Submitted to *INFORMS Journal on Computing*

Supplement to "An Improved Combinatorial Benders Decomposition Algorithm for the Human-Robot Collaborative Assembly Line Balancing Problem"

Dian Huang, Zhaofang Mao*, Kan Fang*, Enyuan Fu
College of Management and Economics, Tianjin University, Tianjin 300072, China, huangdian@tju.edu.cn, maozhaofang@tju.edu.cn, kfang@tju.edu.cn, fey609333118@tju.edu.cn

Michael L. Pinedo
Stern School of Business, New York University, New York 10012, USA, mlp5@stern.nyu.edu

1. Proof of Propositions

PROPOSITION 1. *For any given station, if none of the tasks assigned to this station is subject to any precedence relationship with respect to any other and they all can be processed preemptively under any one of the three processing alternatives, then the minimum station time can be achieved by processing these tasks in parallel.*

Proof. As mentioned before, we know that $t_{ip_R} = \lceil 2t_{ip_H} \rceil$ and $t_{ip_C} = \lceil 0.7t_{ip_H} \rceil$ for any task i that can be processed by either automated or collaborative executions. Since each task can be processed preemptively, we can divide each task into two parts so that the first part is twice the workload of the second part. We then process these two parts concurrently, that is, we process the first one by manual execution and the second one by automated execution. This way, the human worker and the cobot will complete their assembly work at the same time, and the processing time of this task becomes $\lceil 2/3t_{ip_H} \rceil$, which is smaller than the ones by either fully automated or fully collaborative executions.

PROPOSITION 2. *Let*

$$\underline{C} = \max\left\{t_{\min}, \left\lceil \frac{T_{\min}}{m} \right\rceil\right\},$$

where $T_{\min}$ is the minimum total processing time of all the tasks, which is given by

$$T_{\min} = \begin{cases} T_C + T_H, & T_R \leq T_H, \\ T_C + \frac{3}{10}T_R + \frac{7}{10}T_H, & T_H < T_R \leq T_H + 10/7T_C \\ \frac{1}{3}T_R + \frac{2}{3}T_H + \frac{20}{21}T_c, & T_H + 10/7T_C < T_R. \end{cases}$$

Then $\underline{C}$ is a lower bound for the cycle time in the HRCALBP-II.

Proof. We assume that the number of cobots is enough and the precedence relationship can be neglected, then the minimum total processing time $T_{\min}$ can be obtained from the following three cases:

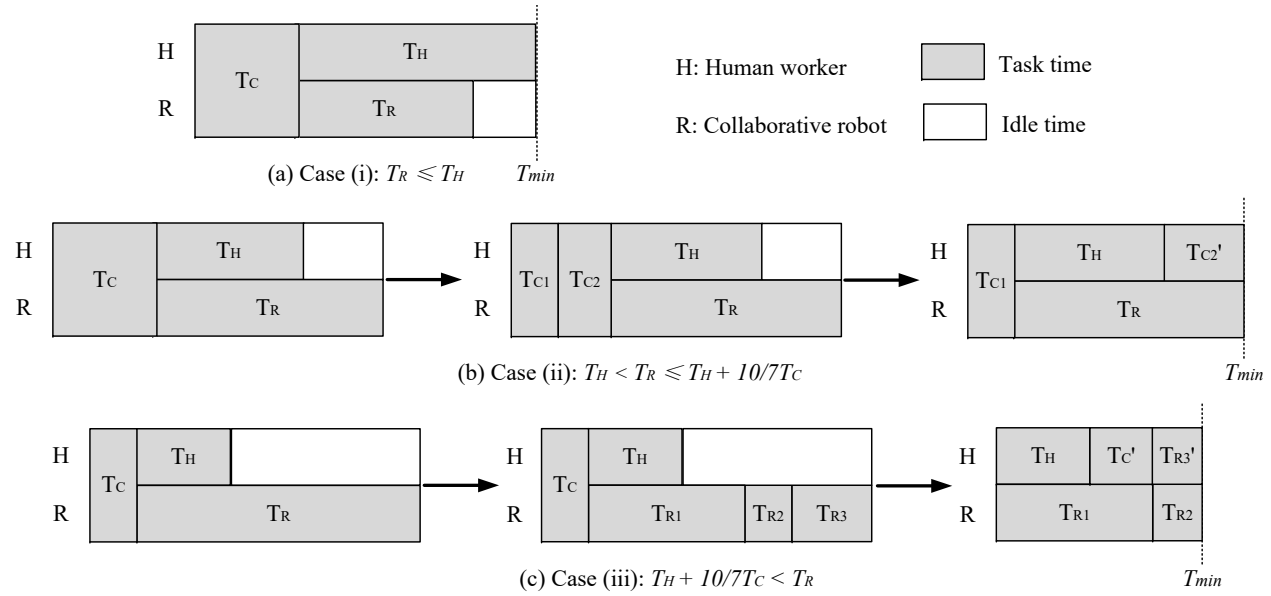

Figure A1 **Three Cases of $T_{\min}$.**

Case (i): $T_R \leq T_H$. In this case, Figure A1(a) shows the corresponding schedule with minimum total processing time by processing the tasks in $\mathcal{I}_H$ and $\mathcal{I}_R$ concurrently, that is, the tasks in $\mathcal{I}_H$ and $\mathcal{I}_R$ are processed by human worker and cobot respectively, and the total processing time $T_{\min}$ is $T_C + T_H$.

Case (ii): $T_H < T_R \leq T_H + 10/7T_C$. In this case, we may remove some of the tasks from $\mathcal{I}_C$ and process them manually, so that the human worker and cobot can finish at the same time, and thus reduce $T_{\min}$. Figure A1(b) shows the change of schedule by performing the above procedure, from which we can see that $T_{C1} + T_{C2} = T_C$, $T_{C2}{}' = 10/7T_{C2}$ and

$T_H + T_{C2}{}' = T_R$. We find that $T_{C1} = T_C - 7/10T_R + 7/10T_H$, and $T_{\min} = T_{C1} + T_R$; that is, $T_{\min} = T_C + 3/10T_R + 7/10T_H$.

Case (iii): $T_H + 10/7T_C < T_R$. In this case, we can remove all the tasks from $\mathcal{I}_C$ and some of the tasks from $\mathcal{I}_R$, and process these manually, so that the human worker and cobot can process tasks concurrently and finish their assembly work at the same time. Figure A1(c) shows the change of schedule by performing the above procedure, from which we can see that $T_C{}' = 10/7T_C$ and $T_{R3}{}' = 0.5T_{R3}$. Thus we have $T_{\min} = 1/3T_R + 2/3T_H + 20/21T_C$.

PROPOSITION 3. *The cycle time $C(\sigma)$ obtained by Algorithm 1 is an upper bound for the HRCALBP-II.*

Proof. Obviously, the schedule obtained by Algorithm 1 is feasible to the HRCALBP-II, therefore $C(\sigma)$ is an upper bound for the cycle time for HRCALBP-II.

PROPOSITION 4. *For the HRCALBP, the maximum decrease β_{ip} obtained by removing task i with processing alternative p from the station is equal to t_{ip}.*

Proof. According to the processing alternative p of task i, the maximum decrease β_{ip} can be calculated as follows:

