WeaveNet for Approximating Assignment Problems: Appendix

A Further implementation details

A.1 Scaling function

p_{ij}^A has a value range of $[1, N]$, which depends on the size of the input problem instance. Moreover, general deep-learning frameworks (e.g., PyTorch and TensorFlow) initialize model weights under the assumption where the maximum value in the input data is around 1.0. Hence, to re-scale p_{ij}^A to s_{ij}^A, we used the following function h, which is

$$\begin{aligned} h(p_{ij}^A) &= ((1 - C_{min})(N - p_{ij}^A))/N + C_{min} \\ h(p_{ji}^B) &= ((1 - C_{min})(N - p_{ji}^B))/N + C_{min}, \end{aligned} \tag{11}$$

where C_{min} should be a constant value in range $(0, 1)$ and is set to 0.1 in our experiments.

A.2 Theoretical evaluation for the calculation cost

A.2.1 Computational complexity

With $D_{\max} = \max\{D', D\}$, the theoretical calculation cost of a single set encoder on a single core processing unit is $O(N^2 D_{\max}^2)$, where a single linear operation in the convolution requires D^2 and we repeat it for $2N \times M$ elements ($N \geq M$). The other operations in WeaveNet, such as cross-concatenation, have only less cost. Hence, the entire WeaveNet has its calculation cost of $O(LN^2 D_{\max}^2)$.

We can reduce this into $O(L \log(N D_{\max}))$ with an ideal parallel processing unit. We have $2N \times M$ elements for the convolution and can linearly operate them independently. Each linear operation can be done in parallel except for weighted sum aggregation, which takes $O(\log(D_{\max}))$. Besides, we have a max-pooling operation, which consists of D parallel max operations. It can also be done in parallel and each max operation for N elements takes $O(\log N)$. Therefore, we can execute the calculation of a single set encoder in $O(\log(N) + \log(D_{\max})) = O(\log(N D_{max}))$.

A.3 Information propagation with WeaveNet

By stacking two or more FW layers, every latent feature of ij-th element in Z_ℓ^* ($\ell \geq 2$) has a receptive field that covers all over the preference lists. Fig. 8 illustrates the receptive field, where green elements are upstream components of the ij-th element, and E_0, E_1 are encoders that involve all the input into the calculation of each feature in the output sequence. Let us backtrack the path of back-propagation from the ij-th element in Z_2^A. It derives from the i-th row of Z_1^A and the i-th column of Z_1^B. Focusing on the i-th column (e.g., the green column) of Z_1^B, each element (in the k-th row) derives from all the elements in the k-th row of S^B, plus all the elements in the k-th column of S^A. In this way, all the elements in S^A and S^B contribute to a column of Z_1^B, then an element of Z_2^A. Symmetrically, all the elements in S^A and S^B contribute to a column of Z_1^A, then an element of Z_2^B. Hence, we can see that stacking two FW layers can cover the entire bipartite graph in its reference.

We also visualize the back-propagation path of DBM in Fig. 9. Note that DBM also satisfies the requirements (b), (c), and (d) in 4.1. There are two major difference between WeaveNet and DBM: their parameter efficiency and the choice of the local structure E_ℓ. DBM applies the weftwise and warpwise communications alternately in a single-stream architecture. Hence, each encoder is specialized in each direction. In contrast, every encoder in WeaveNet is trained for both directions, and thus it is **twice parameter efficient**. Also, the local structure of DBM is sub-optimal to encode the relative identity of each outgoing edge among $N \times M$ pairs. The impact of difference in the local structure appears as a performance gain in Fig. 5, from DBM-6 to SSWM-6.

A.4 Loss weights

Through the experiments, we set the loss weights $\lambda_m = 1.0$, $\lambda_s = 0.7$, and $\lambda_f = \lambda_b = 0.01$ to train any learning-based methods. We adjusted these parameters in the following process.

