220.65466	0.788410381
100-4000	9022.78592	7088.42076	0.785613315
500-100	11338.17428	9756.49814	0.860499927
500-500*	21752.36478	17539.51092	0.806326627
500-1000*	24442.86658	19940.45384	0.815798498
500-2000*	27989.61545	23251.13052	0.830705608
500-4000*	33418.16878	28493.30824	0.852629252
1000-100	15552.4292	14008.81138	0.900747478
1000-500*	32518.24922	27386.54598	0.842190051
1000-1000*	38328.5987	32711.81696	0.853457159
1000-2000*	46971.96357	40924.45383	0.87125278
1000-4000*	58313.29638	50529.65511	0.866520301

Figure 4. Average revenue data by numgoods-numbids for arbitrary distribution. The top table contains no dominated bids.

	Revenue (TwoAtom)	% Total (TwoAtom)	Revenue (IBM)	% Total (IBM)	Revenue (Greedy)	% Total (Greedy)
100-100	24382447	100	24382447	100	23008526	94.36512258
100-500	41972664.2	100	41690442	99.3276048	37963441	90.44801354
100-1000	46388567.8	100	46148855.8	99.483252	42683932.8	92.0139052
100-2000	48171385.4	100	47529961	98.6684535	45982562.4	95.45617594
100-4000	49508792.2	100	49491821.2	99.9657212	48601987.4	98.16839644
500-100	37037270.8	100	37037270.8	100	36760162	99.25181096
500-500	118063865.6	100	118063865.6	100	112771198	95.51711476
500-1000	162538109.8	100	162531601	99.9959955	150197731.8	92.40770179
500-2000	199501228	100	199271274	99.8847355	182894074.6	91.67566357
500-4000	224026648.4	100	223485677	99.7585236	205861075	91.89133367
1000-100	43500951.8	100	43500951.8	100	43342243.4	99.63516109
1000-500	156516040.8	100	156516040.8	100	152437100.8	97.39391568
1000-1000	237749623.8	100	237736533.6	99.9944941	223852973.2	94.15492215
1000-2000	324535795.8	100	324457454.2	99.9758604	300394220.6	92.56119802
1000-4000	400526070	100	400168768.4	99.9107919	364764245	91.07128657

	Revenue (TwoAtom)	% Total (TwoAtom)	Revenue (IBM)	% Total (IBM)	Revenue (Greedy)	% Total (Greedy)
100-100	24166404	100	24166404	100	23015085.4	95.23587125
100-500	42411842.4	100	41799954.8	98.55727182	39372011.4	92.83258914
100-1000	45762496.4	100	45452448	99.32248364	41843655.2	91.43656595
100-2000	47797085.2	100	47463260.4	99.30157917	45129766.6	94.41949527
100-4000	48896157.4	100	48735389.2	99.67120484	47303986.2	96.7437703
500-100	41201526.8	100	41201526.8	100	40857988.2	99.16619935
500-500	118729676.6	100	118729676.6	100	112966884	95.14629134
500-1000	162298575.2	100	162249483.2	99.96975204	150147462.8	92.51311209
500-2000	200265047.6	100	200083997.8	99.90959491	180955147.4	90.35782807
500-4000	223510334.8	100	222390699.4	99.49906773	205122366.2	91.7731014
1000-100	42631767.6	100	42631767.6	100	42574332.2	99.86527558
1000-500	155257242	100	155257242	100	151132362.6	97.34319678
1000-1000	237084999.2	100	237084999.2	100	224896270	94.85892012
1000-2000	327945589.4	100	327854030.2	99.97208098	302151059.4	92.13450925
1000-4000	396450298.4	100	395748246.2	99.82291546	359478494.2	90.67429023

Figure 5. Average revenue data by numgoods-numbids for two atom distribution. The top table contains no dominated bids.

	Revenue (IBM)	Revenue (Greedy)	100*(Greedy/IBM)
100-100	16390.7761	14688.13394	0.896121932
100-500	26388.732	23614.07804	0.894854593
100-1000	29420.54	25786.94372	0.876494576
100-2000	30599.0612	27839.4142	0.909812691
100-4000	31376.7524	29237.60052	0.931823668
500-100	33584.9725	33033.78075	0.983588143
500-500	83920.14846	76968.76251	0.917166663
500-1000	106379.1173	93760.83444	0.881383836
500-2000	128073.3814	111433.2732	0.87007364
500-4000*	142243.9198	126585.5006	0.889918534
1000-100	37308.09182	36739.13318	0.984749726
1000-500	119177.2637	111167.4411	0.932790682
1000-1000	163951.6623	150207.5214	0.916169555
1000-2000	212616.8053	189055.4709	0.889184045
1000-4000	254031.7171	222092.8833	0.87427226