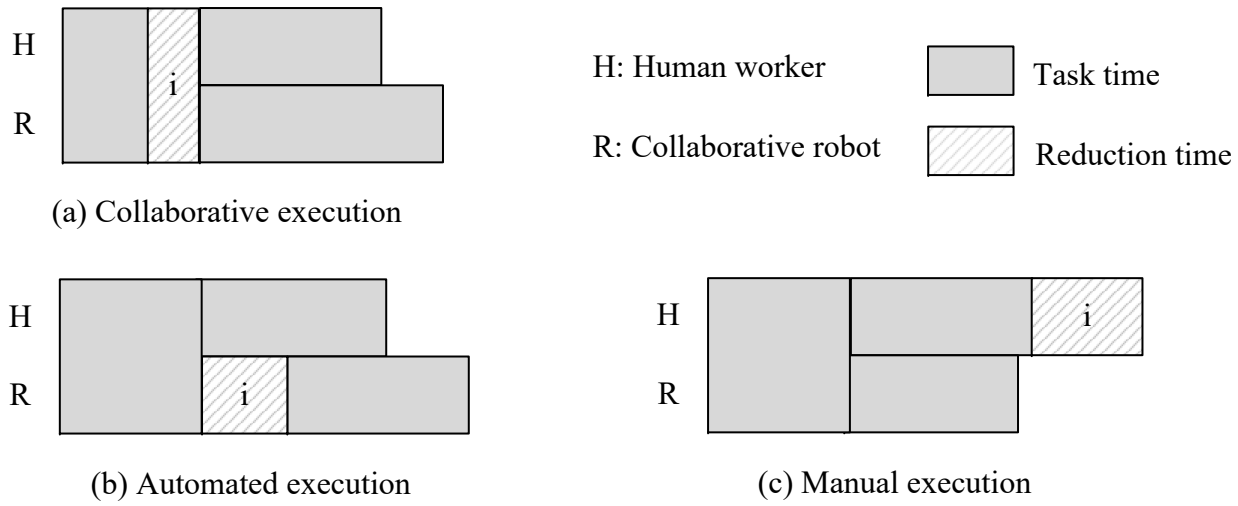

Figure A2 Illustration for the Calculation of the Maximum Decrease.

(i) If task i is processed collaboratively as shown in Figure A2(a), then it is easy to see that β_{ipC} is exactly equal to t_{ipC}.

(ii) If task i is processed by the cobot (see Figure A2(b)) or by the human worker (see Figure A2(c)), then the decrease of station time is less than or equal to the processing time

of this task. In particular, if task i is processed consecutively by the human worker or the cobot, then the decrease β_{ip} is also equal to t_{ip} for $p \in p_H, p_R$.

Combining the above two results, we have $\beta_{ip} = t_{ip}$ for $p \in \mathcal{P}$.

PROPOSITION 5. *Cut (14) is a valid cut.*

Proof. This proposition can be proved by using similar arguments as in the proof of optimality cuts proposed by Zohali et al. (2022) (see Proposition 5 in their paper).

2. The Summary of Notations

Table A1 The Summary of Notations.

Notation	Description
Indices:	
i, j, h	index of task
k, k', g	index of station
p, p'	index of processing alternative
Parameters:	
$\mathcal{I}$	Set of tasks, i.e., $\mathcal{I} = \{1, \ldots, n\}$
$\mathcal{K}$	Set of stations, i.e., $\mathcal{K} = \{1, \ldots, m\}$
$\mathcal{P}$	Set of processing alternatives, i.e., $\mathcal{P} = \{p_H, p_R, p_C\}$
E	Set of immediate precedence relations, i.e., arcs of the precedence diagram
$PR_i (PR_i^a)$	Set of immediate (all) predecessors of task i
$SU_i (SU_i^a)$	Set of immediate (all) successors of tasks i
PR^0	Set of tasks that have no immediate predecessors, $PR^0 = \{i \in \mathcal{I} \mid PR_i = \emptyset\}$
SU^0	Set of tasks that have no immediate successors, $SU^0 = \{i \in \mathcal{I} \mid SU_i = \emptyset\}$
t_{ip}	Processing time of task $i \in \mathcal{I}$ with processing alternative $p \in \mathcal{P}$
$t_i^{\min}$	Minimum processing time of task i under different alternatives, i.e., $t_i^{\min} = \min\{t_{ip} \mid p \in \mathcal{P}\}$
$\underline{C}$	Lower bound for the cycle time
$\overline{C}$	Upper bound for the cycle time
q	Maximum number of cobots to be allocated
LP_{ij}	Processing time of all the tasks on the longest path from i to j
MS_{ij}	Minimum difference of stations between tasks i and j
ES_i	Earliest station for task i
LS_i	Latest station for task i
FS_i	Set of stations to which task i is feasibly assignable
FT_k	Set of tasks feasibly assignable to station k

