Response Letter CYB-E-2024-08-2321

Hyper-Heuristic Optimization Using Multi-Feature Fusion Estimator for PCB Assembly Lines with Linear-Aligned-Heads Surface Mounters

Foreword

Dear Editor-in-Chief, Associate Editor, and Reviewers,

We sincerely thank you for your thorough reading and insightful advice that helped us significantly improve the manuscript's quality. We have carefully considered all received comments for the preparation of the revised version. The major changes to the document are summarized as follows:

1) Model reformulation. Due to the inadequate statement of the model in the original version, we have revised the formulation of its decision variables, added the description of the types of decision variables, and removed redundant constraints and variables, in particular adjusting the constraints on assembly priority of components. The revised model is now easier to understand with more complete constraints.
2) Algorithm description. To make the algorithmic formulation clearer, we further explained the core parts of the proposed algorithms, such as the handling of component assembly priority by low-level heuristic operators, the grouping strategy of components, and the bias assigned to placement points. The key steps of the pseudo-code are annotated in the manuscript, so that readers can better understand algorithm flow.
3) Experimental analysis. For full illustration of the effectiveness of the proposed algorithm, we supplemented the validation conditions for the experiments, analyzed the sources of error in the assembly time estimator, and provided an explanation of the reasons for the long running times in solving efficiency experiments. The reported experiments fully illustrate the superiority of the proposed solution, and at the same time provide directions for further research.

Besides, we have adjusted the relevant inadequacies in the presentation of the whole manuscript. We believe that the revised version will be more conducive to throwing light on the problem and providing reference for other practitioners.

In summary, we made detailed revisions based on all received comments. Point-by-point responses to these comments can be found below. Reviewers' comments are in *italicized red font*, whereas our responses are given in normal black font. Changes in the manuscript are given in blue underlined font, and excerpts from the original manuscript in black underlined font. We sincerely hope this revised version meets the requirements for publication in *IEEE Transactions on Cybernetics*. Thank you very much for taking your valuable time to review our manuscript again.

Sincerely,
The Authors

Content

Responses to Reviewer #1 Comments

1. *There are limited analyses of the coupling between the single machine optimization and entire line optimization. The authors should give further explanation of the two problems, as they are also the motivation for the suggested framework.*

Response: Thank you for your comment. Our initial statement in the Introduction is as follows:

"However, they face difficulty in both the schedule of a single machine and the optimization of the entire line. The efficiency of single-machine scheduling affects the search process for line optimization, which in turn decides assembly tasks for single machines. Solving these two coupled optimization problems poses a significant challenge."

For a clearer explanation of the relationship between the two optimizations and the motivation for the algorithm design, the problem formulation has been further elaborated by adding the following to the first paragraph of Section III.A:

"Component allocation determines the assembly task for surface mounters and inputs into a single-machine optimization, which evaluates the quality of the solution … Improving search efficiency for high-quality solutions is critical to line optimization. A large number of combinations for component allocation makes it difficult to get high-quality solutions, and computing effort increases rapidly as the problem scales up, needing massive resources even for small-scale data. … Considering that long time for single-machine optimization can affect assembly efficiency, a fast assembly time estimator is necessary."

2. *The implementation of the population-generating code is confusing and the dealing on code length also needs to be further explained.*

Response: Thanks you for your comment. The generation of populations and the length of individual patterns are randomized, with each gene corresponding to a pattern, i.e., an LLH operator. All individuals are initialized with random lengths and genes combinations. The random length is just the number of component group. The number of component groups is based on the average number of placement points for each type of component. We divide the placement point of component type i equally into $\hat{\theta}_i$ parts, where

$$\hat{\theta}_i = \max\left(\epsilon \cdot \sum_{i'} \theta_{i'} \cdot \phi_i / \sum_{i'} \phi_{i'}, \theta_i\right) \quad \forall i \in I$$

There are two aspects in dealing with combination patterns of different lengths. When converting the genes to a component allocation result, if the length of individual genes is less than the number of component groups, the genes are cyclically accessed from the beginning to complete the assignment process. Crossover and mutation operations may make the length of individual genes greater than the number of the component group, and then the excess is truncated. The following figure more visually illustrates this process.

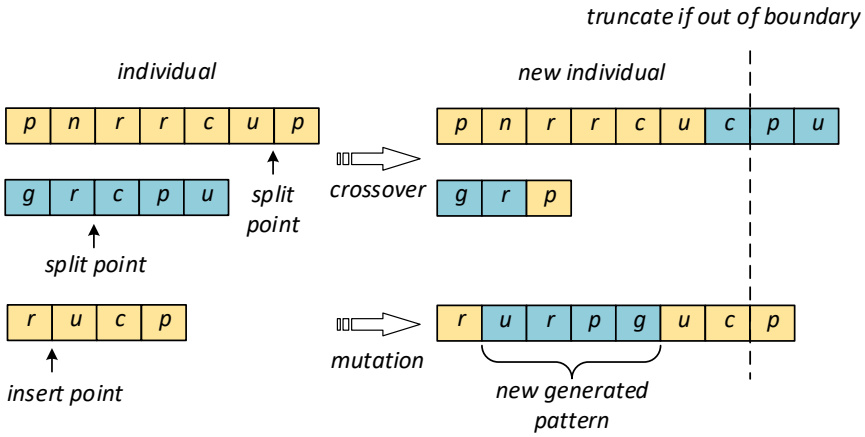

Fig. R1. Crossover and mutation operations

The corresponding revision in Section IV.C is as follows:

"The length of genes is limited to the number of component division groups. ~~Cyclic access to individual patterns during component allocation is applied to handle the case when gene length is less than the limit value.~~ All individuals are initialized with random lengths and pattern combinations. If the length of individual genes is less than the number of component groups, the genes are cyclically accessed from the beginning to complete the assignment process. Each one of two genes selects a split point and performs a crossover operation to exchange gene segments, and the mutation operator inserts randomly generated patterns at a split point. Truncated procedures are applied to individuals whose length exceeds the limit value. "

3. *The mechanism for handing component duplication (also known as component division) is not clearly presented. The authors should think about how to better present the algorithm.*

Response: Thank you for your comment. We apologize for the lack of clarity in our statement. We have reorganized the description of grouping strategies. Component grouping is the smallest unit for making component assignments. Too much grouping leads to less efficient solving, whereas too little grouping may lead to less efficient search. We set a benchmark value for component grouping based on the number of available feeders, and the corresponding revision in Section IV.E is as follows:

"Available feeders are allocated to different machines for the same component type proportionally to the number of points. The specific machine to which each point is assigned needs to be determined. Before executing hyper-heuristic search, the average number of points for each type of component is multiplied by a value that serves as a grouping threshold. Components with points that exceed the threshold are divided into groups.

$$\hat{\theta}_i = \max\left(\epsilon \cdot \sum_{i'} \theta_{i'} \cdot \frac{\phi_i}{\sum_{i'} \phi_{i'}}, \theta_i\right) \quad \forall i \in I \tag{16}$$

where parameter ϵ regulates the number of groups. This grouping strategy balances search efficiency and diversity. The number of available feeders for a component restricts the maximum number of allocated machines, but not as a basis for grouping"

4. A similar problem arises with the role of R_{im} in component grouping.

Response: Thank you for your comment. Bias $\mathcal{R}_{im}$ is applied in the process of placement points assignment to surface mounters. For linear-aligned heads, the different placement heads that undertake the pick-and-place task can lead to deviations between actual moving position and placement point position. We introduce $\mathcal{R}_{im}$ to distinguish the additional deviation values introduced by pick-and-place tasks of different placement heads, to shorten the path of the process, which improves the optimization effect of clustering. For mainstream surface assembly process optimization methods, the head task assignment of the surface assembly optimization does not depend on the specific position of the placement point, which enabled us to design this algorithm.

For more clearly articulating the role of bias $\mathcal{R}_{im}$ in algorithm design, we have made the following revision at the end of Section IV.E:

Component allocation determines the upper limit of the number of placement points of each type assigned to surface mounters in the clustering process.

~~Based on this, the distribution of the duplicated points on heads affects their distance from the center point of the machine.~~ The distribution of points has an impact on assembly efficiency, especially for surface mounters with linear-aligned heads, and the placement position of a point depends on the PAP head. Current research divides surface mounter optimization into head task assignment and pick-and-place sequence, where the former does not depend on the distribution. The head task determines deviations from the center position due to the alignment of the heads. A bias $\mathcal{R}_{im}$ associated with the head task assignment is applied in the clustering process, as follows:

$$\mathcal{R}_{im} = \sum_{k \in K} \sum_{h \in H} \left(\sum_{p \in P} x_p \cdot \lambda_{ip} \cdot u_{ikhm} - h \cdot \rho \right) \quad \forall i \in I, m \in M$$

5. The authors categorize their low-level heuristics as data- or target-driven. What is the distinction between target-driven and estimated sub-objectives (which could be the same thing)?

Response: Thank you for your comment. Target-driven LLHs are designed based on estimated sub-objectives, where the main difference is that LLHs in the allocating stage only compare relative values of sub-objectives between different mounters without estimating specific values, with the aim of speeding up the efficiency of the component allocation process. The estimation method we present in the evaluation of the solution, aims to improve the accuracy of the assembly-time estimator with the help of specific sub-objective values. The corresponding revision in Section IV.B is as follows:

"Target-driven LLHs are related to assembly efficiency, and key sub-objectives are extracted as a basis for component allocation. Instead of specific values, they compare the relative values of the sub-objective between surface mounters, which ~~optimal value of the sub-objectives~~ can be estimated without a specialized optimization procedure."

6. In constraint (4), the sum of component index seems to be redundant.

Response: Thank you for your comment. We apologize for duplicating the component term in Constraint (4), although this does not affect the solution of the model. The revised constraint is

$$e_{skm} \leq \sum_{h \in H} v_{[s+(h-1)\cdot\tau]khm} \leq N \cdot e_{skm} \quad \forall s \in S, k \in K, m \in M \tag{4}$$

where v_{skhm} indicates if head h picks up components from slot s in cycle k of machine m, and e_{skm} indicates if component is picked up when the left-most head aligns to slot s of machine m in cycle k.