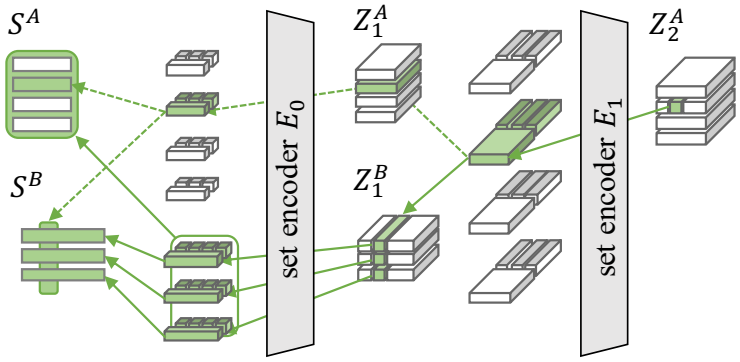

Figure 8: Backward path of WeaveNet (two-stream architecture) from ij-th feature in Z_2^A. The upper stream terminates at i-th row of S^A and i-th column of S^B, which are the weights on outgoing and incoming edges of the agent a_i, respectively. The lower stream refers to all the elements in S^A and S^B.

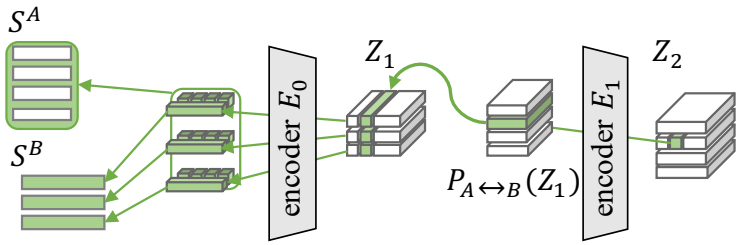

Figure 9: Backward path of DBM (single-stream architecture) from ij-th feature in Z_2, where the stream refers to all the elements in S^A and S^B. This structure is common with SSWN.

As a preliminary model, we prepared WN-15 with a set encoder with $D = 64$ and $D' = 256$. In this investigation, we used a balanced dataset UU and the most biased dataset UD.

Because we experimentally found the tendency that the model hardly outputs a stably matched solution without minimizing $\mathcal{L}_m$, we first tried to fix $\mathcal{L}_m$. Fig. 10 shows the success rate of stable matching with different $\mathcal{L}_m$ in the range $\{0.001,\ 0.005,\ 0.010,\ 0.050,\ 0.100,\ 0.500,\ 1.000\}$, where WN-15($n$) is the model trained and validated with the samples of $N = n$. From this result, we decided to set $\lambda_m = 1.0$ and use it as the maximum weight among the loss weights.

Next, fixing $\lambda_m = 1.0$, we observed the trend in success rate of stable matching against λ_s. Fig. 11 shows our investigation of $\mathcal{L}_s$ in the range $\{0.01, 0.10, 0.30, 0.50, 1.00\}$. Here, WN-dual is a WeaveNet variant to deal with the inconsistent distributions of UD (see B.2 for details of WN-dual). As a result, we found that λ_s should simply be large enough (ca. $\lambda_s \geq 0.30$).

To investigate sensitivity of the model against λ_f and λ_b, in this experiment, we set $\lambda_s = 0.3$ as the minimum satisfiable value in Fig. 11[7], and obtained Figures 12 and 13. From these figures, we found that too-strong weights for fairness may disrupt stability, which matches a theoretical expectation. Hence, we decided to use $\lambda_f = \lambda_b = 0.01$, which achieved the highest success rate of stable matching in the search range of $\{0.01, 0.02, 0.03\}$.

A.5 Network architecture

We can customize the shape of WeaveNet architecture finely by using set encoders with different shapes in each FW layer; however, we avoided such a complex architecture design to simplify the analysis and decided to use common-shape set encoders for all L FW layers. With this setting, we have only three hyper-parameters to decide the architecture: L, D, and D'. The shortcut path is regularly connected to the input of ℓ-th layer with an even number of ℓ.

[7] We used $\lambda_s = 0.7$ in any other experiments to stabilize the results, as stated in Section 5.

