Albus, Marcel; Hornek, Timothée; Kraus, Werner; Huber, Marco F.



# Towards scalability for resource reconfiguration in robotic assembly line balancing problems using a modified genetic algorithm

Journal of Intelligent Manufacturing

**Provided in Cooperation with:**
Springer Nature

*Suggested Citation:* Albus, Marcel; Hornek, Timothée; Kraus, Werner; Huber, Marco F. (2024) : Towards scalability for resource reconfiguration in robotic assembly line balancing problems using a modified genetic algorithm, Journal of Intelligent Manufacturing, ISSN 1572-8145, Springer US, New York, NY, Vol. 36, Iss. 2, pp. 1175-1199, https://doi.org/10.1007/s10845-023-02292-0

This Version is available at:
https://hdl.handle.net/10419/318658

Journal of Intelligent Manufacturing (2025) 36:1175–1199
https://doi.org/10.1007/s10845-023-02292-0

# Towards scalability for resource reconfiguration in robotic assembly line balancing problems using a modified genetic algorithm

**Marcel Albus**[1] · **Timothée Hornek**[1] · **Werner Kraus**[1] · **Marco F. Huber**[1,2]

Received: 15 June 2023 / Accepted: 24 November 2023 / Published online: 12 January 2024


**Abstract**
Assembly lines are still one of the most used manufacturing systems in modern-day production. Most research affects the building of new lines and, less frequently, the reconfiguration of existing lines. However, the first is insufficient to meet the reconfigurable production paradigm required by volatile market demands. Consequent reconfiguration of resources by production requests affects companies' competitiveness. This paper introduces a problem-specific genetic algorithm for optimizing the reconfiguration of a Robotic Assembly Line Balancing Problem with Task Types, including additional company constraints. First, we present the greenfield and brownfield optimization objectives, then a mathematical problem formulation and the composition of the genetic algorithm. We evaluate our model against an Integer Programming baseline on a reconfiguration dataset with multiple equipment alternatives. The results demonstrate the capabilities of the genetic algorithm for the greenfield case and showcase the possibilities in the brownfield case. With a scalability improvement through computation time decrease of up to ~2.75×, reduced number of equipment and workstations, but worse objective values, the genetic algorithm holds the potential for reconfiguring assembly lines. However, the genetic algorithm has to be further optimized for the reconfiguration to leverage its full potential.

**Keywords** Reconfiguration · Assembly line balancing · Genetic algorithm · Production

**Mathematics Subject Classification** 91B38 · 90C10 · 91B32

## Introduction

Assembly lines play a pivotal role in the modern-day manufacturing of nearly all products of daily life. All assembly processes must be executed efficiently while maintaining high-quality products to accomplish a final product. Most often, this efficiency is achieved through the use of assembly lines. An assembly line is a flow-oriented production system, often used in the mass production of standardized products with different aims, such as cost and production times reduction (Boysen et al., 2008). This production system consists of multiple workstations along a paced line, producing finished products in a fixed amount of time. In each workstation, assembly operations are performed, which are divided into a set of elementary operations named *tasks* (Scholl et al., 2010). To perform one task, a *task time* is required until the processing is finished (Scholl et al., 2010). Each workstation along the line performs one or more tasks to achieve the final product. A *cycle time* constraint leads to a fixed production rate (Scholl & Becker, 2006). Subsequent tasks are performed based upon an assembly graph (or assembly order) containing all steps to achieve the final product (Boysen et al., 2008). For decades, efforts have been undertaken to improve the efficiency and cost-effectiveness of assembly lines due to their economic importance as production systems (Rubinovitz et al., 1993; Baybars, 1986; Li et al., 2020). Many types of assembly lines are investigated, such as fully automated, hybrid, and manual operation (Becker & Scholl, 2006). While manual lines are characterized by high variable costs and low investment costs, automated assembly

✉ Marcel Albus
marcel.albus@ipa.fraunhofer.de

[1] Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Nobelstraße 12, Stuttgart 70569, Germany

[2] Institute of Industrial Manufacturing and Management IFF, University of Stuttgart, Stuttgart 70569, Germany

lines necessitate high investment costs in dedicated equipment while having lower variable costs.

More recently, assembly lines have produced low volumes of highly customized products, shifting from mass production to mass personalization (Piller, 2001; Koren, 2010). More than static resource allocation is needed for these dynamic production processes due to the reconfigurable production paradigm required by volatile market demands (Koren & Shpitalni, 2010). Increasing demand variety requires more flexible and efficient production systems (Çil et al., 2017). This need is tackled by increased exploitation of resources at hand, a necessity to keep costs competitive under varying production conditions. While static, highly automated production systems lack the required flexibility, industrial robots and flexible equipment can aid in tackling these challenges. They are used to leveraging cost efficiency while maintaining high quality and flexibility to counteract the changing market conditions (Nilakantan & Ponnambalam, 2016). Nevertheless, manufacturing companies struggle to stay competitive through increased competition, variety in demand, and additional constraints through customization (Wiendahl et al., 2004; Bukchin et al., 2002). Modern-day productions shift from creating new production lines to the *reconfiguration* of existing lines to address these issues and to save investment costs (Oliveira et al., 2012; Falkenauer, 2005). However, the research direction of reconfiguration and rebalancing assembly lines needs to be studied more in literature (Makssoud et al., 2015; Yelles-Chaouche et al., 2020). As a result, our paper addresses this field of research with the following main contributions:

(i) A reconfiguration model for the Robotic Assembly Line Balancing Problem with Task Types (RALBP-TT) was adapted to a Genetic Algorithm (GA).
(ii) A problem-specific GA structure was designed to solve the adapted model and benchmarked on a reconfiguration dataset against an Integer Programming (IP) baseline.
(iii) Scalability improvements of up to ~2.75× through computation time decrease were achieved on the IP baseline.

## Related work

### Assembly line balancing

The demand for flexible and automated equipment leads to increased market availability of production resources. However, the flexibility of equipment comes with a drawback: multiple equipment alternatives are available for each task, known as the *equipment selection problem* (Bukchin & Rubinovitz, 2003; Oesterle et al., 2019) As a result, manual planning becomes infeasible due to the intractably many possible combinations of conceptual assembly line designs (Falkenauer, 2005). The increased combinatorial possibilities are often counteracted by focusing on the Simple Assembly Line Balancing Problem (SALBP), which uses the following assumptions (Becker & Scholl, 2006; Baybars, 1986; Sewell & Jacobson, 2012; Battaïa & Dolgui, 2013; Boysen et al., 2008):

(i) One homogeneous product is produced.
(ii) Paced line with a fixed cycle time.
(iii) Deterministic task times.
(iv) No assignment restrictions other than precedence constraints.
(v) Serial layout with one-sided stations.
(vi) All stations are equally equipped concerning production capabilities.
(vii) Maximize the line efficiency.

The goal of the SALBP is often to maximize line efficiency or to minimize the number of equipment along the line (Sewell & Jacobson, 2012). This objective function is unsuitable when different equipment types have different costs and capabilities (Nicosia et al., 2002). In this case, less equipment do not automatically result in optimized production cost or efficiency. Moreover, in many manufacturing companies, these assumptions do not hold. Additional company requirements, specialized production equipment with designated characteristics, or different optimization goals contradict the assumptions.

### Robotic assembly line balancing

The demand of modern production facilities for more efficient systems requires equipment to be specialized production resources with designated characteristics, such as robots, among other things. Robot usage improves productivity and product quality and allows additional flexibility in the manufacturing process (Gao et al., 2009). However, not all specialized types of equipment have the same production capabilities and efficiency at different tasks, resulting in varying processing times. Some types of equipment are better suited for one set of tasks, while others are inappropriate or cannot perform the tasks because of production limitations (Bukchin & Tzur, 2000). The use of flexible equipment (e.g.robots) in assembly line balancing allows the inclusion of capabilities, additional industrial requirements, and equipment-dependent variable task times in the problem formulation to meet modern production requirements (Falkenauer, 2005; Javaid et al., 2021). This usage of automated production equipment in assembly line balancing is modelled and known as the Robotic Assembly Line Balancing Problem (RALBP) (Rubinovitz et al., 1993). However, this

branch of the Assembly Line Balancing Problem (ALBP) is not limited to using robots but models specific characteristics and assumptions of flexible production equipment, observed, for example, in robotic environments. With the enhancement of ALBPs with these characteristics, two subproblems arise: (1) a task to equipment and workstation assignment, and (2) an equipment selection problem (Boysen et al., 2022; Çil et al., 2017). The following assumptions are stated for the RALBP (Chutima, 2022; Levitin et al., 2006; Yoosefelahi et al., 2012):

(i) The line is balanced for a single product.
(ii) An assembly task is an elementary operation. As such, it cannot be further subdivided.
(iii) A precedence graph represents the precedence relations between tasks. The precedence relations are known and fixed.
(iv) Task times are deterministic floating-point values, and their values depend on the selected equipment to perform the task.
(v) A workstation can be used to execute any task if the robot assigned to the workstation can complete it.
(vi) A single robot is allocated to each workstation.
(vii) All machines are always available without capacity limitation or breakdown.
(viii) Material handling, transportation, loading and unloading times are negligible or are included in task times.
(ix) Equipment setup times are considered.
(x) The purchase cost of robots is considered.