3. Algorithms

Algorithm 1 A greedy-based heuristic for the HRCALBP-II

Use the SALOME method to solve the HRCALBP-II by restricting that all the tasks must be processed by human workers. Let $\mathcal{I}_k$ denote the set of tasks that are assigned to station k.
Calculate the corresponding station time for each station k, i.e., the total processing time of tasks at this station, and denote it as st_k.
Allocate cobots to stations in nonincreasing order of station times unless all cobots have been assigned.
for $k \in \mathcal{K}$ **do**
  if A cobot is assigned to station k **then**
    while $\mathcal{I}_k \neq \emptyset$ **do**
      Find the task i^* with the earliest start time, that is, task i^* has no predecessors or the completion time of its predecessors is the smallest one.
      Calculate the completion time of task i^* under three processing alternatives respectively, and select the alternative with the smallest completion time to process task i^*.
      $\mathcal{I}_k \leftarrow \mathcal{I}_k \backslash \{i^*\}$.
Denote the corresponding schedule of all the tasks as schedule σ.
Update the station time st_k for each station k under schedule σ.
Output the cycle time of schedule σ, that is, $C(\sigma) = \max_{k \in \mathcal{K}}\{st_k\}$.

Algorithm 2 The algorithmic outline of the TPLS method

Given the main function iteration number whileIndex and the current lower bound $\underline{C}$.
if whileIndex%2 = 0 **then**
  Solve MP2 (i.e., Model (9)) with the local search constraints, i.e., Constraints (7).
else
  while 1 **do**
    Solve MP1 (i.e., Model (8)) under given time limit $\overline{T}$ and obtain the resulting Status.
    if Status is feasible **then**
      Break
    else if Status is infeasible **then**
      $\underline{C} \leftarrow \underline{C} + 1$.
    else if Status is unknown **then**
      Solve MP2 (i.e., Model (9)) under gap tolerance $\mathcal{G}$ and obtain the objective value C'.
      if C' is equal to the previous cycle time found by MP2 **then**
        $\mathcal{G} \leftarrow \mathcal{G} - 0.01$.
      Break

Algorithm 3 The algorithmic outline of Algorithm ICBD

Initialize LB, UB, $LD_{lj}, MS_{ij}, E_i, L_i, FS_i, FT_k$.
Counter $\leftarrow 1$, Dict $\leftarrow \{\}$, whileIndex $\leftarrow 0$.
while Counter $\neq m$ **do**
  Counter $\leftarrow 1$, maxStationTime $\leftarrow 0$, whileIndex $\leftarrow$ whileIndex $+ 1$.
  Solve the master problem by using the two-phase procedure (see Algorithm 2) and obtain x^*, r^*, C^* as well as the gap of the solution gap^* to MP2 (see Model (9)).
  if $gap^* = 0$ **and** $C^* > LB$ **then**
    $LB \leftarrow C^*$.
  for $k \in \mathcal{K}$ **do**
    stationTime $\leftarrow 0, AT_k = \{i \mid \sum_{p \in \mathcal{P}} x^*_{ikp} = 1\}, x^*_k = \{(i,p) \mid x^*_{ikp} = 1\}$.
    if $r^*_k = 0$ **then**
      stationTime $\leftarrow \sum_{i \in AT_k} t_{iP_H}$, **Counter** $\leftarrow$ **Counter** $+ 1$.
    else
      if $x^*_k \in$ **Dict then**
        Read data from the dictionary: stationTime $\leftarrow$ **Dict**$[x^*_k]$.
      else
        Solve SP_k **and obtain the stationTime. If infeasible, stationTime** $\leftarrow -1$.
        if stationTime $= -1$ **then**
          maxStationTime $\leftarrow UB$. **Add feasibility cuts (12).**
        else if $C^* <$ **stationTime** $\leq UB$ **then**
          Add optimality cuts (14).
        else if stationTime $\leq C^*$ **then**
          Counter $\leftarrow$ **Counter** $+ 1$.
        Update information about assignment of tasks and station time to dictionary: **Dict**$[x^*_k] \leftarrow$ **stationTime.**
      maxStationTime $\leftarrow \max($**maxStationTime**, **stationTime**$)$.
  if maxStationTime $< UB$ **then**
    $UB \leftarrow$ **maxStationTime, Recalculate** $MS_{ij}, E_i, L_i, FS_i, FT_k$.
    Add variable-fixing-cuts (15).
Return UB, LB, C^*.