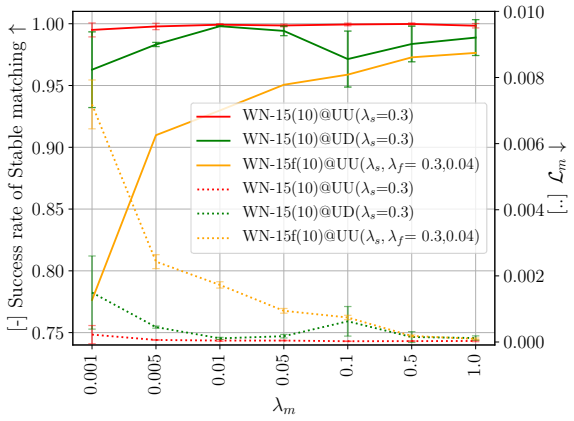

Figure 10: Sensitivity against λ_m

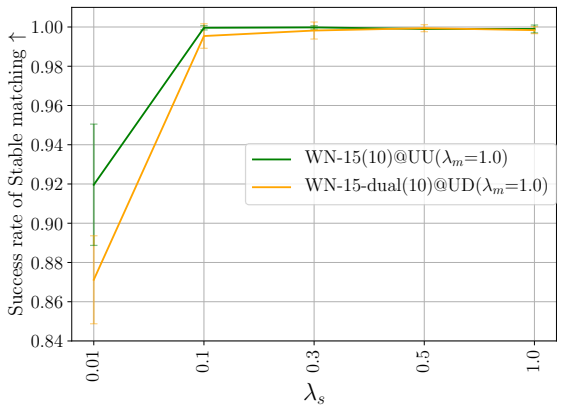

Figure 11: Sensitivity against λ_s

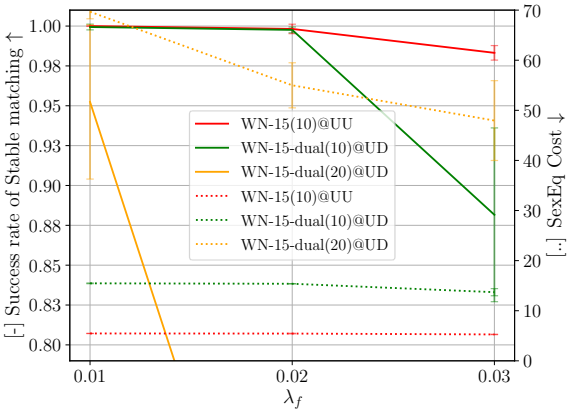

Figure 12: Sensitivity against λ_f

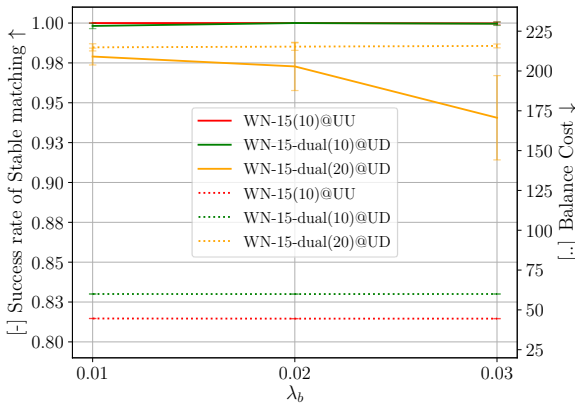

Figure 13: Sensitivity against λ_b

Table 5: Comparison in parameter efficiency among different shape architectures. Deep achieved the best success rate in stable matching despite its smallest architecture.

Name	L	D	D'	# of params.	Stably Matched (%)
Deep	30	22	44	117k	95.7%
Wide1	15	32	64	119k	76.0%
Wide2	15	24	98	120k	73.1%

Table 5 shows the success rate of stable matching after 100,000 iterations of training. Here, we drew samples from the UU distribution with $N = 30$ for both training and validation. The result shows that the deepest model performs the best despite its smallest number of parameters. Hence, we decided to use the set encoder that has $D \leq 32$ and $D' = 2D$.