Due to a demand for increased efficiency, production and quality limitations must be reduced as much as possible (Psarommatis et al., 2020). Production limitations can occur due to specialized equipment capable only of performing one set of task types, such as joining, handling, controlling, adjusting, or separation (VDI Richtlinie, 1990; Lotter & Wiendahl, 2012). These specialized pieces of equipment might lead to production bottlenecks when used outside their capabilities. At worst, a piece of equipment is so specialized as to perform only one task type and is infeasible for all others (Albus & Huber, 2023). Therefore, the equipment has to be used for its designated production purpose. Furthermore, zero-defect manufacturing (Psarommatis & Kiritsis, 2019; Psarommatis et al., 2022), company-specific operation restrictions (Falkenauer, 2005), improved energy efficiency (Greinacher et al., 2020; Zhang et al., 2019), reduced emissions (Ji & Wang, 2019; Touzout & Benyoucef, 2019), sustainability, and Industry 4.0 trends (Gupta et al., 2021) require specialized production equipment, unintentionally creating potential limitation.

To include these production limitations and more company-specific additional requirements, we introduced the RALBP-TT formulation in Albus and Huber (2023). In this paper, we focus on the same three representative task types, namely *joining*, *handling*, and *separation*. However, the approach is not limited to these three task types but can incorporate more task types if desired.

## Reconfiguration

The aforementioned RALBP assumptions are used in the problem of designing new assembly lines but are applicable as well for the more common problem of changing an existing line. This becomes necessary to meet new production requirements, adapt to demand variety, or save investment costs (Oliveira et al., 2012; Makssoud et al., 2014). The hardly predictable variety in customer demands, due to increased customization, is a result of the ongoing shift from mass production to mass personalization (Piller, 2001), known as the reconfigurable production paradigm (Wiendahl et al., 2004; Bukchin et al., 2002; Koren & Shpitalni, 2010). With this shift comes increased competition, workload changes, quality requirements, and shortening product life cycles. These challenges lead to manufacturing companies struggling to achieve low costs and maintain high productivity, creating the need for efficient and flexible production systems (Çil et al., 2017). Many companies perform regular assembly line adjustments to address these issues, resulting in the problem of newly allocating tasks to workstations due to readjustments of an existing line. The reallocation of tasks is a problem called assembly line rebalancing (Falkenauer, 2005) or *reconfiguration* (Koren & Shpitalni, 2010; Yelles-Chaouche et al., 2020). Reconfiguration is an increasingly important topic for flexible production systems, such as robotic assembly lines, where more investigation is necessary (Çimen et al., 2022). Additionally, reconfiguration can lead to sustainable manufacturing by reducing waste and reusing existing production resources, an important topic in many modern-day productions (Chutima, 2022; Gupta et al., 2021).

Multi-cost approaches enabling cost-efficient assembly line reconfiguration are presented in different formulations (Makssoud et al., 2014; Pereira et al., 2018; Nilakantan et al., 2017; Boysen et al., 2022; Li et al., 2021). However, a real-world based cost-model, including savings for selling obsolete production equipment and investment in new equipment to meet the production requirements, is closer to companies' needs since competitiveness through cost reduction is a critical driver for many companies (Koren & Shpitalni, 2010; Napoleone et al., 2021). When the existing production line lacks the required capabilities to continue production, a trade-off arises between new investment, reuse, and selling unused equipment because the current line can only manufacture with new resource investment.

These requirements above are reflected in the cost model of Albus and Huber (2023) and adapted in this paper to the new GA structure, to formulate the objective functions.

This model incorporates new investment, reconfiguration and processing costs, and savings and differentiates between different locations for production equipment. Since new equipment needs to be bought, resulting in investment costs, whereas already available equipment in company storage or equipment in the current production needs to be reconfigured. All equipment are stored in a collective resource database, including its availability at vendors or in company storage, summarizing possible equipment at hand for the company. Equipment costs for already available equipment include a reconfiguration cost consisting of assembly, disassembly, and moving times for every piece of equipment. Additional processing costs represent the operating cost of a piece of production equipment for a fixed period, including all additional costs relying upon operation. The collective resource database extends the idea of a single resource database (Albus & Seeber, 2021) where all equipment has to be bought. Additionally, it allows for more company-specific requirements, such as time-critical reconfiguration, where only available equipment should be used, if possible.

## Solution methods

The problem of assigning tasks to equipment and these equipment to workstations is addressed with various solution techniques: heuristic, meta-heuristic, and exact methods. Heuristic approaches are presented in Capacho et al. (2006); Zhang et al. (2018); Borba et al. (2018); Kammer Christensen et al. (2017), but are often only suitable for first estimations due to their limitations in complex problem scenarios. Exact methods use branch and bound concepts (Dolgui & Ihnatsenka, 2009; Bukchin & Tzur, 2000; Vilà & Pereira, 2014; Ogan & Azizoglu, 2015), branch, bound and remember (Morrison et al., 2014; Sewell & Jacobson, 2012) or (mixed) IP (Öztürk et al., 2013; Bukchin & Tzur, 2000; Lopes et al., 2017) and more, for further information we refer to Li et al. (2020). However, these methods suffer from scalability issues and often claim effectiveness only on small problem instances. As a result, the development of more efficient and effective algorithms for solving large RALBPs is necessary (Chutima, 2022). In contrast, much research focuses on meta-heuristic approaches (Chutima, 2022), which counteract the scalability problem of exact methods. Meta-heuristic approaches are successfully applied in a wide variety of different assembly line problems: Ant Colony Optimization (ACO) approaches for U-shaped lines (Huo et al., 2018; Blum, 2008), in combination with mixed model production (Hamzadayi & Yildiz, 2012; Fisel et al., 2017; Kucukkoc et al., 2019), two-sided assembly lines (Gansterer & Hartl, 2018) or stochastic task times (Erel et al., 2005; Chakravarty & Shtub, 1986; Li et al., 2021). An overview can be found at Oesterle and Amodeo (2016) or Çil et al. (2017). GA approaches for the solution of RALBP are presented in Daoud et al. (2014); Levitin et al. (2006); Gao et al. (2009); Nilakantan et al. (2015); Yoosefelahi et al. (2012); Li et al. (2018, 2021), including Particle Swarm Optimization (PSO) for U-shaped lines (Nilakantan & Ponnambalam, 2016) and ACO (Huo et al., 2018; Kucukkoc & Zhang, 2016). Often GAs for single-objective problem formulations solve the problem in two steps and combine the algorithm with a heuristic (Rubinovitz et al., 1993). The recombination procedure for the GA is performed by a crossover operator, such as Fragment Reordering Crossover (Rubinovitz & Levitin, 1995), Order Crossover (Davis, 1985), Partially mapped Crossover (Goldberg et al., 1985), or Point Crossover (Nilakantan et al., 2015).

However, most of the GA approaches focus on the design of new production lines (greenfield), and only a few are applicable in the context of reconfiguration (brownfield), such as Yang et al. (2013); Zhang et al. (2018); Touzout and Benyoucef (2019). The objectives of Zhang et al. (2018) are minimizing the cycle time and rebalancing cost, where rebalancing cost is measured by the amount of tasks' reassignment between and within workstations. Whereas Yang et al. (2013) and Touzout and Benyoucef (2019) refer to equipment characteristics without equipment-dependent task times. These methods are not applicable in the context of RALBPs with equipment-dependent task times and the objective of reconfiguration cost reduction.

To the best of our knowledge, no GA is currently available to solve the reconfiguration of assembly lines with flexible production equipment and equipment-dependent task times. Another significant aspect of current GA in the context of RALBP is the lack of consideration for company specific requirement-sets. As a result, a new genetic model is necessary to solve the increasingly relevant reconfiguration of assembly lines. Furthermore, none of the abovementioned methods can account for different task types, an enhancement of the balancing problem to reflect the characteristics of modern-day production facilities, introduced in the RALBP-TT.

Accordingly, the main contribution of this paper is a GA for the cost-oriented reconfiguration of paced, straight-line RALBP-TT, including additional company constraints. This GA tackles the need for efficient and effective algorithms to solve the reconfiguration, thus closing in on large-scale RALBPs. Additionally, the GA includes task types represented in real-world production systems. We present a mathematical model with the possibility of supplementing the existing line if not all equipment are suitable for the new production.

This model uses investment costs, processing costs, and possible equipment savings to address the issues of current multi-cost approaches. Additionally, the model improves sustainable manufacturing by encouraging the reuse of exist-

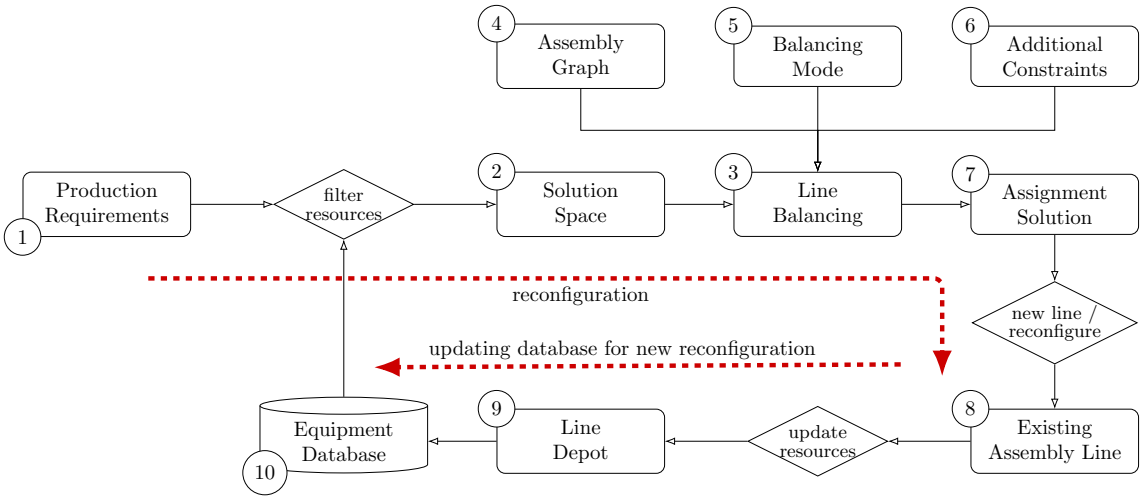



**Fig. 1** Flowchart overview of the reconfiguration approach for the assembly line balancing. Showcased are the inputs and the outputs of the *line balancing* (node 3), where the major contributions of the paper are. Additionally, the updating of the database for a new reconfiguration is showcased

ing equipment. The performance of the GA is evaluated on RALBP-TT problem instances.