	Revenue (IBM)	Revenue (Greedy)	100*(Greedy/IBM)
100-100	15902.7602	14624.47274	0.919618516
100-500	26664.5638	23010.685	0.862968739
100-1000	29185.9248	25448.4251	0.871941707
100-2000	30437.805	27474.4656	0.902642802
100-4000	31458.1996	28908.10244	0.918936964
500-100	32395.22546	31548.32852	0.973857353
500-500	81291.37479	75148.9457	0.92443935
500-1000	106676.3661	95122.48016	0.891692168
500-2000	126699.9649	111712.2709	0.881707197
500-4000	142621.7363	125798.6125	0.88204376
1000-100	39443.57446	39324.94806	0.996992504
1000-500	119311.2468	112537.6048	0.943227129
1000-1000	162705.2759	148357.5599	0.911817758
1000-2000	209204.6168	187596.8384	0.89671462
1000-4000*	253685.8714	222478.2421	0.876983179

Figure 6. Average revenue data by numgoods-numbids for three atom distributions. The top table contains no dominated bids. Certain numgoods-numbids (indicated by *) had IBM CPLEX runs that were terminated after 300 seconds and thus may have a higher revenue than reported.

rather well, consistently staying within 10% of our optimal revenue.

4.2.2 Time

Figure 3 illustrates that the greedy algorithm easily is the fastest of the three algorithms. IBM is second fastest followed by TwoAtom.

We must note that some of the timing difference between TwoAtom and CPLEX is caused the use of python for TwoAtom vs. an industry solver for the IP problem. However, interestingly enough, it appears that CPLEX does not require significantly more time as the size of the problem increases, which is encouraging for distributions with primarily two-atom bids.

4.3 Three Atom

4.3.1 Revenue

CPLEX serves as a our ground truth for our optimal revenue, as we are certain all results are within a 5% factor of optimal.

In three-atom distributions, greedy does fairly well, staying within a factor of 15% away from the optimal revenue (see figure 6). However, it seems that as the size of the problem increases, the factor away from optimal also increase. Again, this is because the WDP approximation problem is NP-hard, but it's unfortunate that even the smallest instance of the NP-hard problem already exhibits some signs of this behavior for an albeit simple approximation algorithm.

4.3.2 Time

CPLEX, although it only had one instance where it timed out, exhibits exponential growth as the progblem size increases, which is as expected. However, CPLEX doesn't blow up nearly as fast as it does in the arbitrary distribution, which is good for practical purposes, since in a large multi-good auction, we could possible expect the length of bids to be rather small.

Greedy, as shown in the other two distributions, does very well time-wise.

5. Analysis

Figure 2 and 3 illustrate some noticable trends. For a given number of atoms and number of bids, the IBM algorithm achieves a higher total revenue than does the greedy algorithm, as expected. The 2-atom solution achieves the optimal revenue for bids of exactly 2 atoms, with the IBM solver very close to the mark set by the 2-atom solution. Greedy lags somewhat further behind both 2-atom and the IBM solver, and it especially lags behind the IBM solver for bids of arbitrary size. Additionally, the effect from type of algorithm on revenue is smaller than the effect of adding more atoms.

For timing, the greedy solution far outstrips the other types of solutions in all categories, followed by the IBM solver and finally by 2-atom. The effect from type of algorithm on time taken to solve is much greater than the effect of adding more atoms.

6. Conclusions

From the work we have done, we have determined the following:

- Having dominated bids does not appear to affect the revenue nor the time of any of the algorithms.
- Using the two-atom solution did not appear to provide any significant benefit over the IBM solver, neither in terms of revenue, nor in terms of runtime.
- Greedy provides a significant runtime advantage over the IBM solver, at the cost of a significant portion of revenue.

7. Future Work

Although we began exploring the max three-atom version of the WDP, we ultimately did not end up implementing it, as the problem is NP-Complete. A future direction this project could certainly take is exploring ways to optimize solutions to this and other related NP-Complete problems, including 3-set packing.

References

[1] Leyton-Brown, K., Pearson, M., Shoham, Y. (2000, October). Towards a universal test suite for combinatorial auction algorithms. In Proceedings of the 2nd ACM conference on Electronic commerce (pp. 66-76). ACM.

[2] Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. Artificial intelligence, 135(1), 1-54.

[3] Van Rantwijk, J. (2013, April 07). Maximum Weighted Matching Retrieved May 01, 2016, from `http://jorisvr.nl/article/maximum-matching`.