We further investigated the impact of network depth on the problem. To observe how the strongly NP-hard target increases the difficulty in optimization, we plotted both results by WN-L and WN-Lf with $L \in \{6, 18, 30, 42, 54, 60\}$. Fig. 14 shows the trend of success rate against different L. Here, the models were trained and validated with the UU dataset at $N = 20$. We can see from the result that $L = 6$ is not enough to stably match samples of $N = 20$, but $L = 18$ is enough if only for that purpose. Besides, we observed that a deeper stack of layers tends to improve SEq slightly.

A.6 Hyper-parameter settings for the other baselines

Here, we denote the hyper-parameters for learning-based baseline methods, including those used in B.1 and B.2. The detail is summarized in Table 6. Note that MLP could not converge when more than three hidden layers are stacked. Hence, we set the number of layers to be three, as used in Li (2019).

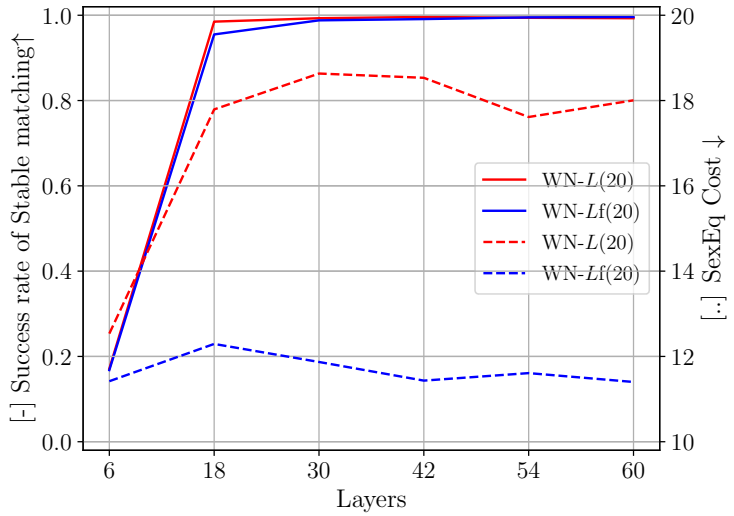

Figure 14: Success rate of stable matching (solid, ↑) and SEq (dashed, ↓) according to L.

Table 6: Hyper-parameters for each baseline, where D' represents the length of key and query features for self-attention, and the max-pooling input for the set encoder. Note that the D-dimentional feature of WN is cross-concatenated and processed as a $2D$-dimensional features in the encoders.

Model	Encoder	D	D'	Res.	# of params.
MLP-3	dense layer	100	-	w/o	28k
GIN-2	graph conv.	44	-	w/o	29k
DBM-6	max pooling	48	-	w/o	29k
DBM_A-6	self-attention	48	32	w/	28k
SSWN-6	set encoder	48	48	w/o	25k
SSWN-60	set encoder	64	64	w/	740k
WN-6	set encoder	24	48	w/o	25k
WN_A-6	self-attention	32	28	w/	29k
WN-18	set encoder	32	64	w/	143k
WN-60	set encoder	32	64	w/	493k
WN-80	set encoder	32	64	w/	659k

For GIN Xu *et al.* (2019), we put two graph convolutional layers because the original paper reported it performs best and some other papers pointed our the oversmoothing problem of GCNs Li *et al.* (2018); Oono and Suzuki (2020). The layers are followed by a single linear layer, which output $\hat{m}$.

Note that we used the same loss weights decided in A.4 for all the methods since the loss weights derive from the task property rather than the architecture. For simplicity in comparison, we also used the three hyper-parameters, L, D, and D', to identify the architecture following to A.5. We experimentally decided whether to use the residual structure or not for the models which have more than four layers. Hence, the shown results are always better ones.

B Additional experiments

B.1 Additional learning-based baselines and ablations.

We compared WeaveNet to more comprehensive baselines: a variation of MLP, DBM, and WeaveNet. Fig. 15 shows the results.