The remainder of the paper is organized as follows: In Section "Problem definition", the problem formulation and notation are introduced, and the model is developed, starting from a decision variable formulation to a complete model. In Section "Methods", the representation, genetic operators, decoding heuristic and the new GA are explained. The model is then evaluated in Section "Results and discussion" by comparison of multiple problem instances with an exact approach. Finally, the conclusions and outlook are presented in Section "Conclusion and outlook".

## Problem definition

This section introduces the approach, our assumptions, the variable definition, and a mathematical formulation of the problem with two objective functions. One objective function represents the design of new assembly lines, and the other objective function is for the reconfiguration of assembly lines.

### Approach

This section presents the overall approach for assembly line balancing and reconfiguration. Additionally, the flowchart in Fig. 1 illustrates the approach, showcasing the inputs and the outputs of the line balancing and reconfiguration.

The input to the reconfiguration line balancing is the production requirements (node 1), such as weight, operations, or tool requirements, which are used to filter the equipment database. This filtering is based on the task requirements (e.g., weight, positioning accuracy, or distance), the equipment's mechanical characteristics (e.g., supported payload, provided accuracy or reach), and the equipment's supported operations, such as joining (e.g., welding or riveting), handling, and separation. These characteristics and operations can be added application-specific. They are irrelevant to the proposed algorithm since the algorithm is generally applicable. Then, the filtering procedure saves all suitable equipment meeting the production requirements in a solution space (node 2). The equipment database used for filtering includes new equipment available for purchase, equipment in company storage, and, if necessary for the reconfiguration, equipment in the current assembly line. This database represents real-world production equipment with different capabilities and characteristics. For this paper, we assume a filled database is available depicting the production capabilities of the customer. For more details on the filtering procedure and the resulting solution space, we refer to Albus and Seeber (2021) because this is not detailed in the further course of this paper.

The major contributions of the paper are in the *line balancing* (node 3). Here, the assembly line balancing is performed based on the assembly graph (node 4), possible equipment in the solution space, the balancing mode (node 5), and company-specific additional constraints (node 6). The balancing mode specifies if a new line is balanced or an existing line is reconfigured. As a result, an assignment solution output (node 7) of tasks to equipment and workstations is received. The assignment solution is used to set up a new assembly line or to reconfigure an existing line (node 8). Then, the available pieces of equipment in the line depot (node 9) are updated with the information from the new line. Next, the equipment database (node 10) is updated with all information from the line depot because these pieces of equipment can be used in the following reconfiguration

**Table 1** Variable description for the problem definition and mathematical model

| Variable name | Variable description |
|---|---|
| $n$ | Number of tasks |
| $m$ | Number of workstations |
| $r$ | Number of equipment |
| $i$ | Task index, $i \in \{1, \ldots, n\}$ |
| $j$ | Equipment index, $j \in \{1, \ldots, r\}$ |
| $k$ | Workstation index, $k \in \{1, \ldots, m\}$ |
| $t_{ij}$ | Processing time of task $i$ when performed by equipment $j$, in s |
| $ct$ | Cycle time, in s |
| $\mathcal{P}_i$ | Set of immediate predecessors of task $i$ |
| type($i$) | Type of task $i$ |
| $\mathbf{D}$ | Line depot containing already available equipment |
| $d_j$ | Number of existing equipment $j$ in the line depot |
| $a_j$ | Number of assigned equipment $j$ |
| $\underline{EC}$ | Equipment costs vector, in \$ |
| $ec_j$ | Cost of equipment $j$ |
| $\underline{PC}$ | Processing costs per second vector, in $\frac{\$}{s}$ |
| $pc_j$ | Processing cost for equipment $j$ |
| $\underline{SC}$ | Savings vector, in \$ |
| $sc_j$ | Savings per equipment $j$ |
| $\mathcal{C}$ | Company specific requirement set |

cycles. This cycle can be repeated for every new production requirement since the existing assembly line will be considered for reconfiguration.

## Assumptions

This paper adopts the RALBP-TT assumptions of Albus and Huber (2023), additionally stated here to clarify the problem setting:

(i) There is a given database containing all available equipment for the production process. Filtering of the database into solution spaces is applied based on task requirements against mechanical characteristics and supported operations.
(ii) Each solution space consists of at least one feasible piece of equipment and associates each piece of equipment with specific costs. The costs are investment costs, processing costs, and savings.
(iii) The previous assignment solution of the existing line is available.
(iv) The total duration time of tasks assigned to a given station must not exceed the predetermined cycle time.
(v) The total duration time of tasks assigned to a given equipment must not exceed the predetermined cycle time.
(vi) Assembly, disassembly, and moving times are considered for every piece of equipment individually and are offset with the savings for the equipment.
(vii) A subset of tasks can be enforced in one workstation.
(viii) Not all equipment in the solution space needs to be assigned.

## Variable definition

In Table 1, the variable notation for the mathematical model formulation is given.

## Mathematical formulation

We introduce different decision variables, each addressing a different issue: (i) the task-to-workstation assignment, (ii) the equipment-to-workstation assignment, and (iii) the task-to-equipment assignment. The following binary decision variables correspond to the aforementioned issues, respectively. We define for every task $i$ and station $k$:

$$x_{ik} = \begin{cases} 1, & \text{if task } i \text{ is assigned to workstation } k \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Additionally, we define for every equipment $j$ and every station $k$

$$y_{jk} = \begin{cases} 1, & \text{if equipment } j \text{ is assigned to workstation } k \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Lastly, for every task $i$ and every equipment $j$

$$z_{ij} = \begin{cases} 1, & \text{if task } i \text{ is carried out by equipment } j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Next, the mathematical formulation of the constraints of the IP is given, and the objective functions are defined in Section "Objective functions".

$$\sum_{k=1}^{m} k \cdot x_{gk} - \sum_{k=1}^{m} k \cdot x_{ik} \leq 0 \quad \forall g \in \mathcal{P}_i, i \tag{4}$$

$$\sum_{k=1}^{m} k \cdot x_{gk} - \sum_{k=1}^{m} k \cdot x_{ik} = 0 \quad \forall g \in \mathcal{P}_i, i \tag{5}$$

s.t. $i, g \in \mathcal{C}$

$$\sum_{k=1}^{m} x_{ik} = 1 \quad \forall i \tag{6}$$

$$\sum_{j=1}^{r} z_{ij} = 1 \quad \forall i \tag{7}$$

$$z_{ij} \cdot t_{ij} \geq 0 \quad \forall i, j \tag{8}$$

$$\sum_{i=1}^{n} x_{ik} \sum_{j=1}^{r} z_{ij} \cdot t_{ij} \leq ct \quad \forall k \tag{9}$$

$$x_{ik} \in \{0, 1\} \quad \forall i, k \tag{10}$$

$$z_{ij} \in \{0, 1\} \quad \forall i, j \tag{11}$$

$$y_{jk} \in \{0, 1\} \quad \forall j, k \tag{12}$$

The precedence constraints are enforced by constraint (4). Constraint (5) ensures that task $i$ and task $g$ get assigned to the same workstation if the tasks follow each other in the precedence graph. This allows for a company-specific requirement set $\mathcal{C}$ in the task-to-workstation assignment. For instance, if a handling task follows a separation task, both must be assigned to the same workstation. Thus, the set $\mathcal{C}$ is defined as $\mathcal{C} = \{\text{type}(g)\text{=separation} \wedge \text{type}(i)\text{=handling} | \forall g \in \mathcal{P}_i, i\}$, Constraint (6) imposes that each task $i$ gets assigned to exactly one station $k$. Additionally, each task must be assigned to one piece of equipment. This is guaranteed by constraint (7). If equipment $j$ cannot perform task $i$, the corresponding processing time $t_{ij}$ is assumed to be $t_{ij} = -1$. This encodes equipment $j$ is infeasible, as all other processing times are non-negative floating point values. Therefore, constraint (8) ensures each task is assigned to suitable equipment, able to process the task. Constraint (9) ensures that the total time of all tasks at one workstation cannot exceed the cycle time. The domain of the decision variables $x_{ik}$, $z_{ij}$, and $y_{jk}$ is defined in constraint (10) to (12).