7. The meanings of the notation T_1 to T_5 require further explanation.

Response: Thank you for your comment. We apologize for neglecting to formulate this point. We have added it to the explanation of the relevant notations in the modelling section as follows:

$$\min \max_{m \in M} \left(\begin{array}{c} T_1 \cdot \sum_{k \in K} g_{km} + T_2 \cdot \sum_{k \in K \setminus \{|K|\}} \sum_{h \in H} n_{khm} + T_3 \cdot \sum_{k \in K} w_{km} + T_4 \cdot \sum_{s \in S} \sum_{k \in K} e_{skm} + \\ T_5 \cdot \sum_{i \in I} \sum_{k \in K} \sum_{h \in H} u_{ikhm} \end{array} \right) \tag{1}$$

The objective (1) of the model is to minimize the maximum ~~assembly time~~ weighted key assembly metrics among all machines, ~~using key metrics~~ with different weights T_1 for assembly cycle, T_2 for nozzle change, T_3 for pick-up movement, T_4 for pick-up operations, and T_5 for placement operations.

8. In Table IV, ensure that the table's header (e.g., maximum and minimum) is correct.

Response: Thank you for your comment. We are sorry for this error and have fixed it in Table IV as shown below:

TABLE IV
PARAMETERS OF TRAINING AND TESTING DATA

	# of Samples	Outlier %	Mean	Median
Training Sets	2000	11.25	128.67	130.13
	Minimum	Maximum	Std. Dev	
	2.71	232.95	71.67	
Testing Sets	# of Samples	Outlier %	Mean	Median
	400	10.75	126.76	127.11

Minimum	Maximum	Std. Dev
3.80	311.38	72.23

9. The references are cited in an incorrect order in Page 1.

Response: Thank you for your comment. We have adjusted the citation format and numbering of references throughout the manuscript where similar problems were found.

Responses to Reviewer #2 Comments

1. *While the authors claim that it is possible to estimate the optimal value of the sub-objective of the assembly process, it remains unclear how to justify the optimality of the obtained value.*

Response: Thanks for your comment. We apologize for this inaccurate statement. The proposed sub-objectives estimation method is just used for comparison of the relative values of sub-objectives between machines and cannot be solved as an optimum (and also please note that to get an optimum is not necessary in this context). Furthermore, the values of the estimated sub-objectives are intermediate in the allocation process and not the final result of the optimization algorithm. To clarify all this, we have made changes in the manuscript as follows:

"Target-driven LLHs are related to assembly efficiency, and the key sub-objectives are extracted as a basis for component allocation. Instead of specific values, they compare the relative values of the sub-objective between surface mounters, which ~~optimal value of the sub-objectives~~ can be estimated without a specialized optimization procedure."

2. *The authors employ nozzle change indicator in both low-level heuristic design and estimator data preparation. What is the difference between the two?*

Response: Thank you for your comment. In the heuristic iterative search, the algorithm needs to quickly assess the quality of the solution. For the low-level heuristic operator, the metric is the likelihood of a nozzle change occurrence, whereas for the time estimator the metric is the number of nozzle changes. The former is measured as the mean square deviation of the average number of points assigned to the head by the nozzle, whereas the latter is determined using the heuristic algorithm. We consider nozzle change metrics from different perspectives to improve the operational efficiency of the algorithm.

3. *The authors claim that the coding length is dictated by the number of component groups. How does this relate to the data parameters?*

Response: Thank you for your comment. The length of the code is a random value not exceeding the number of component groups, which are related to the number of feeders available and placement points. In the case of a component with multiple feeders, the simplest approach is to divide them equally, which may result in an uneven distribution of points between machines. Increasing the number of component groups while limiting the upper limit of the number of assignable machines can solve the problem of uneven distribution, but it increases the solution space and reduces solving efficiency. Therefore, an algorithm that determines the number of component groupings based on the average number of placement points for each type of component is proposed, and parameter ϵ is used to regulate the number of groups.

$$\hat{\theta}_i = \max\left(\epsilon \cdot \sum_{i'} \theta_{i'} \cdot \phi_i / \sum_{i'} \phi_{i'}, \theta_i\right) \quad \forall i \in I \tag{16}$$

4. How do low-level heuristics handle component priority and machine constraints

Response: Thank you for your comment. Low-level heuristics handle component priority and machine constraints with feasible machine set. All LLHs are based on the set of assignable surface mounters, and the relevant component allocation is also based on this. We have revised the statement in Section IV.B to make the process easier to understand, as follows:

"The number of component feeders and surface mounter specifications restrict the component allocating process. All LLHs are based on the set of assignable surface mounters. ~~LLHs take into account the limitations imposed by the allocation of components of the same type. Priority constraints limit the machines that can be allocated, and the component is replaced with one assigned to fulfill the requirement if no machine is allocatable.~~ The feasible set is adjusted based on the component-assigned mounters. When the number of assigned mounters equals that of available feeders, the indices of assigned mounters are regarded as the new feasible set. Otherwise, the indices of all mounters are regarded as the feasible set. Component prioritization needs to be checked first to see if the loop is closed between constraint relationships and, if so, there is no solution. Otherwise, if during the component allocation process a newly allocated component breaks the priority constraint, the assigned components that do not satisfy the constraint relationship are replaced and re-allocated with the same strategy."

5. In the flow of the algorithm, the authors should annotate the key steps so that the reader is clear about what they mean.