MLP-O is a variant of MLP that faithfully follows the loss functions proposed in Li (2019). The difference is in the matrix constraint loss function $\mathcal{L}_m$. Ours defined in Eq. (7) is cosine-distance

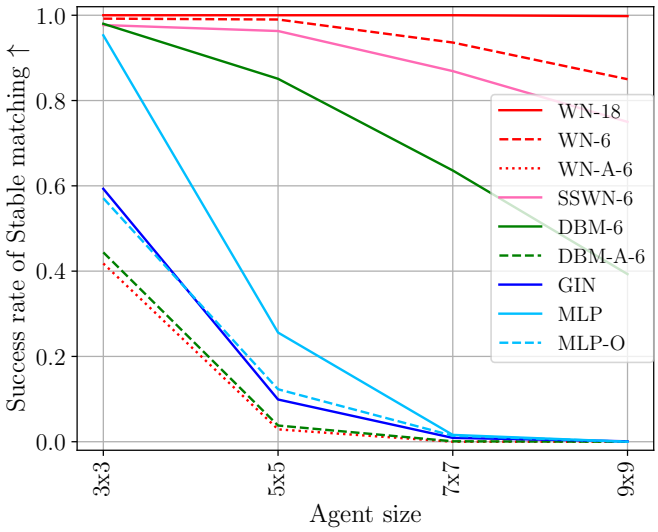

Figure 15: Change in success rate of stable matching (↑) according to N, as a complement of Fig. 5.

based, while the one proposed in the original paper is Euclidean-distance based, which is

$$\mathcal{L}_m(\hat{m}^A, \hat{m}^B) = \sum_{\{i,j\} \in A \times B} |\hat{m}^A_{ij} - \hat{m}^B_{ji}|. \tag{12}$$

The above loss function restricts i-th column and i-th row of $\hat{m}$ to be the same point in the $N \times M$ space. In other words, the column and row must match in a scale variant manner. We considered that this is unnecessarily strict for maintaining $\hat{m}$ to be symmetric. Hence we adopted the cosine distance for $\mathcal{L}_m$. In Fig. 15, we observed that MLP has a clear performance gain from MLP-O by our cosine-distance-based $\mathcal{L}_m$.

DBM proposed in Gibbons *et al.* (2019) has two variations; one uses max-pooling with one convolutional layer[8] as the layer-wise encoder (DBM) and self-attention (DBM_A). We also integrated self-attention into the WeaveNet architecture (**WN_A-6**) to complete the comparison of the encoder choice. As a result, we have confirmed that self-attention does not work well for stable matching problem.

B.2 Additional ablation in $N = 20,\ 30$

We provide a detailed ablation study to finely analyze the symmetric and asymmetric variants, the effect of two-stream architecture, the size of training samples, and the binarization operation. Here, we trained all models with $N = 30$.

First, we refer to the symmetric and asymmetric variants of WN-60f/b as **WN-60f/b-sym** and **WN-60f/b-asym**, respectively. Note that both models have the two-stream architecture, but **sym** shared batch normalization layers and **asym** does not. In addition, **asym** has an additional channel in inputs for side-identifiable code. In addition to them, we further prepared **WN-60f/b-dual**, a model that separately holds the entire set encoder for Z^A_ℓ and Z^B_ℓ at each layer to deal with asymmetric inputs. Since UD is a highly asymmetric dataset, the symmetric variant (WN-60f/b-sym) does not work appropriately while the asymmetric variants (*asym* and *dual*) could deal with it. In the dataset Lib, where we assumed an unknown bias strength, *asym* worked as well as *sym* and *dual*. This result showed that we can safely apply the asymmetric variation for any situation. Note that the performance gain of **asym** (or **dual**) from **sym** is achieved by the side-inequality. In other words, for an ethical application, we can say the gap is the cost of equality when applying **sym** for such a biased situation.

[8]The max-pooled feature is copied and concatenated to the input feature as the set encoder, but it does not have convolution before the max-pooling operation.

Table 7: Ablation study for the network architecture difference with $pSEq$ (↓), SEq (↓), and the success rate of stable matching (↑). The scores shown in Table 1 are underlined.