## Objective functions

The following section presents two cost-related objective functions, one for the greenfield and one for the brownfield case. The cost approach of Albus and Huber (2023) was adopted to match the proposed GA structure, and consists of investment costs in the greenfield case and investment costs, processing costs, and savings in the brownfield case. Since all additional costs for the greenfield setup can be included in the investment costs, it is enough to use only investment costs for the objective function. The investment costs vector $\underline{EC}$ contains for each equipment $j$ the cost $ec_j$. A minimization objective function for the greenfield cost $C_g$ approach is defined as

$$\min C_g = \sum_{k=1}^{m} \sum_{j=1}^{r} y_{jk} \cdot ec_j \quad \in \mathbb{R}. \tag{13}$$

In the brownfield case, the line depot $\mathbf{D} \in \mathbb{Z}^{r' \times m'}$ describes the amount of equipment $j$ in the existing line as $d_j$, and $r'$ is the number of equipment and $m'$ as the number of workstations of an existing line. Processing costs $\underline{PC}$ represent the operating cost of equipment $j$ for a fixed period, with all additional costs relying upon operation. This processing cost can include energy consumption, inspection, and maintenance costs, environmental cost aspects, and more company-defined variable costs. The total processing cost is $pc_j$ for every piece of equipment $j$.

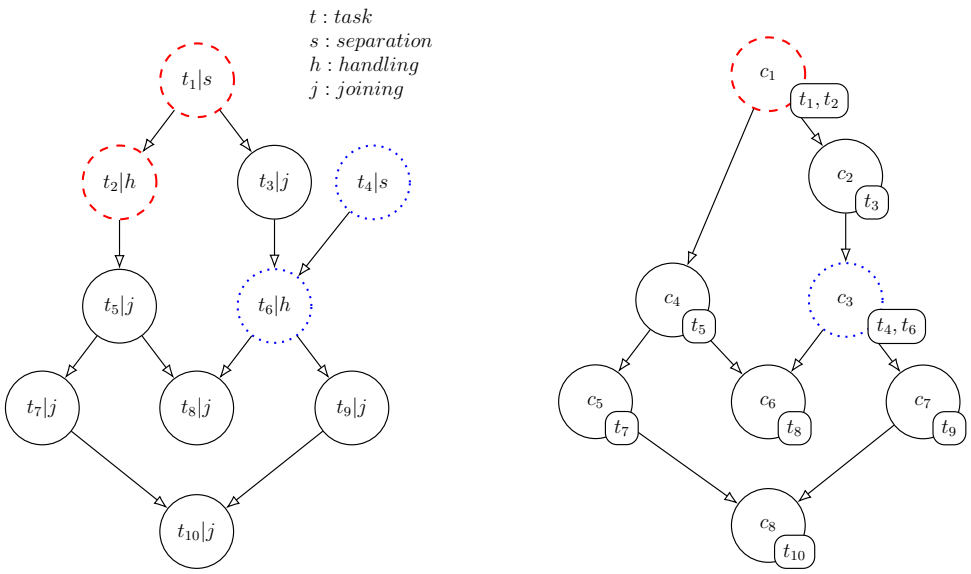
Equipment in the line depot $\mathbf{D}$ have only processing costs and no investment costs since they are readily available, yet not free to use. The amount of money for selling available equipment $j$ is represented as $sc_j$ in the savings vector $\underline{SC}$. Alternatively, one could use storage costs as savings, to represent equipment which is put into storage and not directly sold. This can be defined for every equipment independently, to account for special equipment which should not be sold. We assume that the investment costs are higher than the savings $0 \leq sc_j \leq ec_j$. If a new piece of equipment is obtained, the investment costs $ec_j$ are used. An auxiliary variable $a_j$ is introduced, which represents the amount of one equipment $j$ in the total line,

$$a_j = \sum_{k=1}^{m} y_{jk} \quad \forall j. \tag{14}$$

Using the auxiliary variable $a_j$, we can define the brownfield minimization objective function $C_b$ as

$$\min C_b = c_{inv} + c_{prc} + c_{sav} \quad \in \mathbb{R} \tag{15}$$

$$c_{inv} = \sum_{j=1}^{r} \max(0, a_j - d_j) \cdot ec_j \tag{16}$$




(a) Task precedence graph. (b) Cluster precedence graph.

**Fig. 2** An example assembly precedence graph with task numbers and task types is shown in Fig. 2a. This graph is used to illustrate the clustering procedure of tasks. Tasks $\{t_1, t_2\}$ are clustered in $c_1$ and $\{t_4, t_6\}$ are clustered in $c_3$ based on exemplary customer requirements. The precedence graph with corresponding clusters is shown in 2b, clustered tasks are indicated by style and color (Color figure online)

**Table 2** Cluster notation for grouping separation tasks preceding handling tasks. This enforces the clustered tasks into the same workstation

| Variable name | Variable description |
|---|---|
| $n'$ | Number of clusters |
| $i'$ | Cluster index $i' \in \{1, \ldots, n'\}$ |
| $v$ | Mapping from tasks to clusters: $v : i \rightarrow i' := \{v(i) \vert i \in \{1, \ldots, n\}\} = \{i' \vert \forall i' \in \{1, \ldots, n'\} \exists i\}$ |

$$c_{prc} = \sum_{j=1}^{r} a_j \cdot pc_j \tag{17}$$

$$c_{sav} = \sum_{j=1}^{r} \min(0, a_j - d_j) \cdot sc_j \tag{18}$$

Equation (16) represents the investment costs for all equipment, the processing costs are added with Eq. (17), and with Eq. (18) the negative savings (or positive storage costs) are added.

## Methods

The following section presents a GA approach for solving the optimization problem introduced in Section "Problem definition". First, the binary variables of the IP problem formulation are substituted with a vector representation for use by the GA and the proposed heuristic. Additionally, clusters enforce the task type-related constraint (5). Next, problem-specific genetic operators and heuristics are combined into the GA.

### Representation

This section defines the vector representation for the algorithm and the cluster notation. The different task types in the problem formulation can be exploited to reduce the number of relevant precedence constraints. We define the exemplary company-specific requirement set $\mathcal{C} = \{\text{type}(g)\text{=separation} \wedge \text{type}(i)\text{=handling} \vert \forall g \in \mathcal{P}_i, i\}$ enforcing separation tasks preceding handling tasks to be carried out in the same workstation. However, different requirement sets are possible. Therefore, tasks affected by this constraint can be considered as one task when assigning them to workstations. As a result, we introduce task clusters, which are sets of tasks to be assigned to a single workstation. Further information on the clusterization procedure is provided in Appendix A. Figure 2a represents an example task precedence graph, whereas Fig. 2b showcases the summarized cluster graph. Furthermore, we extended our notation for the clusters according to Table 2.

We use $v(i) : i \rightarrow i'$ as a mapping from task $i$ to cluster $i'$, and the number of clusters is upper bound by the number of tasks $n$. Therefore, $1 \leq n' \leq n$, and additionally we define all clusters to be non-empty.

Next, a solution representation with three vectors is provided. The decision variables (i.e., $x_{ik}$, $y_{jk}$, and $z_{ij}$ of Eqs. (1) to (3)) of the IP formulation given in Section "Problem definition" are replaced by vectors. Because with the new representation, clusters are assigned to workstations and thus, precedence constraints applying to tasks inside the same cluster can be omitted. The remaining precedence con-

straints apply to clusters, i.e., the precedence constraints $g \rightarrow i \ \forall g \in \mathcal{P}_i, i$ turns into $g' \rightarrow i' \ \forall g' \in \mathcal{P}_{i'}, i'$.

The first vector $\underline{u} \in \mathbb{Z}^{n'}$ with elements $u_{i'} = \pi(i')$, contains a permutation of all non-empty clusters $i'$, denoted by $\pi(i')$. The clusters in the vector respect the cluster precedence constraints. A reordering procedure to convert random permutations into a feasible permutation order was proposed by Rubinovitz and Levitin (1995). However, the procedure requires an acyclic cluster precedence graph, which is enforced in our problem setting.

The vector $\underline{w} \in \mathbb{Z}^{n'}$ contains the number of the workstation, to which a cluster is assigned. Therefore, $w_{i'} = k_{i'}$ gives us the workstation number $k$ to which cluster $i'$ is assigned. The maximum number of workstations $m$ is up to change during optimization because better equipment combinations yield a lower number of maximum workstations. Workstation numbers in vector $\underline{w}$ are ordered ascending and contain at least one cluster.

The third and last vector $\underline{q} \in \mathbb{Z}^n$ contains the equipment-to-task assignment, with element $q_i = j_i$ as the equipment $j$ assigned to task $i$.

## Genetic algorithm

The previous sections introduced the different problem-specific designs of the modified GA used to solve the RALBP-TT. In this section, the specialized GA is described, and an overall flowchart is presented in Fig. 3.

The previously introduced methods operate on the individual triplets of $(\underline{u}, \underline{w}, \underline{q})$. Because the GA operates on populations with population size $\mu$, so $\mu$ possibly different population triplets $(\underline{u}, \underline{w}, \underline{q})$ are considered at once. Each individual's genome is given by its cluster permutation vector $\underline{u}$.