4. Implementation Details of Different Acceleration Strategies

In this section, we provide the details of four acceleration strategies, i.e., the `TriangleSearch`, `CrossSearch`, `InactiveSearch`, and `SubsetPruning` procedures. For each task $i \in AT_k$ that is assigned to station k, we define $\widetilde{PR_i^a}$ and $\widetilde{SU_i^a}$ as its predecessors and successors that are assigned to the same side of task i, respectively, and define $\widehat{PR_i^a}$ and $\widehat{SU_i^a}$ as its predecessors and successors that are assigned to the opposite side of task i, respectively. In particular, for each task i to be investigated, we call it the *active* task and all the other tasks the *inactive* tasks. With this notation, we can now adapt the search procedures by Huang et al. (2022) for finding infeasible subsets of tasks, i.e., SAT_k, for the HRCALBP-II as follows:

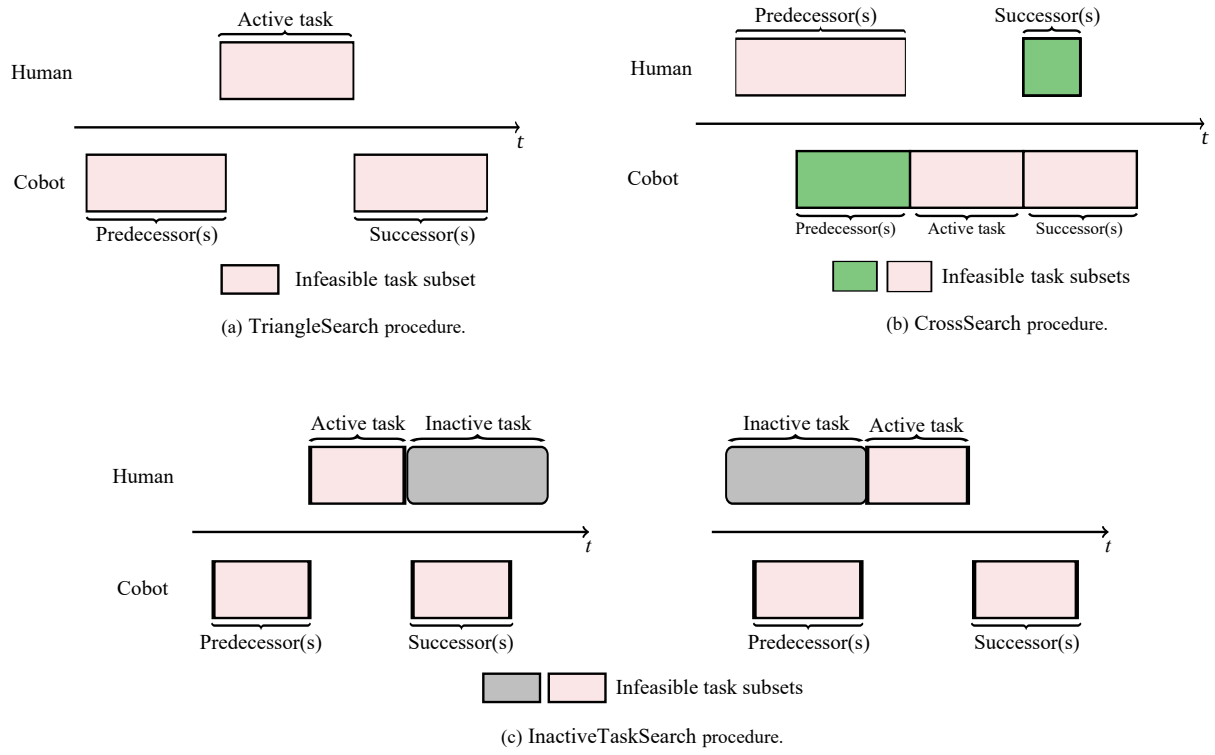

Figure A3 Schematic of the TriangleSearch, CrossSearch and InactiveSearch Procedures.