Response: Thank you for your comment. Although we had some annotations in the algorithms, we agree it was not clear for readers to completely understand their implementation. We apologize for such an oversight. In the revised manuscript, annotations have been added to all the key steps of the algorithms that, combined with the statements in the main text, we sincerely hope have increased readability to an acceptable. The corresponding revisions are shown below:

Algorithm 1: Hierarchical Greedy Head Assignment

Input : Nozzle heads γ, component points ϕ'

Output: Number of pick-up operations $\mathcal{O}$

Set a $1 \times |J|$ vector $\mathcal{L}$, a $1 \times |J|$ vector $\mathcal{N}$, and a $1 \times \sum_{i \in I} \phi'_i$ vector $\mathcal{K}$ of all zeros;

Sort $i \in I$ decreasingly with ϕ'_i;

for $i \in I$ **do**

  $j \leftarrow \sum_{j' \in J} \xi_{ij'} \cdot j'$; // assign the nozzle j compatible with component i

  if $\mathcal{N}_j$ Mod $\gamma_{jm} = 0$ **then**

    $\mathcal{L}_j \leftarrow \mathcal{L}_j + 1$; // nozzle allocation is full and start a new cycle

  end

  /* Update the max. number of allocated points and heads */

  Set cycle index $c \leftarrow \mathcal{L}_j$, $\mathcal{K}_c \leftarrow \max(\mathcal{K}_c, \phi'_i)$, $\mathcal{N}_j \leftarrow \mathcal{N}_j + 1$

end

$\mathcal{O} \leftarrow \sum_{c=1}^{c=\sum_{i \in I} \phi'_i} \mathcal{K}_c$

Algorithm 2: Nozzle Change Computation Heuristic

Input : Nozzle heads γ, component points ϕ'
Output: Number of nozzle changes N^*

Set $1 \times |H|$ vector $\mathcal{T}$ of all zeros, $1 \times |H|$ vector $\mathcal{N}$, $V \leftarrow 0$, $V^* \leftarrow \infty$ and $N^* \leftarrow 0$;
while $V \leq V^*$ **do**
  Set $1 \times \gamma_{jm}$ nozzle group $\mathcal{G}_{jm}$ with $\sum_{i \in I} \phi_i \cdot \xi_{ij} / \gamma_{jm}$ points for $j \in J$;
  for $n \in \mathcal{G}_{jm}, j \in J$ **do**
    /* assign nozzle groups to heads */
    $h \leftarrow \arg\min_{h' \in H} \{\mathcal{T}_h\}$, $\mathcal{N}_h \leftarrow j$, $\mathcal{T}_h \leftarrow \mathcal{T}_h + n$
  end
  Set number of cycles $V \leftarrow \max_{h \in H} \mathcal{T}_h$;
  while true do
    /* balance the heads with max. and min. points */
    $h' \leftarrow \arg\max_{h \in H} \mathcal{T}_h$, $h'' \leftarrow \arg\min_{h \in H} \mathcal{T}_h$;
    if $\mathcal{N}_{h'} = \mathcal{N}_{h''}$ **then**
      break;
    end
    /* compare the metrics for cycle and nozzle change */
    $j' \leftarrow \mathcal{N}_{h'}$, $\mathcal{H}_1 \leftarrow \{h \mid \mathcal{N}_h = j', h \in H\}$, $j'' \leftarrow \mathcal{N}_{h''}$, $\mathcal{H}_2 \leftarrow \{h \mid \mathcal{N}_h = j'', h \in H\}$; **if** $T_3 \cdot (\mathcal{T}_{h'} - \mathcal{T}_{h''}) > T_2 \cdot ||\mathcal{H}_2| - |\mathcal{H}_1||$ **then**
      break;
    end
    /* update assignment result */
    $N \leftarrow ||\mathcal{H}_2| - |\mathcal{H}_1||$, $V \leftarrow V - T_3 \cdot (\mathcal{T}_{h'} - \mathcal{T}_{h''}) + T_2 \cdot N$, $\mathcal{T}' \leftarrow \mathcal{T}$;
    for $h \in \mathcal{H}_1 \cup \mathcal{H}_2$ **do**
      $\mathcal{T}_h \leftarrow \sum_{h' \in \mathcal{H}_1 \cup \mathcal{H}_2} \mathcal{T}'_{h'} / (|\mathcal{H}_1| + |\mathcal{H}_2|)$, $\mathcal{N}_h \leftarrow j'$;
    end
  end
  if $V < V^*$ **then**
    /* add nozzle groups and re-allocate */
    $V^* \leftarrow V$, $N^* \leftarrow N$, $\gamma_{j'm} \leftarrow \gamma_{j'm} + 1$;
  end
end

Algorithm 3: Aggregated Clustering Algorithm for Duplicated Component Points

Input : Available feeder θ_i, component points set P, ~~machine-assigned points U~~, points position (x_p, y_p), machine-component assignment u_{ikhm} and r_{im}

Output: Machine-allocated points $\overline{\mathcal{P}}$

Set machine-assigned sets $\mathcal{P}_m \leftarrow \varnothing$ and number of machine-assigned points $\mathcal{U}_{im} \leftarrow 0, i \in I, m \in M$;

for $m \in M$ **do**

  for $i \in \{i' \mid U_{i'm} r_{i'm} > 0, \theta_{i'} = 1, i' \in I\}$ **do**

    $\mathcal{U}_{im} \leftarrow |P_i|$, $\mathcal{P}_m \leftarrow \mathcal{P}_m \cup P_i$;

  end

  Set center points $\mathcal{X}_m \leftarrow \sum_{p \in \mathcal{P}_m} x_p / |\mathcal{P}_m|$, $\mathcal{Y}_m \leftarrow \sum_{p \in \mathcal{P}_m} y_p / |\mathcal{P}_m|$ of each machine ;