Next, we outline the step-by-step process of the GA. As detailed in Section "Related work", there exists no GA that adequately addresses the reconfiguration of RALBPs. Consequently, we developed our own GA implementation to tackle the problem setting outlined in Section "Problem definition" including company-specific requirements, modifying the conventional GA workflow. Detailed information on the problem-specific algorithm design for greenfield and brownfield scenarios for station and equipment assignment using clusters is provided in Appendix B. This cluster assignment tackles companies' needs to enforce specific tasks into one workstation or enforce the separation of tasks, which should be divided due to safety or production requirements. The overall flowchart in Fig. 3 depicts the workflow, with the newly developed stages highlighted in green. First, an initial random population (node 1) is generated with a permutation of all clusters, followed by a first assignment of equipment and workstations to clusters (node 2) and evaluation of the objective function (node 3). The initial assignment uses the

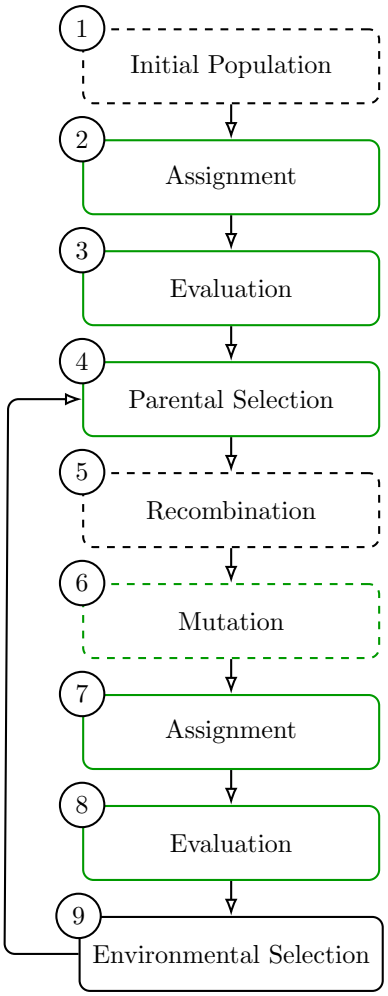



**Fig. 3** Flowchart of the sequential steps performed in the proposed GA for optimizing the objective. The dashed outlines mark states, which need to be reordered to obtain a feasible genome concerning the precedence relations. Highlighted in green are problem-specific designs of the introduced GA (Color figure online)

developed station and equipment assignment procedure to determine suitable equipment for the clusters efficiently. This design leverages the efficiency and exploration of the two equipment heuristics strategies to ensure the initial population is a semi-optimized starting point with sufficient variance. As a result, subsequent optimization steps are provided with a reasonable initial configuration. Since a feasible initial solution is not guaranteed due to randomization, the reordering procedure of Rubinovitz and Levitin (1995) is applied after every genome modification. This reordering procedure repairs possible violated precedence constraints in the genome and has a guaranteed termination. Because this reordering is applied multiple times, we marked every node in the GA flowchart using the reordering procedure in green, with dashed lines in Fig. 3. In the parental selection (node 4), the next-generation parents are chosen according to their objective value, and low-cost parents are chosen for

reproduction. Next, children are generated by recombining (node 5) the parents chosen in the selection step through Order Crossover (OX). OX performs a permutation of all clusters for the children (Davis, 1985) since it guarantees that each cluster is present within every child. This ensures the operation of the reordering procedure, which is performed again because OX does not guarantee feasible precedence constraints for the children. A mutation (node 6) is required for the next step, which introduces a small modification to the genome by swapping two clusters. The mutation operates on the clusters to satisfy the company-specific requirements. Again, the possibly infeasible genome is repaired by applying a reordering procedure, followed by an assignment (node 7) and evaluation (node 8). Lastly, a subset of individuals must be removed to keep the population size stable. In the environmental selection (node 9), the weakest individuals (the most expensive solutions) are replaced by new children created with OX. This sequence is identical for the greenfield and the brownfield problem formulation.

As a stopping criterion, we introduce the concept of age. All individuals get assigned an age, initially set to zero when genomes are generated or by the combination of parents. In every iteration, where an individual is not replaced by a child, the age of that individual increases by one. When the age of the best individual exceeds a user-defined threshold, the GA stops. As a result, age measures the number of generations the best individual has remained unchanged, which indicates its maturity and resistance to change. The oldest generation acquires the best objective value, representing the best result the GA can achieve in the given horizon. This has proven to be the best suited stopping criterion for the modified GA in the RALBP-TT.

### Heuristic

A heuristic is used for the assignment procedure to provide an initial baseline. The heuristic takes a valid permutation of all clusters and assigns workstations and equipment using a lower bound time estimation. This time estimation is computed using the fastest feasible robot for each task. Lower bounds $ct_{\min}(i')$ for the time spent to carry out all tasks in a single cluster $i'$ are computed as

$$ct_{\min}(i') = \sum_{i \in i'} \min_{\forall j} t_{ij} \quad . \tag{19}$$

The equipment heuristic combines two strategies, represented by two scores for each piece. First, a cost score normalizes the investment cost for a piece of equipment by the number of tasks it can perform. Second, the time score normalizes the total time of all possible tasks for a piece of equipment by the number of tasks it can perform.

This time score is used to encourage the choice of equipment with a lower normalized task time, which can reduce the number of stations. The overall cost is not directly reduced with the time score. However, the possibility of using a single piece of equipment for more tasks can reduce the overall number of equipment and, thus, indirectly reduce the total cost. A fixed trade-off parameter $\alpha$ is used to favor either cheaper equipment costs (cost score $s_{c,j}$) or faster processing times (time score $s_{t,j}$) in the heuristic, as total score $s_j = (1-\alpha) \cdot s_{c,j} + \alpha \cdot s_{t,j}$. The heuristic then assigns the equipment with the lowest total score $j = \arg\min s$ for every task if the equipment can perform the task.

Since the heuristic trade-off parameter depends on the available equipment, the genetic algorithm optimizes every generation's trade-off $\alpha$ with the following approach. For the first generation, $\alpha$ is randomly drawn from the unit interval. This initialization allows for more variance in the initial population and can encourage the choice of quicker equipment, which reduces the number of stations. In subsequent generations, the parameter is inherited from one parent, and noise is added to the inherited parameter to allow exploration in the solution space. As a result of this optimization, the trade-off parameter does not require manual tuning.

## Results and discussion

The presented GA and related procedures yield feasible solutions, but they cannot guarantee optimal solutions. In the following sections, we analyze the performance of the proposed approach by applying the algorithm to a reconfiguration dataset with 20 tasks and different equipment alternatives. Next, we compare the results against the exact IP solution approach of Albus and Huber (2023) based on the defined performance metrics.

### Experiments

For a comprehensive analysis of the results, we use our previously reconfiguration dataset[1] (Albus and Huber, 2023). This dataset includes pre-defined investment costs, processing costs, and savings for each of the $n = 20$ tasks and $r$ in $\{5, 10, 15, 20\}$ equipment alternatives.

The instances for every number of equipment alternatives $r$ are solved using an IP solver[2] and the presented GA. The GA is implemented in *C++* to achieve the best performance, the IP uses a Python interface to the solver. Data handling for the results, loading of the reconfiguration dataset and solution export were performed in *Python 3.10*. Since the IP solutions are optimal, we use them as the baseline for comparison.

[1] Available at https://github.com/maralbus/reconfiguration_dataset

[2] Gurobi 9.5.2 https://www.gurobi.com/

Since the GA has multiple hyperparameters to be adjusted, the values for the most significant parameters, namely *age* and *population size*, are determined by experiments. The experiment values are in the range of {100, . . . , 1000}, which covers the significant range of these parameters. Further information on the experiments are provided in Appendix C. Based on the experiments, the GA uses a population size of 300 individuals and replaces two-thirds ($p_{abandon} = 0.66$) of the individuals in each generation. Calculation terminates when the best solution reaches the age of 500. Results are averaged over ten runs for each instance.

Experiments empirically determined the population size, replacement split, and age. These experiments have shown that the average number of generations is the most crucial parameter. To reach convergence, the average number of generations is 116.1 for the greenfield and 102.0 for the brownfield problem. Because of this, the chosen age value is confirmed to be a suitable trade-off parameter between optimization time and early termination. It could be argued that the chosen age value is already too conservative, with the maximum convergence generations ranging from 203.8 ($r = 5$) to 328.6 ($r = 20$) for the greenfield and 224.5 ($r = 5$) to 263.4 ($r = 20$) for the brownfield case. However, the stop criterion was chosen to guarantee that the GA has enough generations to evaluate its capabilities on the total cost while maintaining its computation time reduction benefit. Additional experiments on the age criteria were performed to evaluate the robustness and performance increase/decrease through extended/reduced age values. These experiments confirm the chosen age value. Additional information is provided in Appendix C Section "Age parameter". In conclusion, no further optimization could be achieved even if the GA would run the same amount of time as the IP baseline.

## Metrics

In this section, the metrics for the evaluation are described and summarized in Table 3. The metrics are used for the evaluation of the experiments, and computed for solutions of every instance. The number of equipment $R$, the number of (non-empty) workstations $S$, the cost objective $C$, the efficiency $E$ according to Eq. (20), and the computational runtime $T$ are evaluated.

$$E = \sum_{k=1}^{m} \sum_{i=1}^{n} \frac{x_{ik} \cdot t_i}{m \cdot ct} \tag{20}$$

A solution's objective value $C$ can be used as a quality measure. $R$, $S$, and $E$ are technical properties of the assembly line and are used to explain differences in the objective value. Moreover, the execution time $T$ of the algorithm is also included, timed on an AMD Ryzen 7 3700X with 64 GB RAM. Since the GA can deviate in its assignment solution from run to run, we evaluate the average and best run of all runs against the IP solution. This demonstrates the expected outcome of a singular run in the average and possibilities of the GA in the best values.