(i) The `TriangleSearch` Procedure. In this procedure, for any active task i, we define $SAT_k^T = \widehat{PR_i^a} \cup \{i\} \cup \widehat{SU_i^a}$ as the set of tasks that includes task i and its predecessors and successors that are assigned to the opposite side, and $\bar{t}_{\text{sum}}^T = \sum_{i \in SAT_k^T} \sum_{p \in \{p' | x^*_{ikp'}=1\}} t_{ip}$ as the total processing time of tasks in SAT_k^T. Then, we check whether $\bar{t}_{\text{sum}}^T$ is greater than the current upper bound. If so, we conclude that SAT_k^T is an infeasible subset of tasks, and a new feasibility cut can be generated. Figure A3(a) shows the schematic of such procedure.

(ii) The `CrossSearch` Procedure. In this procedure, for any active task i, we define $SAT_k^{C1} = \widehat{PR_i^a} \cup \{i\} \cup \widetilde{SU_i^a}$ as the set of tasks that includes task i, its predecessors in the opposite side and its successors on the same side, and $SAT_k^{C2} = \widetilde{PR_i^a} \cup \{i\} \cup \widehat{SU_i^a}$ as the set of tasks that includes task i, its predecessors on the same side and its successors on the opposite side. Let $\bar{t}_{\text{sum}}^{C1}$ and $\bar{t}_{\text{sum}}^{C2}$ be the total processing time of tasks in SAT_k^{C1} and SAT_k^{C2} respectively. Similar to the `TriangleSearch` procedure, we check whether any of these two values is greater than the current upper bound. If so, we let SAT_k^C be the corresponding infeasible subset of tasks, and generate new feasibility cut(s). Figure A3(b) shows the schematic of such procedure.

(iii) The `InactiveSearch` Procedure. In this procedure, for any active task i, we select an inactive task $j \notin \widetilde{PR_i^a} \cup \widetilde{SU_i^a}$ that is assigned to the same side of task i with maximum processing time. Let $SAT_k^I = \{j\} \cup \widehat{PR_i^a} \cup \{i\} \cup \widehat{SU_i^a}$. Then we check whether the total processing time of tasks in sets $\{j\} \cup \{i\} \cup \widehat{SU_i^a}$ or $\widehat{PR_i^a} \cup \{i\} \cup \{j\}$ is greater than the current upper bound. If so, we conclude that SAT_k^I is an infeasible subset of tasks, and a new feasibility cut can be generated. Figure A3(c) shows the schematic of such procedure.

(iv) The `SubsetPruning` Procedure. Let $\overline{SAT_k} = SAT_k^T \cup SAT_k^C \cup SAT_k^I$. For any infeasible set in $\overline{SAT_k}$, we further investigate whether there exists any subset $\tilde{I}$ that is still infeasible so as to strengthen each cut in $\overline{SAT_k}$. By implementing each of the above three procedures on set $\tilde{I}$, if we can still verify that this set is infeasible, then we will add set $\tilde{I}$ to the final infeasible subset of tasks, i.e., SAT_k.

5. Parameter Tuning

As mentioned in our two-phase procedure for solving the master problem (see Section 4.2.2), there are two important parameters that need to be fixed, i.e., the time limit $\overline{T}$ for MP1 and the tolerance gap $\mathcal{G}$ for MP2. As we know, with different parameters, the corresponding computational performance may be significantly different. Therefore, we select the following candidates for each parameter, i.e., $\overline{T} \in \{0, 5, 10, 15, 20, 25\}$ and $\mathcal{G} \in \{0, 2\%, 4\%, 6\%, 8\%, 10\%\}$, with a total number of $6 \times 6 = 36$ parameter combinations. For each parameter combination of $(\overline{T}, \mathcal{G})$, we randomly select 20 instances from the medium- and large-sized instances, respectively. Figure A4 shows the corresponding computational results for these selected instances, in which the left y axis and right y axis represent the average running time and average gap, respectively. From these results, we can see that when $\overline{T} = 15$ and $\mathcal{G} = 8\%$, Algorithm ICBD achieves the best performance. In the remaining of our experiments, we set the parameters of $\overline{T}$ and $\mathcal{G}$ to the above values respectively.