end

while true do

  $\overline{\mathcal{X}} \leftarrow \mathcal{X}$, $\overline{\mathcal{Y}} \leftarrow \mathcal{Y}$, $\overline{\mathcal{U}} \leftarrow \mathcal{U}$, $\overline{\mathcal{P}} \leftarrow \mathcal{P}$;

  for $p \in \{p' \mid p' \in P_i, \theta_i > 1, i \in I\}$ **do**

    $m \leftarrow \arg\min_{m' \in M} \{(\mathcal{X}_{m'} - x_p + \mathcal{R}_{im'})^2 + (\mathcal{Y}_{m'} - y_p)^2 \mid \overline{\mathcal{U}}_{im'} < \sum_{k \in K} \sum_{h \in H} u_{ikhm'} U_{im'}\}$ as the allocated machine, $\overline{\mathcal{P}}_m \leftarrow \overline{\mathcal{P}}_m \cup \{p\}$;

    $\overline{\mathcal{U}}_{im} \leftarrow \overline{\mathcal{U}}_{im} + 1$;

    /* update num. of assigned points and center of mounters */

    $\mathcal{X}_m \leftarrow \mathcal{X}_m + (x_p - \mathcal{X}_m - \mathcal{R}_{im}) / |\overline{\mathcal{P}}_m|$, $\mathcal{Y}_m \leftarrow \mathcal{Y}_m + (y_p - \mathcal{Y}_m) / |\overline{\mathcal{P}}_m|$;

  end

  if $\overline{\mathcal{X}} = \mathcal{X}$ **and** $\overline{\mathcal{Y}} = \mathcal{Y}$ **then**

    break;

  end

end

6. *page 1, in abstract, it states: "Printed circuit board assembly line scheduling (PCBALS) is critical to production, which is a major difficulty". That phrase is inappropriate because scheduling is a means, not a difficulty*

Response: Thank you for your comment. We apologize for the misrepresentation due to the order of the phrases and have adjusted this as:

"Printed circuit board assembly line scheduling (PCBALS) ~~is critical to production efficiency, which~~ is a ~~major difficulty~~ difficult task in the electronic industry for assembly lines using surface mounters~~, which is critical to production efficiency.~~ "

7. *page 2, second column, the phase "modeling is difficult to implement effectively ". The authors need interpret the exact meaning of "implement effectively".*

Response: Thank you for your comment. We are sorry that such a statement in the manuscript is inaccurate. We wanted to discuss the practical limitations of mathematical programming, in spite of its theoretical ability to achieve optimal solutions. First, because of the difficulty for modeling the

actual problem. For some problems, it is difficult to express optimization objectives and constraints in mathematical form. Also, complex mathematical forms or non-linear term can make models unsolvable. Second, solving some models requires a huge amount of computational resources, and the time for model solving dramatically increases with the scale of the problem. The text in Section II has been revised as follows:

"Although mathematical modeling can solve problems optimally, ~~yet~~ it is difficult to obtain mathematical expressions for some real-world applications and, even when this is possible, their implementation may require unacceptably high computational complexity. ~~complex and difficult to implement effectively.~~ "

8. page 3, second column, the type (or range of values) of decision variables in the model needs to be clearly stated.

Response: Thank you for your comment. We apologize for the error. The revised Table I is shown below (please note that only changed parts are listed here).

TABLE I

NOTATIONS OF THE MATHEMATICAL MODEL

Notation	Description
Indices & Sets	
ξ_{ij}	$= 1$, iff. component type i is compatible with nozzle type j $(= 0$, otherwise$)$
η_{im}	$= 1$, iff. component type i is compatible with machine m $(= 0$, otherwise$)$
Decision Variables	
g_{km}	Binary variable, $= 1$, iff. any point is assembled in cycle k of machine m
u_{ikhm}	Binary variable, $= 1$, iff. component type i is assigned to head h in cycle k of machine m
v_{skhm}	Binary variable, $= 1$, iff. head h picks up components from slot s in cycle k of machine m
f_{ism}	Binary variable, $= 1$, iff. component i is assigned to slot s of machine m
e_{skm}	Binary variable, $= 1$, iff. component is picked up when the left-most head align to slot s of machine m in cycle k
n_{khm}	Binary variable, $= 1$, iff. head h of machine m changes nozzle between cycles k and $k+1$
r_{im}	Binary variable, $= 1$, iff. component type i is assembled by machine m
w_{km}	Integer variable, which indicates slots crossed by heads during pickup in cycle k of machine m

9. page 6, second column, the input of the algorithm point position is unclear.

Response: Thank you for your comment. We follow the notation of the model in the description of the algorithm to avoid confusions due to excessive use of notation, with the difference that we treat them as known variables. The input of algorithm 3 is revised as "Available feeder θ_i, placement points set P, ~~machine-assigned points U,~~ points position (x_p, y_p), machine-component assignment u_{ikhm}

and r_{im}", where the notations θ_i, P, u_{ikhm} and r_{im} have been given in Table I, and (x_p, y_p) are the coordinates of the placement point p on the PCB.