**Table 3** Evaluation metrics for the benchmark of the baseline IP against the presented GA approach, the superscript indicates the solution method

| Metric | IP | GA | Objective |
|---|---|---|---|
| Total costs $C$ | $C^{ip}$ | $C^{ga,min}$ | Minimize |
| | – | $C^{ga,avg}$ | |
| Num. of equipment $R$ | $R^{ip}$ | $R^{ga,min}$ | Minimize |
| | – | $R^{ga,avg}$ | |
| Num. of workstations $S$ | $S^{ip}$ | $S^{ga,min}$ | Minimize |
| | - | $S^{ga,avg}$ | |
| Efficiency $E$ | $E^{ip}$ | $E^{ga,max}$ | Maximize |
| | – | $E^{ga,avg}$ | |
| Computational runtime $T$ | $T^{ip}$ | $T^{ga,min}$ | Minimize |
| | – | $T^{ga,avg}$ | |

$ga$ : Genetic Algorithm
$ip$ : Integer Programming
Additionally, $min/max$ indicates the best solution of all runs, and $avg$ indicates the average over all runs

## Discussion

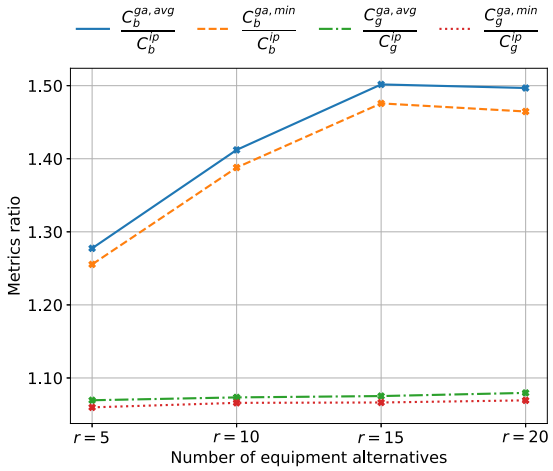
This section analyzes and discusses the evaluation metrics for the GA and IP solutions. Next, the average and best values of all solutions are discussed, and evaluated on the metric ratios of Eq. (21). For the GA, the $min$ ($max$ for efficiency $E$) values represent the best solution over ten runs for each instance of one equipment alternative $r$. The $avg$ values represent the average over ten runs for each instance for one equipment alternative $r$.

$$\frac{C^{ga}}{C^{ip}}, \frac{R^{ga}}{R^{ip}}, \frac{S^{ga}}{S^{ip}}, \frac{E^{ga}}{E^{ip}}, \frac{T^{ga}}{T^{ip}} \tag{21}$$

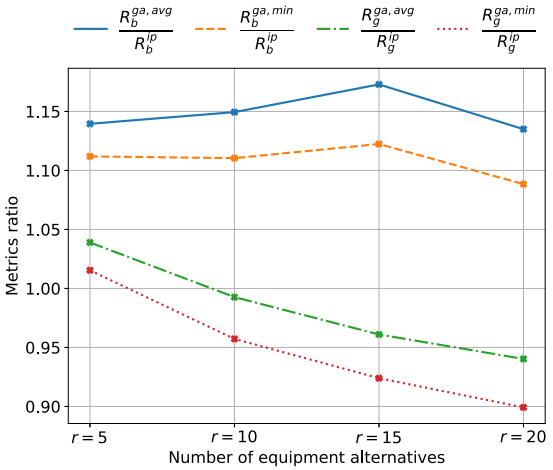
These ratios compare the total costs $C$, the number of equipment $R$, the number of used workstations $S$, the line efficiency $E$, and the computation time $T$. The greenfield and brownfield problem formulation results are presented in Fig. 4, and the exact numbers generated by the experiments are reported in Table 4. For the $ga$ to $ip$ metrics of $C$, $R$, $S$, and $T$, a quotient value smaller than one indicates a better performance of the GA compared to the IP. Whereas, for metric $E$, a quotient value bigger than one indicates a better performance.

### Total cost

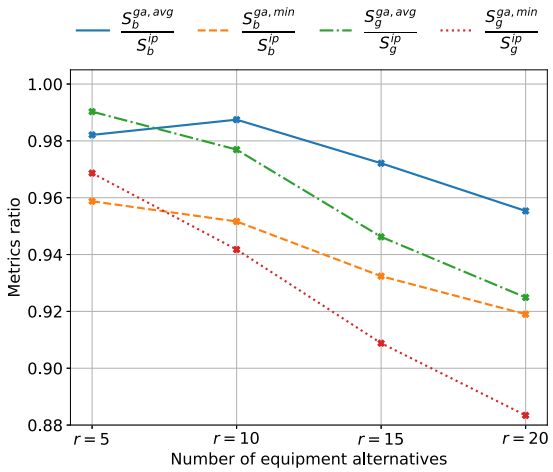
The average cost ratios $C$ for the greenfield instances (see Fig. 4a) show a deviation from the optimal costs by 6.95%



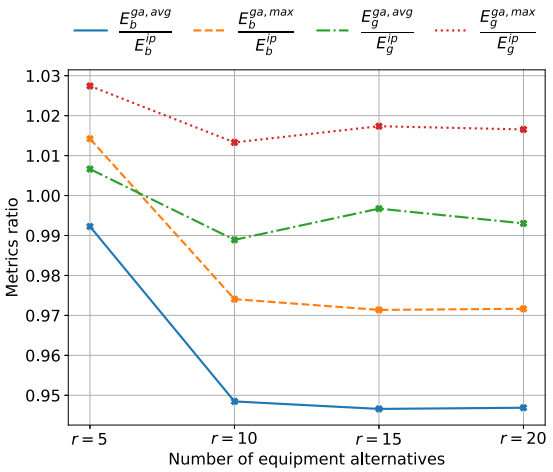

(a) Cost ratio comparison.



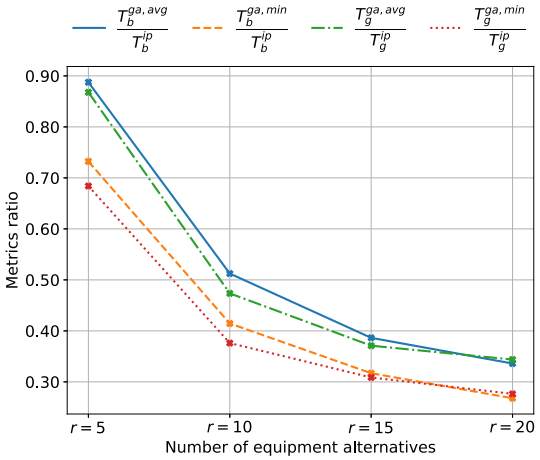

(b) Number of equipment ratio comparison.




(c) Number of workstations ratio comparison.




(d) Efficiency ratio comparison.




(e) Computation time ratio comparison.

**Fig. 4** Analysis plots for the comparison between GA and IP solutions for the greenfield $g$ and brownfield $b$ problem instances. For the GA, the $min$ ($max$ for $E$) values represent the best solution over ten runs for each instance, averaged over all instances for one $r$. The $avg$ values represent the average over ten runs for each instance, averaged over all instances for one $r$. The number of equipment alternatives for each instance is shown on the x-axis. The metrics ratio is shown on the y-axis. Depicted metrics ratios include cost $C$ in Fig. 4a, number of used equipment $R$ in Fig. 4b, number of used workstations $S$ in Fig. 4c, efficiency $E$ in Fig. 4d and computation runtime $T$ in Fig. 4e

**Table 4** Experiment results for the evaluation metrics total costs $C$, number of equipment $R$, number of workstations $S$, efficiency $E$ and computational runtime $T$ for the problem instance with $r$ in {5, 10, 15, 20} available equipment alternatives

| Objective | Metric | Eval | $r = 5$ | $r = 10$ | $r = 15$ | $r = 20$ |
|---|---|---|---|---|---|---|
| Greenfield | $C^{ga}/C^{ip}$ | avg | **1.069** | 1.073 | 1.075 | 1.080 |
| | | min | **1.060** | 1.066 | 1.066 | 1.069 |
| | $R^{ga}/R^{ip}$ | avg | 1.039 | 0.993 | 0.961 | **0.940** |
| | | min | 1.015 | 0.957 | 0.924 | **0.899** |
| | $S^{ga}/S^{ip}$ | avg | 0.990 | 0.977 | 0.946 | **0.925** |
| | | min | 0.969 | 0.942 | 0.909 | **0.883** |
| | $E^{ga}/E^{ip}$ | avg | **1.007** | 0.989 | 0.997 | 0.993 |
| | | max | **1.027** | 1.013 | 1.017 | 1.017 |
| | $T^{ga}/T^{ip}$ | avg | 0.868 | 0.474 | 0.371 | **0.344** |
| | | min | 0.684 | 0.376 | 0.309 | **0.276** |
| Brownfield | $C^{ga}/C^{ip}$ | avg | **1.277** | 1.412 | 1.502 | 1.497 |
| | | min | **1.255** | 1.388 | 1.476 | 1.465 |
| | $R^{ga}/R^{ip}$ | avg | 1.140 | 1.149 | 1.173 | **1.135** |
| | | min | 1.112 | 1.110 | 1.122 | **1.088** |
| | $S^{ga}/S^{ip}$ | avg | 0.982 | 0.987 | 0.972 | **0.955** |
| | | min | 0.959 | 0.952 | 0.932 | **0.919** |
| | $E^{ga}/E^{ip}$ | avg | **0.992** | 0.948 | 0.947 | 0.947 |
| | | max | **1.014** | 0.974 | 0.971 | 0.972 |
| | $T^{ga}/T^{ip}$ | avg | 0.887 | 0.512 | 0.386 | **0.336** |
| | | min | 0.732 | 0.415 | 0.317 | **0.268** |

Shown are the averaged $avg$ and best $min$ ($max$ for efficiency $E$) GA to IP greenfield and brownfield ratios. Highlighted are the best results in each row