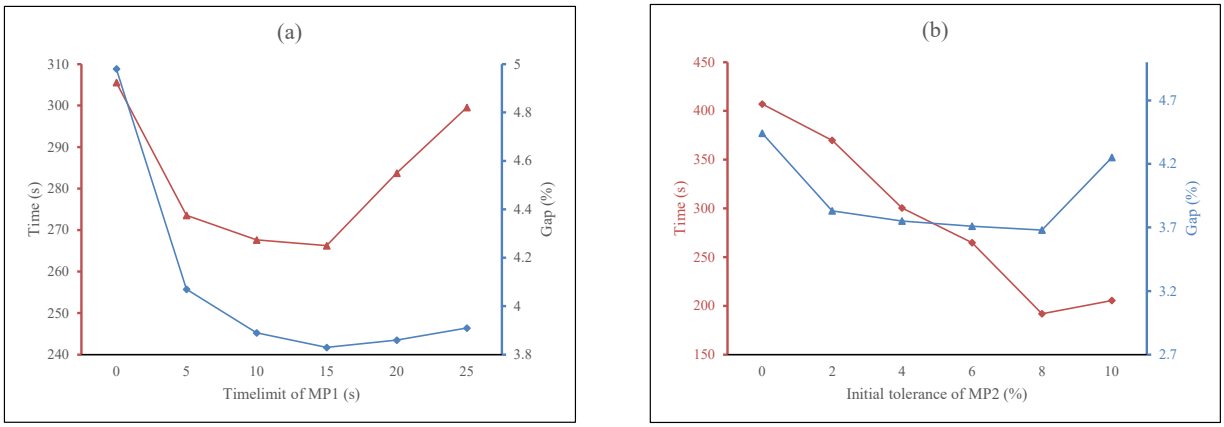

Figure A4 **Average Performance of Algorithm ICBD with Different Values of $\overline{T}$ and $\mathcal{G}$.**

6. Computational Results of the EMIP model, Algorithms ICBD, CBD1 and CBD4

Table A2 Computational Results of the EMIP model, Algorithms ICBD, CBD1 and CBD4.