Responses to Reviewer #3 Comments

1. *The authors point out that the estimator's results may be affected by the point distribution, but how might this issue be resolved?*

Response: Thank you for your comment. Since the distributional characteristics of the placement points cannot be directly encoded, they inevitably have an impact on the accuracy of the estimator. To solve this issue, we generate data with random distribution characteristics right at the time of model training, to reduce the effect of distribution characteristics on it. The distribution characteristics of the placement points determine the assembly path of the gantry during the assembly process, which accounts for a relatively small percentage of the whole assembly process. Experimental results show that our proposed method has an average error of 3.43% and a maximum error of 16.57% on the testing set, which are significantly better than the state of the art. For the component allocation process, we also apply multi-population search strategies and determine the final solution by providing multiple candidate solutions combined with specific results from time estimation and single-machine optimization for more accurate results.

2. *The motivation of bias for aggregated clustering algorithm should be given.*

Response: Thank you for your comment. Bias $\mathcal{R}_{im}$ is applied in the process of placement points assignment to surface mounters. For linear-aligned heads, the different placement heads that undertake the pick-and-place task can lead to deviations between actual moving position and placement point position. We introduce $\mathcal{R}_{im}$ to distinguish the additional deviation values introduced by pick-and-place tasks of different placement heads, to shorten the path of the process, which improves the optimization effect of clustering. For mainstream surface assembly process optimization methods, the head task assignment of the surface assembly optimization does not depend on the specific position of the placement point, which enabled us to design this algorithm.

For more clearly articulating the role of bias $\mathcal{R}_{im}$ in algorithm design, we have made the following revision at the end of Section IV.E:

Component allocation determines the upper limit of the number of placement points of each type assigned to surface mounters in the clustering process.

~~Based on this, the distribution of the duplicated points on heads affects their distance from the center point of the machine.~~ The distribution of points has an impact on assembly efficiency, especially for surface mounters with linear-aligned heads, and the placement position of a point depends on the PAP head. Current research divides surface mounter optimization into head task assignment and pick-and-place sequence, where the former does not depend on the distribution. The head task determines deviations from the center position due to the alignment of the heads. A bias $\mathcal{R}_{im}$ associated with the head task assignment is applied in the clustering process, as follows:

$$\mathcal{R}_{im} = \sum_{k\in K}\sum_{h\in H}\left(\sum_{p\in P} x_p \cdot \lambda_{ip} \cdot u_{ikhm} - h \cdot \rho\right) \quad \forall i \in I, m \in M$$

3. *In the model building, the authors introduced a symbol m_q to indicate the machine index of the first/last component to be mounted under the condition of priority constraints. However, in practice, we typically only know the priority relationship (that is which components must be assembled before/after another), how do the authors handle this condition?*

Response: Thank you for your comment. m_q is an integer variable, which indicates the last (first) machine to assemble component type i (i'), $q = (i, i')$. However, we noticed that this variable is redundant and removed it. Constraints (10), (12), and (13) associated with priority constraint are rewritten as follows:

$$r_{im} \le \sum_{k\in K}\sum_{h\in H} x_{ikhm} \le N \cdot r_{im} \quad \forall i \in I, m \in M \tag{10}$$

$$m - N \cdot (1 - r_{im}) \le m' + N \cdot (1 - r_{i'm'}) \quad \forall q = (i, i') \in Q, m \in M, m' \in M \tag{12}$$

$$\begin{aligned} &\max_{k\in K, h\in H}\{k \cdot x_{ikhm}\} + N \cdot (r_{im} + r_{i'm} - 2) \le \\ &\min_{k\in K, h\in H}\{k \cdot x_{i'khm} + N \cdot (1 - x_{i'khm})\} \ \forall q = (i, i') \in Q, m \in M \end{aligned} \tag{13}$$

Constraint (10) indicates the relationship between single-machine surface optimization and assembly line component allocation. Constraints (12) and (13) correspond to priority constraints on different machine assignments and on the same machine, respectively.

4. *In algorithm 1, line 4, the component type is incorrectly assigned to the component index.*

Response: Thank you for your comment. We apologize for this clerical error. We have fixed the problem and added notes on key aspects of the algorithm to ensure that the algorithm is accurately described.

$$j \leftarrow \sum_{j'\in J} \xi_{ij'} \cdot j'$$

In addition, we have explained the key steps of the algorithm to ensure that it is easier for the reader to understand it.

5. *In algorithm 3, the authors explain how each cluster's starting positions are determined, but they do not consider the scenario in which all components assigned to the machine have more than one feeder.*

Response: Thank you for your comment. We apologize for the unclear statement. For aggregated placement points assignment methods, the choice of initial points does not have a significant impact

on the results. The center of all the placement points can be selected as its initial point to speed up the computation process. The center point of each mounter will be automatically adjusted by the influence of the assigned position of the placement point for each component type. The related description is revised as follows:

"Algorithm 3 provides an aggregative clustering heuristic. The components with a single feeder have all of their points allocated to one machine, resulting in the center points of each machine. For machines without assigned components, we take the center of all points as their center point. "

6. *The randomly generated data is chosen for the network training data, does this affect the accuracy of the network and, consequently, the final outcome?*

Response: Thank you for your comment. Randomly generated data actually improves the accuracy of the estimator. As mentioned earlier, distributional characteristics have an effect on estimation accuracy, and randomized data have a relatively uniform distribution that improves fitting accuracy. In practical network training, it is difficult to obtain a large amount of data from the production line. Less data makes the network not fully exploit its characteristics. Randomly generated data solves this problem.