($r = 5, avg$) to 7.96% ($r = 20, avg$). For the best cost ratios, only 5.98% ($r = 5, min$) to 6.94% ($r = 20, min$) deviation from the baseline is achieved. The slight difference between best and average values of 0.90% confirms that the GA achieves similar objective values on subsequent runs with low deviation. However, all objective values are worse than the IP solution. The optimal cost for brownfield instances deviates even further than in greenfield instances. The average deviation from the optimal solution is between 27.74% ($r = 5$, $avg$) and 50.17% ($r = 15, avg$) for the brownfield solutions. Additionally, the best brownfield objective values range from 25.54% ($r = 5, min$) to 47.58% ($r = 15, min$) worse than the baseline. The best objective values are achieved with the fewest equipment alternatives. This increase in total costs for the brownfield to greenfield comparison hints towards problems with the more complex reconfiguration tasks of balancing the trade-off between new investment, reuse, and selling unused equipment. An extended computation time for better objective values through an increased stop criterion would not benefit the GA since convergence is reached within the given horizon. A possible explanation for the increase in total costs might be found in the sequential nature of the GA. Tasks, equipment, and stations are considered sequentially, which proves more beneficial for the greenfield instances where no available pieces of equipment need to be considered. However, for the brownfield instances, this hinders the GA from achieving similar results as for the greenfield instances. Although a trade-off parameter $\alpha$ is already introduced to help with exploration, to overcome these local minima, increased exploration could benefit the GA. On the other hand, computational benefits and, thus, scalability are limited by this countermeasure. In conclusion, further improvements for the GA, or a completely new GA, are necessary to reach their full potential.

#### Number of equipment

When comparing the number of equipment $R$ used in the solutions, the greenfield instances show a steady improvement. Depending on the available equipment alternatives, one can first observe the use of 3.90% ($r = 5, avg$) more equipment. This decreases to less equipment ($r$ in {10, 15, 20}, $avg$) than the baseline with more equipment alternatives. With more equipment alternatives, the average GA solutions improve steadily, at best 5.97% ($r = 20, avg$) less equipment than the baseline. The best solutions of the GA confirm this improvement trend, at best 10.07% ($r = 20, min$) better than the baseline. In contrast, the GA tends to use more equip-

ment than the baseline in the brownfield instances, 14.92% ($avg$) more on average. The evaluation of the best values $min$ shows an improvement to 10.82% ($min$) on average for the brownfield instances. Still, more equipment are used every time in comparison to the baseline. The ratios and numbers of assigned equipment $R$ improve when the number of available equipment alternatives increases for both the greenfield and the brownfield instances. This showcases the potential for even more equipment alternatives, varying types of equipment or different problem settings. The presented numbers showcase the capabilities of the GA to leverage the equipment at hand to better assignment combinations in the greenfield setting. However, reusing already available equipment for the brownfield instances is more challenging for the GA than for the IP, as discussed in the previous section. This increase in the amount of used equipment $R$ confirms the additional complexity and resulting problems with reconfiguring existing equipment against the investment in new equipment. We assume that the equipment ratios in the brownfield setting are connected to the worse objective values of the GA. The GA struggles to achieve the same quality of objective values as in the greenfield solutions. Since the optimization in this study is focused on total costs, the usage of more pieces of equipment is not discouraged. However, the evaluation showcases a correlation between the amount of equipment used and total costs. This correlation depends on the characteristics of the equipment available and might vary for different equipment databases. With better objective values $C$, the equipment $R$ ratios would presumably showcase identical behavior for both problem formulations. Further experiments could investigate different objective functions focused on optimizing the number of equipment if required. The values are presented in Fig. 4b.

#### Number of workstations

The best GA solutions achieve 5.82% ($r = 5, min$) up to 11.66% ($r = 20, min$) less used workstations $S$ than the IP solution for the greenfield instances. On average, all instance solutions use fewer workstations, up to 7.51% ($r = 20, avg$) fewer workstations. This trend continues for the brownfield instances. The best average results showcase 4.47% ($r = 20, avg$) fewer workstations used in the brownfield setting. The values increase for the best results, with 4.12% ($r = 5, min$) to 8.10% ($r = 20, min$) better ratios on the number of used workstations $S$. On average, the GA showcases the capability to use fewer workstations than the IP solution. This result confirms the capabilities of the GA to use less equipment in the greenfield case through favorable assignment solutions. Additionally, this showcases one of the most notable differences between the greenfield and brownfield instances on all metrics for the GA solutions. The number of used workstations is always better for the GA solutions than the IP solution, which was not the case for the costs $C$ and the number of equipment $R$. This result strengthens the showcased performance of the GA solution. It confirms that the self-optimized trade-off parameter $\alpha$ can help to reduce the number of workstations through exploration and favorable equipment-to-workstation combinations. Additionally, we conclude that the GA is more conservative when opening new stations, especially with more equipment alternatives. With this finding, the higher solution cost of the GA must result from the choice and combination of equipment in the workstations because less equipment and workstations than the baseline are required. Additional variance in the recombination or mutation steps of the GA could benefit in finding favorable combinations to decrease the total cost. All ratios are shown in Fig. 4c.

#### Efficiency

The best instance solutions achieve up to 2.74% ($r = 5, max$) better maximum efficiency $E$ compared to the IP solutions on the greenfield instances. For all equipment alternatives, the best solutions achieve better efficiency than the baseline. The averaged solutions achieve better efficiency for fewer equipment alternatives, with 0.66% ($r = 5, avg$), but drop up to 1.11% ($r = 10, avg$) worse efficiency on all other equipment alternatives. The numbers for the brownfield instances showcase reduced efficiency. The best runs achieve only 1.42% ($r = 5, max$) better efficiency than the IP solution. At worst, the efficiency decreases to 2.86% ($r = 15, max$), worse than the baseline. Additionally, the average efficiency declines for more equipment alternatives, reaching its worst efficiency with 5.34% ($r = 15, avg$) worse than the baseline. Only one of the best runs can achieve the same efficiency in the brownfield setting as the IP baseline. However, some solutions have shown higher efficiency $E$ and thus make more efficient use of equipment concerning the cycle time constraint, but only in the best runs. We assume the slightly worse efficiency, together with the improvements in the number of equipment and workstations, is directly connected with the higher total cost of the objective. Fewer and faster equipment are selected at increased cost, leading to worse efficiency but fewer equipment and workstations. Efficiency ratios are shown in Fig. 4d.

#### Computation time

Next, the computation time $T$ advantage of using a GA is presented, and ratios are shown in Fig. 4e. The measured average runtime shows that the GA greenfield solutions are 1.15× ($r = 5, avg$) to 2.91× ($r = 20, avg$) faster and the brownfield solutions are 1.13× ($r = 5, avg$) to 2.98× ($r = 20, avg$) faster in terms of computation time. Absolute times for the GA are $< 1$ second for all instances and equipment alternatives At best, the computation is 1.46× ($r = 5,$

$min$) to 3.61× ($r = 20, min$) faster in the greenfield setting, and 1.37× ($r = 5, min$) to 3.73× ($r = 20, min$) in the brownfield setting. On average, the GA is $\sim$ 2.22× faster than the IP for greenfield and $\sim$ 2.66× for the brownfield problem instances. An expected computation time benefit is apparent, which confirms that the GA computation was stopped sufficiently early by the *age* criterion. However, the decrease in the objective value might only be tolerable in some industries. The most promising aspect of Fig. 4e is the constant decrease in computation time with increased equipment alternatives. With more equipment alternatives and thus increased complexity, the GA extends its computation time benefit whereas the IP baseline scales poorly. This behavior indicates the scalability benefits of the GA for complex problem settings with more equipment alternatives and larger problem instances. Additional benefits could be leveraged by further decreasing the stop threshold. However, further experiments are necessary to confirm the consequences of early termination on the total cost. The experiment results demonstrate the capabilities of the GA regarding approximation ability and computational efficiency.

The benefit of the proposed method is higher for the greenfield problem due to a better objective value (on average 7.44% worse than the baseline). Additionally, it improves computation time (on average 2.22× less computation time). For the brownfield problem, the benefit firmly declines for the objective value (on average 42.20% worse than the baseline) and computation time (on average 2.16× less computation time). At best, the proposed method decreases computation time by $\sim$ 2.67× on the brownfield problem, and up to $\sim$ 2.75× on the greenfield problem. All values are showcased in Fig. 4, and the exact data values are presented in Table 4.

## Conclusion and outlook

This paper's main contribution is a Genetic Algorithm (GA) to efficiently solve the reconfiguration and setup of new assembly lines to achieve flexible production systems. The algorithm is a step towards scalability in Robotic Assembly Line Balancing Problem with Task Types (RALBP-TT), thus closing in on large-scale problems.

A mathematical model is proposed to solve the RALBP-TT by combining equipment selection and assembly line balancing with equipment alternatives for reconfiguring existing lines. For the reconfiguration, investment costs, savings for selling unused equipment, and processing costs of the whole line for a given time horizon are considered. This model's goal is assigning tasks to equipment to workstations while minimizing the total cost of the line. Additional company-specific constraints are included in the model to respect specialized equipment capabilities.

We present a GA to solve the mathematical model and compare the results to an Integer Programming (IP) solution for a reconfiguration dataset consisting of various problem instances with multiple equipment alternatives. The aim of the GA model is the decrease of computation time in comparison with exact solution methods to enable scalability.