Class (tasks)	m	q	RF	#	EMIP model					Algorithm ICBD					Algorithm CBD1				Algorithm CBD4			
					OBJ	Gap	Feas	Opt	CPU	OBJ	Gap	Feas	Opt	CPU	OBJ	Feas	Opt	CPU	OBJ	Feas	Opt	CPU
Small (20)	5	1	0.2	50	1181.26	0.93	50	25	908.65	1176.7	0	50	50	7.86	1176.7	50	50	27.88	1176.7	50	50	14.59
	5	1	0.4	50	1167.46	0.77	50	25	934.06	1162.22	0	50	50	17.25	1162.22	50	50	83.05	1162.22	50	50	56.7
	5	2	0.2	50	1115.7	1.64	50	25	904.26	1110.52	0	50	50	13.17	1110.52	50	50	114.55	1110.52	50	50	13.91
	5	2	0.4	50	1086.08	0.98	50	25	947.6	1082.78	0	50	50	33.23	1082.78	50	50	76.42	1082.78	50	50	151.04
	10	2	0.2	50	658.06	1.21	50	39	402.31	657.94	0	50	50	11.33	657.94	50	50	45.27	657.94	50	48	423.96
	10	2	0.4	50	641.38	1.81	50	38	451.64	640.96	0	50	50	39.95	640.96	50	50	19.78	640.96	50	43	1171.64
	10	4	0.2	50	649.22	0.16	50	46	202.49	649.22	0	50	50	7.46	649.22	50	49	173.1	649.22	50	49	197.45
	10	4	0.4	50	620.44	2.19	50	40	371.62	620.34	0	50	50	37.3	620.34	50	50	95.23	620.34	50	41	1391.38
Medium (50)	13	3	0.2	50	1123.12	7.3	50	1	1767.65	1076.14	0.42	50	24	162.27	1097.42	50	24	3868.32	1079.14	50	23	4259.96
	13	3	0.4	50	1118.28	7.04	50	0	1800	1064.66	0.67	50	17	330.84	1085.12	50	25	3635.54	1068.54	50	19	4941.76
	13	5	0.2	50	1067.58	8.03	50	1	1789.94	1029.06	0.65	50	26	264.23	1058.62	50	25	4030.43	1033.08	50	25	3989.55
	13	5	0.4	50	1059.62	8.39	50	0	1800	1009.98	1.52	50	14	609.05	1076.98	50	24	3920.96	1012.92	50	14	5543.77
	25	5	0.2	50	654.84	3.57	50	31	733.51	644.7	0.77	50	40	85.17	650.14	50	35	2698.97	645.12	50	29	2995.15
	25	5	0.4	50	653.28	5.42	50	28	812.69	635.64	1.64	50	36	101.3	641.56	50	35	2502.65	637.06	50	29	2997.53
	25	10	0.2	50	641.34	3.23	50	35	589.87	636.1	0.27	50	42	89.17	641.9	50	40	1612.87	636.6	50	34	2452.53
	25	10	0.4	50	632.5	6.15	50	29	771.03	617.54	1.15	50	39	136.79	623.88	50	34	2512.72	618.22	50	31	2767.06
Large (100)	25	5	0.2	50	1042.96	10.6	50	0	1800	966.16	3.32	50	0	398.5	1019.08	50	0	28799.14	967.1	50	0	28744.9
	25	5	0.4	50	1042	10.51	50	0	1800	968.94	3.55	50	0	662.26	985.56	50	0	28366.96	964.56	50	0	28743.85
	25	10	0.2	50	1038.94	17.61	50	0	1800	919.4	5.05	50	1	851.08	1008.04	50	0	28799.91	908.02	50	1	28512.6
	25	10	0.4	50	1038.28	17.64	50	0	1800	927.42	7.26	50	0	1119.95	938.5	50	1	28265.92	897.7	50	0	28744.83
	50	10	0.2	50	609.94	4.05	50	36	578.58	593.6	2.32	50	38	159.47	616.32	50	35	10453.04	591.52	50	38	6917.75
	50	10	0.4	50	605.92	4.95	50	29	854.67	585.42	2.49	50	38	215.63	596.34	50	34	8462.44	581.2	50	38	6958.22
	50	20	0.2	50	608.22	5.45	50	36	579.52	584.82	1.62	50	38	223.34	628.82	50	33	10654.4	583.02	50	38	6920.98
	50	20	0.4	50	599.96	5.71	50	33	746.12	574.18	3.01	50	38	275.16	590.66	50	38	8031.67	572.96	50	38	6982.13
Sum	/	/	/	1200	/	/	1200	522	/	/	/	1200	791	/	/	1200	782	/	/	1200	738	/
Average	/	/	/	/	860.68	5.64	/	/	1047.76	830.6	1.49	/	/	243.82	848.32	/	/	7385.47	829.06	/	/	7328.89

Note: "Gap" indicates the average optimality gaps. "OBJ" indicates the average cycle time of the HRCALBP. "Feas" and "Opt" indicate the numbers of feasible solutions and optimal solutions, respectively. "CPU" indicates the average computational times (in seconds). The results of "Algorithm CBD1" and "Algorithm CBD4" are directly from Sikora and Weckenborg (2023), in which the authors implement their algorithms on machines with 8 virtual cores of a 2.5GHz Intel Xeon Platinum 8180 processor and 32 GB RAM.

References

Huang D, Mao Z, Fang K, Yuan B (2022) Combinatorial Benders decomposition for mixed-model two-sided assembly line balancing problem. *International Journal of Production Research* 60(8):2598–2624.

Sikora CGS, Weckenborg C (2023) Balancing of assembly lines with collaborative robots: comparing approaches of the Benders' decomposition algorithm. *International Journal of Production Research* 61(15):5117–5133.

Zohali H, Naderi B, Roshanaei V (2022) Solving the type-2 assembly line balancing with setups using logic-based Benders decomposition. *INFORMS Journal on Computing* 34(1):315–332.