

Table 2
Running Time of BSPA for Different p Values.

Datasets	p									
	10		30		60		90		120	
	t(s)	gap	t(s)	gap	t(s)	gap	t(s)	gap	t(s)	gap
C	60.00	1.40	51.43	1.32	50.00	1.32	51.43	1.40	51.43	1.40
N	60.00	0.68	53.08	0.68	53.08	0.68	53.08	0.68	53.08	0.68
NT-N	132.86	2.66	75.43	2.66	60.86	2.66	60.00	2.66	60.00	2.66
NT-T	135.43	2.66	74.57	2.66	60.86	2.69	60.00	2.66	60.00	2.66
Avg	109.90	2.16	67.50	2.14	57.69	2.15	57.40	2.16	57.40	2.16
KR	117.50	2.70	70.00	2.70	60.00	2.70	60.00	2.70	60.00	2.70
BWMV	435.84	7.62	177.12	7.65	113.28	7.66	91.44	7.66	81.12	7.66
Avg	428.38	7.51	174.61	7.53	112.03	7.55	90.70	7.54	80.63	7.54

Table 3
BSPA Results Under the OF and RF Scenario.

Datasets	OF				RF			
	BSPA _(0.1,30,30,90)		BSPA _(0.1,60,60,90)		BSPA _(0.1,30,30,90)		BSPA _(0.1,60,60,90)	
	gap	t(s)	gap	t(s)	gap	t(s)	gap	t(s)
C	1.40	51.43	1.32	100.00	0.79	45.7	0.75	91.43
N	0.68	53.08	0.67	106.15	0.72	53.1	0.60	101.54
NT-N	2.66	60.00	2.59	120.00	2.14	60.0	2.04	120.00
NT-T	2.66	60.00	2.63	120.00	2.13	60.0	2.06	120.00
Avg.	2.16	57.40	2.10	114.23	1.69	56.3	1.61	111.92
KR	2.70	60.00	2.67	120.00	1.60	60.0	1.60	120.00
BWMV	7.62	91.44	7.56	181.68	3.03	72.7	2.79	141.12
Avg.	7.51	90.70	7.44	180.23	3.00	72.4	2.76	140.63

3.4. Experiment and analysis

Table 3 presents the computational results of the algorithm under both OF and RF scenarios. In this table, the notation BSPA_{0.1,30,30,90} denotes the execution of the algorithm with parameters $b = 0.1$, $T_1 = 30$ s, $T_3 = 30$ s, and $p = 90$. This notation is consistently applied across other tables.

As shown in Table 3, under both OF and RF scenarios, the proposed algorithm demonstrates a consistent reduction in the gap value across all datasets as search time increases. However, despite a twofold increase in search time, the reduction in the gap value remains marginal. Nevertheless, the gap values obtained by our algorithm across all datasets are within an acceptable range. Specifically, for the zero-waste datasets C, N, NT-N, and NT-T, the average fill length exceeds the minimum length by only 2.1%. For the non-zero-waste dataset KR, this deviation is merely 2.67%, while for the large-scale dataset BWMV, it remains limited to 7.57%. These results substantially outperform those of manual filling, indicating that the proposed method can be effectively applied to industrial material cutting to assist enterprises in reducing material waste during the cutting process.

In the previous experiments, we used a platform with a 256-core processor and 120 GB of RAM to run our algorithm. However, this does not imply that the algorithm

proposed in this paper is limited to such high-end configurations, as this setup was chosen solely to meet the specific requirements of the experiments. Our algorithm is capable of running on personal computers used in daily life and can even outperform the results obtained in the aforementioned experiments because personal computers usually have higher CPU frequency. Furthermore, if the algorithm is implemented using more efficient programming languages, such as C or C++, the performance of loading results within the same time limit will be further enhanced. Table 4 presents the algorithm's performance on a widely used PC with an Intel i7-13700K CPU, 16GB of RAM, and the Windows 11 operating system, using standard datasets.

4. Conclusions

This paper presents a beam search-based parallel method for solving the two-dimensional strip packing problem. Experimental results demonstrate that the proposed algorithm can find an efficient placement strategy to minimize the used length of the strip as much as possible. To facilitate future research, the code and datasets are available at <https://github.com/Yzhjdj/BSPA>