7. *When comparing the computational efficiency of the algorithms, the hyper-heuristic performs worse than the genetic algorithm, with one PCB 2-5 taking the longest time to run, please provide an explanation for this result.*

Response: Thank you for your comment. We explain in the manuscript the reason why the hyper-heuristic algorithm takes longer: "The genetic algorithm consists of relatively basic operators, which allow it to search quickly at the cost of solution quality. The hyper-heuristic and hybrid algorithms use a more complex time-fitting approach and account for component duplication, resulting in longer times than that of the genetic algorithm … Evaluating the quality of the candidate solutions takes a large part of the solving time of the hyper-heuristic. "

Although the hyper-heuristic algorithm is not the shortest in terms of time spent, the improvement in assembly efficiency it brings is worthwhile. Both PCB2-5 and PCB2-10 have four types of nozzles, and their longer optimization time in production line $L1$ (consisting of two surface mounters) is due to the longer time spent by the single-machine optimization algorithm in dealing with a larger number of nozzle types. We added the following explanation:

"PCB2-5 and PCB2-10 are more complex. Single-machine optimization takes longer for PCBs with larger number of components and nozzle types, resulting in relatively poor solving efficiency. "

Responses to Reviewer #4 Comments

1. *The primary distinction from previous research is that this paper focuses on optimizing the PCB assembly line for surface mounters with linear-aligned-heads. However, there is less description of the operating characteristics of this mounter (compared to other types of surface mounters). The literature review focuses more on rotary-head type.*

Response: Thank you for your comment. We have researched all the PCB assembly line studies and found that the majority of them centered around rotary-head surface mounters, with little to no research on inline models. Although the two types of surface mounters in the assembly process are basically the same, the special structure of the linear-aligned heads have a great impact on assembly efficiency. Please note that neither in component allocation nor in time estimation have the related studies made special treatment for the simultaneous pickup process. Nonetheless, the research on rotary-heads surface mounters production line is still inspiring for us. We added the following sentence at the end of the literature review section:

"To summarize, the present research focuses more on rotary-heads surface mounter line optimization, which inspires us to further optimize a line consisting of surface mounters with special linear-aligned heads structure in terms of search capabilities and time estimation accuracy. "

2. *The proposed mathematical model contains non-linear terms, like Constraint (9), which leads it unsolvable, and the authors do not give a way to deal with it.*

Response: Thank you for your comment. The proposed model is an integer program model where constraints (9) and constraints (13) are nonlinear terms. Direct linearization of these constraints is difficult and does not guarantee the solvability of the model. In the experimental section, to compare the proposed method with the approximated optimal solution of the model, we assume that the available nozzles are sufficient, which makes Constraint (9) hold constant, while disregarding priority constraints (12) - (13). We added the following statement in the revised version:

"… which is built by extracting key metrics that affect assembly efficiency. To make the model linear and solvable, we assume enough nozzles are available and, in addition, placement priority constraints are ignored."

3. *The reviewer is unable to comprehend the relationship between Constraints (12)-(13) and the priority constraints.*

Response: Thank you for your comment. First, we have removed redundant decision variable $\widetilde{m}_q$. Constraints (12) and (13) correspond to priority constraints on different machine assignments and on the same machine, respectively, where r_{im} indicates if component type i is assigned to machine m.

For example, given priority constraint $q = (i, i') \in Q$ (component type i is assembled before component type i'), if component i is assigned to machine m, then for constraint (12) to hold $r_{i'm'} = 0$ must be satisfied when $m' < m$, whereas $r_{i'm'}$ can take any value (0 or 1) when $m' \geq m$. Similarly, if $r_{i'm'} = 1$, there must be $r_{im} = 0$ when $m < m'$, whereas r_{im} can take any value for $m > m'$.

$$m - N \cdot (1 - r_{im}) \leq m' + N \cdot (1 - r_{i'm'}) \quad \forall q = (i, i') \in Q, m \in M, m' \in M \tag{12}$$

For the condition that components i and i' are assigned to the same machine m, Constraint (13) limits the assembly cycle of component i to be less than or equal to the assembly cycle of component i'. For the case where component i and component i' are assigned different machines, Constraint (13) holds constant.

$$\begin{aligned} &\max_{k \in K, h \in H}\{k \cdot x_{ikhm}\} + N \cdot (r_{im} + r_{i'm} - 2) \leq \\ &\min_{k \in K, h \in H}\{k \cdot x_{i'khm} + N \cdot (1 - x_{i'khm})\} \quad \forall q = (i, i') \in Q, m \in M \end{aligned} \tag{13}$$

4. *Similar problems also exist in algorithm design, where the authors explain few and need to be supplemented.*

Response: Thank you for your comment. In the algorithm implementation, the handling of component assignment priorities is mainly reflected in the design of the low-level operators. We have revised the statement in Section IV.B as follows:

"The number of component feeders and surface mounter specifications restrict the component allocating process. All LLHs are based on the set of assignable surface mounters. ~~LLHs take into account the limitations imposed by the allocation of components of the same type. Priority constraints limit the machines that can be allocated, and the component is replaced with one assigned to fulfill the requirement if no machine is allocatable.~~ The feasible set is adjusted based on the component-assigned mounters. When the number of assigned mounters equals that of available feeders, the indices of assigned mounters are regarded as the new feasible set. Otherwise, the indices of all mounters are regarded as the feasible set. Component prioritization needs to be checked first to see if the loop is closed between constraint relationships and, if so, there is no solution. Otherwise, if during the component allocation process a newly allocated component breaks the priority constraint, the assigned components that do not satisfy the constraint relationship are replaced and re-allocated with the same strategy."