Agents ($N \times M$)	20×20					30×30				
Datasets (Dist. Type)	UU	DD	GG	UD	Lib	UU	DD	GG	UD	Lib
WN-60f-sym										
$pSEq$	**12.16**	**6.53**	**15.56**	82.48	14.59	**18.30**	**10.52**	**27.39**	210.06	**22.14**
SEq	11.44	6.32	15.34	81.21	14.39	16.07	9.64	26.46	194.24	21.16
Stably Matched (%)	99.10	99.40	99.40	96.00	99.50	98.10	99.00	98.00	81.50	98.80
WN-60f-asym										
$pSEq$	-	-	-	**71.34**	**14.53**	-	-	-	**170.35**	22.17
SEq	-	-	-	71.18	14.44	-	-	-	162.61	21.29
Stably Matched (%)	-	-	-	99.50	99.80	-	-	-	93.90	98.60
WN-60f-dual										
$pSEq$	-	-	-	72.42	14.71	-	-	-	173.46	23.92
SEq	-	-	-	71.25	14.58	-	-	-	163.71	22.12
Stably Matched (%)	-	-	-	98.50	99.60	-	-	-	94.80	97.50
SSWN-60f										
$pSEq$	15.06	7.74	17.65	74.50	16.02	34.34	14.57	33.91	197.04	27.25
SEq	12.82	7.41	16.62	70.94	15.20	19.13	12.76	28.33	160.83	22.45
Stably Matched (%)	95.80	99.20	96.50	92.40	97.90	90.00	97.50	89.00	69.20	92.30
WN-60f+Hung										
$pSEq$	11.70	6.36	15.37	71.23	14.45	16.35	9.75	26.90	168.87	21.62
SEq	11.51	6.35	15.37	71.18	14.45	16.14	9.69	26.58	162.73	21.32
Stably Matched (%)	99.80	99.90	100.00	99.60	100.00	99.70	99.90	99.90	95.20	99.20

Table 8: Ablation study for the network architecture difference with $pBal$ (↓), Bal(↓), and the success rate of stable matching (↑). The scores shown in Table 2 are underlined. The best scores are in bold.

Agents ($N \times M$)	20×20					30×30				
Datasets (Dist. Type)	UU	DD	GG	UD	Lib	UU	DD	GG	UD	Lib
WN-60b-sym										
$pBal$	**72.33**	**138.75**	**106.65**	142.37	65.82	**140.40**	**301.59**	**223.02**	322.98	**127.79**
Bal	71.29	138.57	106.50	141.96	65.59	137.70	301.08	221.12	315.51	127.05
Stably Matched (%)	98.00	99.10	98.60	97.30	98.90	97.90	98.60	93.70	80.30	98.10
WN-60b-asym										
$pBal$	-	-	-	**140.79**	65.84	-	-	-	**313.59**	127.93
Bal	-	-	-	140.72	65.63	-	-	-	313.11	127.12
Stably Matched (%)	-	-	-	99.80	99.10	-	-	-	98.80	98.00
WN-60b-dual										
$pBal$	-	-	-	141.23	**65.79**	-	-	-	315.71	128.23
Bal	-	-	-	141.20	65.70	-	-	-	314.79	127.25
Stably Matched (%)	-	-	-	99.60	99.30	-	-	-	98.90	97.90
SSWN-60b										
$pBal$	79.06	139.59	107.90	141.48	67.22	169.22	304.52	225.06	317.31	132.19
Bal	76.54	139.29	107.58	141.37	66.99	150.88	302.90	223.62	314.91	129.52
Stably Matched (%)	92.60	98.50	98.40	99.60	98.60	79.80	96.70	93.90	94.0	92.80
WN-60b+Hung										
$pBal$	71.51	138.62	106.58	140.76	65.68	138.62	301.16	222.01	313.46	127.32
Bal	71.45	138.60	106.55	140.73	65.68	137.92	301.14	221.42	313.13	127.22
Stably Matched (%)	99.90	99.90	99.60	99.90	100.00	99.00	99.90	96.90	99.30	99.20

Comparing *asym* with *dual*, *asym* scored slightly smaller $pSEq$ and $pBal$ costs despite the half number of model parameters. This result experimentally confirmed the advantage of a parameter-efficient model structure.

Second, **SSWN-60**, a single-stream WeaveNet with set encoders, was prepared to measure the contribution of the two-stream architecture. SSWN-60f/b achieved consistently worse result than

Table 9: Number of blocking pairs in the estimated matching with UU $N = 100$. Fail counts outputs that are not a one-to-one matching. Outputs with no blocking pairs are stable matchings.

#Block. Pairs	WN-80f	+Hung.	WN-80b	+Hung.
0	84.4%	89.4%	73.2%	80.8%
1	2.2%	4.6%	6.7%	10.9%
2	0.0%	0.4%	0.3%	1.2%
≥ 3	0.0%	5.6%	0.0%	7.1%
Fail	13.4%	-	19.8%	-

Table 10: Elapsed time for training and testing the models.

Elapsed Time	for training 200,000 iters.	for test 1,000 samples
WN-6 ($N = 10$)	10.39 hours	5.82 s
WN-18 ($N = 10$)	13.10 hours	14.83 s
WN-60 ($N = 20$)	30.10 hours	75.15 s
WN-60 ($N = 30$)	43.58 hours	74.10 s
WN-80 ($N = 100$)	111.10 hours	103.66 s

WN-60f/b (underlined), and collapsed in UD ($N = 30$) in Table 1 and UU ($N = 30$) in Table 2. These results experimentally proved the importance of the two-stream structure.

Finally, to avoid an increase of theoretical calculation cost, WeaveNet applied argmax operation to binarize $\hat{m}$ rather than the Hungarian algorithm, whose calculation cost is $O(N^3)$. To demonstrate how well the network maintains $\mathrm{argmax}(\hat{m})$ to be a one-to-one matching, we compared the result with those obtained with the Hungarian algorithm (**WN-60f+Hung**). Again, the asymmetric variant was applied for UD and Lib. From the result, we confirmed that WN-60f/b+Hung, the reference algorithm, achieved slightly better performance than those without the Hungarian algorithm, but the gain is quite limited. This result implies that the argmax binarization works closely to the Hungarian algorithm without the calculation cost.

To obtain further insight, we prepared Table 9, which summarizes the difference w/ and w/o the Hungarian algorithm at $N = 100$. In this setting, we have more samples with which WN-80f/b fails to even obtain one-to-one matching by the argmax binarization. Binarization by the Hungarian algorithm forces such failure estimation $\hat{m}$ to be a matching; however we obtained only 5.0% additional stable matchings from the 13.4% failure cases with WN-80f, and 7.5% from 19.8% failure cases with WN-80b. A similar number of cases resulted in matchings with more than three blocking pairs.

From these results, we concluded that the Hungarian algorithm does not essentially improve the quality of fair stable matching, although it ensures $\hat{m}$ to be a one-to-one matching.

C Data for reproduction

C.1 Calculation time and computing infrastructure

We trained and tested the learning-based models used in the experiments on a single GPU (Tesla V100, memory size 16GB) mounted on NVIDIA DGX-1. We developed the environment on Ubuntu18.04.

The above training required less than 16GB of memory space in our setting as long as the batch size is no larger than 8. The training and inference time are summarized in Table 10.

C.2 Dataset

We implemented the data generator for UU, DD, GG, UD, and Lib following the explanation in Tziavelis *et al.* (2019) since they are not provided by the authors. In the process, we also extracted the distribution of the LibimSeTi dataset Brozovsky and Petricek (2007), where the LibimSeTi dataset is

accessible in http://konect.cc/networks/libimseti/[9]. The participants are recruited for this dataset by the authors of original paper. Any personal information is removed from the dataset. We filtered out data that do not have bidirectional preference rank, as stated in Tziavelis *et al.* (2019). We yielded the validation and test datasets with a fixed random seed to make them reproducible.

All the code for data generation is included in the submission (with the random seeds and fairness costs of each sample obtained by the traditional algorithmic baselines). They will become publicly available at the timing of publication of this paper.

[9]The download site by the original authors was closed after October 2021. Nonetheless, we can confirm its license at Internet archives, such as https://web.archive.org/web/20200630115514/http://www.occamslab.com/petricek/data/. There are multiple authorized re-distribution sites, such as the konect project, and the dataset is still available for researchers.