5. *In terms of the hyper-heuristic optimization algorithm, the authors use multiple populations to perform the search simultaneously. What is the motivation for doing this (instead of using just one population)?*

Response: Thank you for your comment. The sequence in which components are assigned can have an impact on the load balancing results. We choose multiple populations, with different populations representing one component allocation sequence, to avoid a single sequence becoming a bottleneck limiting the efficiency of the assembly line. Simultaneous searching of multiple populations increases the diversity of results and improves the quality of the solution. In addition, multiple populations can provide candidate solutions with similar estimated assembly times, and the estimation error can be overcome by further executing specific optimization algorithms for the candidate solutions. We have added the following clarification in the revised version:

"Multiple populations with varying component allocation sequences iterate separately to avoid allocation order limiting efficiency gains while providing multiple high-quality solutions for further evaluation."

6. And what are the main advantages of the time estimator over existing estimators?

Response: Thank you for your comment. Most of the existing studies focus on linear fitting, which has relatively poor fitting accuracy. Neural networks have advantages in both fitting effectiveness and efficiency. However, in related studies, even if there exist several non-linear fitting methods using neural networks, they have problems such as insufficient feature selection. The effectiveness of the time estimator has been fully evaluated in the experimental section. From the experimental results, the fitting accuracy of the neural network is significantly better than the linear fitting methods. Even though it is the same neural network, the proposed coding method has higher accuracy.

The manuscript discusses the advantages of the proposed time estimator as follows:

"The complexity of the PCB assembly process makes some properties difficult to uncover, therefore, we propose a heuristic algorithm that estimates performance metrics to improve fitting accuracy. … The mean absolute error and maximal absolute error of training and testing data are listed in Table VI. The performance of the fitting method on the testing set is the basis for evaluating the accuracy of the estimator. It can be seen that the neural network method is more advantageous in time estimation. The proposed estimator encoding method reduces the average absolute error on the testing set from 5.09% to 2.01%, in contrast to the encoding method that simply feeds basic parameters. "

7. How does component duplication relate to the number of available feeders?

Response: Thank you for your comment. The number of feeders available corresponds to the type of component. Multiple feeders can be available for the same type of component to increase productivity by assigning them to different machines, which is referred to as component duplication, as discussed below:

"Available feeders are allocated to different machines for the same component type proportionally to the number of points. The specific machine to which each point is assigned needs to be determined.

Before executing hyper-heuristic search, the average number of points for each type of component is multiplied by a value that serves as a grouping threshold. Components with points that exceed the threshold are divided into groups.

$$\hat{\theta}_i = \max\left(\epsilon \cdot \sum_{i'} \theta_{i'} \cdot \frac{\phi_i}{\sum_{i'} \phi_{i'}}, \theta_i\right) \quad \forall i \in I \tag{16}$$

where parameter ϵ regulates the number of groups. This grouping strategy balances search efficiency and diversity. The number of available feeders for a component restricts the maximum number of allocated machines, but not as a basis for grouping"

8. *In experimental design, are different single machine optimization methods compared to demonstrating the algorithm's generalization?*

Response: Thank you for your comment. The suggestion for the algorithm's generalization as a key metric for algorithmic design is reasonable. We strongly agree with it and would like to make some points. We think that the generalization of the entire-line optimization algorithm is demonstrated by the fact that it shows better results for different data. Our proposed algorithm consists of two parts - time estimator and component allocation. For the time estimator, the surface mount optimization algorithm should guarantee Pareto or near-Pareto optimality for multiple sub-objectives to ensure the accuracy of the time estimation. Inaccurate fitting can make it difficult for component allocation to balance the workload of surface mounters. Productivity improvement of the surface mount optimization can ensure the improvement of the production line efficiency. Therefore, we choose the state-of-the-art single-machine optimization algorithm. Other methods perform poorly in both assembly time fitting and single-machine optimization, and the low production efficiency of the whole line is not meaningful for the actual production.

9. *Add quantitative measure of the comparison of the algorithm and the model, as well as time estimation accuracy in the abstract.*

Response: Thank you for your comment. We have modified the last part of the abstract as follows:

"Experimental results show that (1) the gaps between the solution from HHO-MFFE and the optimal solution of the model are 3.44%~7.28% for small-scale data; (2) the proposed time estimator has higher accuracy than regression and heuristic-based ones, with mean absolute error of 2.01% ~3.43% for training and testing data, respectively; and (3) HHO-MFFE is better than other state-of-the-art algorithms, with average improvement of 4.53%~12.18%."

10. *The title of Section IV is suggested to be consistent with the name of the algorithm described in the flowchart.*

Response: Thanks for your comment. The title of Section IV has been expanded to HYPER-HEURISTIC OPTIMIZATION WITH AN NN-BASED MULTI-FEATURE FUSION ESTIMATOR.

11. The range of values of the constraint variables in the model needs to be pointed out.

Response: Thank you for your comment. We apologize for our incompleteness in the model description. The type of all decision variables (integer or binary) is now described in Table I.

12. When citing, utilize hyphenation for consecutively numbered references and order the citation by number.

Response: Thank you for your comment. We have carefully revised the way references are cited in the text.

13. Notations should be checked at every place in the paper.

Response: Thank you for your comment. We have carefully checked all notations in the manuscript and ensured all of them have suitable explanations.