In conclusion, the proposed method achieves worse total cost, better number of equipment, and used workstations, in comparison with the IP baseline solution. Moreover, the GA achieves close, but worse, efficiency and significant computation time reduction for the greenfield problem. The reconfiguration problem instances have proven to be more complex, increasing the total cost and the number of equipment. Additionally, the efficiency and the benefit of computation time reduction decreases. Nonetheless, the GA still significantly reduces computation time in addition to a decrease in the number of used workstations. In addition, it achieves impressive results for efficiency without optimizing this metric. The results demonstrate the capabilities of the GA regarding approximation ability and scalability by computational efficiency for RALBP-TTs. Furthermore, the GA has proven robust against hyperparameter changes and could extend its computational benefit if a higher total cost is satisfactory. Especially in more complex problem instances, the scalability benefits of the GA become apparent. However, the GA is currently more suited for the greenfield problem, and additional research is necessary for the reconfiguration. Yet, the showcased scalability and robustness could enable production reconfiguration in various novel environments.

Future research could investigate and enhance upon the application of the proposed GAs in reconfiguration scenarios, which is more related to real-world company needs. Multi-objective approaches could improve the method and further leverage the strengths of the GA approach. Furthermore, different objective functions could be investigated since the GA showcases the potential for reducing equipment or workstations. Finally, the proposed approach could be adapted to other types of assembly line balancing and constraints, like U-shaped lines.

**Author Contributions** **MA**: Conceptualization, Methodology, Implementation, Validation, Writing - original draft preparation, Writing - review & editing. **TH**: Methodology, Implementation, Validation, Writing - original draft preparation. **WK**: Writing - review & editing. **MFH**: Writing - review & editing.

**Funding** Open Access funding enabled and organized by Projekt DEAL. The research leading to these results received partial funding from the Federal Ministry For Economic Affairs And Climate Action, Germany in the research project *Verbundprojekt: Catena X Automotive Network (Catena X)*, Grant Agreement No 13IK004N.

**Availability of data and materials** The datasets used during the current study are available in the repository, https://github.com/maralbus/reconfiguration_dataset

## Declarations

**Conflict of interest** (check journal-specific guidelines for which heading to use): The authors have no competing interests to declare that are relevant to the content of this article.

**Ethics approval** Not applicable

**Consent to participate** Not applicable

**Consent for publication** Yes

**Code availability** Not applicable



## A Clusterization procedure

This chapter describes the clusterization procedure of Fig. 2b from the task precedence graph in Fig. 2a. Algorithm 1 shows the clusterization procedure used to compute the cluster precedence graph.

**Algorithm 1** Clusterization procedure

**Require:** pre(·), type(·), $n$
1: **for** $i = 1, \ldots, n$ **do**
2: $u_{i'} \leftarrow \{i\}$ ▷ cluster assignment set $u_{i'}$ with $v(i) \rightarrow i'$
3: **end for**
4: **for** $i = 1, \ldots, n$ **and** $g \in \mathcal{P}(i)$ **do**
5: **if** type$(g)$ = separation **and** type$(i)$ = handling **then**
6: $g' \leftarrow v(g)$
7: $u_{g'} \leftarrow u_{g'} \cup u_{i'}$ with $v(g) \rightarrow g'$
8: $u_{i'} \leftarrow \emptyset$
9: **end if**
10: **end for**

The algorithm initializes the mapping $v(i)$ in lines 1–2. Each task is assigned to a cluster, and $v(i)$ represents the mapping from task $i$ to cluster $i'$. We define the cluster assignment set $u_{i'}$, containing all tasks $\{i, \ldots\}$ in cluster $i'$. The loop in line 4 checks all precedence constraints $\mathcal{P}_i$. If the task pair of the precedence constraint must be performed in a single station, the clusters of the tasks are merged by updating the corresponding mappings in lines 6–8. After the algorithm terminates, some indices may yield empty sets for cluster $u_{i'}$. Therefore, we use a new notation to represent non-empty clusters: $1 \leq i' \leq n'$ with $u_{i'} \neq \emptyset \ \forall i'$. We enumerate the non-empty clusters with $i'$ and use $n'$ as the number of non-empty clusters.

## B Assignment details

A detailed description of the assigning equipment and workstations to clusters is provided next, including algorithms for both greenfield and brownfield scenarios.

### B.1 Station assignment

**Algorithm 2** Station assignment

**Require:** $n'$, $ct$, $u(\cdot)$, $ct_{\min}(\cdot)$, AssignEquipment(·)
1: $ws \leftarrow [0, \ldots, 0]^T$
2: $\bar{e} \leftarrow [0, \ldots, 0]^T$
3: $(\bar{e}^c \leftarrow [0, \ldots, 0]^T)_{\text{bf}}$ ▷ equipment counter
▷ $(\ldots)_{\text{bf}}$: only required in brownfield case
4: $idx \leftarrow 1$
5: $ws^{curr} \leftarrow (-1)$ ▷ current workstation variable
6: **while** $idx < n'$ **do**
7: $ws^{curr} \leftarrow ws^{curr} + 1$
8: $ct^{\text{estimate}} \leftarrow 0$
9: $c^{unassigned} \leftarrow \{\ \}$ ▷ set of unassigned clusters
10: **while** $ct^{\text{estimate}} \leq ct$ **and** $idx \leq n'$ **do**
11: $ws_{idx} \leftarrow ws^{curr}$
12: $c^{unassigned} \cup u_{idx}$ ▷ cluster assignment set $u_{i'}$
13: $ct^{\text{estimate}} \leftarrow ct^{\text{estimate}} + \min u_{idx}$
14: $idx \leftarrow idx + 1$
15: **end while**
16: **if** $ct^{\text{estimate}} > ct$ **then**
17: $c^{unassigned} \cap u_{idx}$
18: $idx \leftarrow idx - 1$
19: **end if**
20: numFailCluster, $\bar{e}$, $(\bar{e}^c)_{\text{bf}} \leftarrow$ AssignEquipment($c^{unassigned}$, $equipment$, $(\bar{e}^c)_{\text{bf}}$)
▷ See Algorithm 3
21: $idx \leftarrow idx -$ numFailCluster
22: **end while**

In Algorithm 2, the algorithm loops through all clusters in line 6. It then assigns as many clusters as possible to the current workstation using a lower bound of processing time spent in a specific cluster as a heuristic. If the processing time of the current workstation violates the global cycle time, the last cluster is removed from the current workstation. However, this workstation assignment is preliminary and might be modified during a subsequent iteration. After this, Algorithm 3 is called to assign pieces of equipment to the current workstation. Note that the assignment procedure is not required to assign pieces of equipment to all clusters provided, and

it returns the number of clusters for which the assignment fails. The heuristic can choose slower and cheaper equipment instead of faster but more expensive equipment, thus requiring the spread of clusters over more workstations due to cycle time constraints. The algorithm guarantees that each piece of equipment is assigned to at least one cluster to ensure termination. In line 20, the current index is set back by the number of unsuccessful clusters to force the corresponding clusters to be reconsidered by subsequent runs of the procedure. Variable vector $\bar{e}$ and equipment counter vector $\bar{e}^c$ are global variables that are modified repeatedly by the equipment assignment procedure, see Algorithm 3. The vector $\bar{e}$ contains the equipment assignments of all tasks, and $\bar{e}^c$ counts how many pieces of equipment are assigned.

### B.2 Equipment assignment

**Algorithm 3** Equipment assignment

**Require:** $u_{\cdot}$, BESTEQUIPMENT$(\cdot)$, $t_{\cdot\cdot}$, $ct$
1: **function** ASSIGNEQUIPMENT($c^{unassigned}$, $\bar{e}$, $(\bar{e}^c)_{\text{bf}}$)
▷ $(\ldots)_{\text{bf}}$: only required in brownfield case
2: numFailCluster $\leftarrow 0$
3: $(\bar{e}^{c,local} \leftarrow [0,\ldots,0]^T)_{\text{bf}}$ ▷ local equipment counter vector $\bar{e}^{c,local}$
4: $t^{unassigned} \leftarrow \{\ \}$ ▷ set of unassigned tasks
5: **for** $i_{\text{cluster}} \in c^{unassigned}$ **do** ▷ set of unassigned clusters $c^{unassigned}$
6: $t^{unassigned} \leftarrow t^{unassigned} \cup u_{i'}$
7: **end for**
8: **loop**
9: $t^{unassign,copy} \leftarrow t^{unassigned}$
10: **while** $|t^{unassign,copy}| > 0$ **do**
11: $j \leftarrow$ BESTEQUIPMENT($t^{unassign,copy}$, $(\bar{e}^c)_{\text{bf}}$)
▷ See Algorithm 4 or 5
12: $(\bar{e}^{c,local}_j \leftarrow 1)_{\text{bf}}$
13: **for** $i \in t^{unassign,copy}$ **do**
14: **if** $\bar{pt}_{i,j} \geq 0$ **then** ▷ with processing time $\bar{pt}_{i,j}$ for each task $i$ and equipment $j$.
15: $e_i \leftarrow j$
16: $t^{unassign,copy} \leftarrow t^{unassign,copy} \setminus \{i\}$
17: **end if**
18: **end for**
19: **end while**
20: $ct^{\text{current}} \leftarrow 0$
21: **for** $i \in t^{unassigned}$ **do**
22: $ct^{\text{current}} \leftarrow ct^{\text{current}} + \bar{pt}_{i,e_i}$
23: **end for**
24: **if** $ct^{\text{current}} > ct$ **then**
25: $t^{unassigned} \leftarrow t^{unassigned} \setminus i$
26: numFailCluster $\leftarrow$ numFailCluster $+1$
27: $(\bar{e}^{c,local} \leftarrow [0,\ldots,0]^T)_{\text{bf}}$
28: **else**
29: $(\bar{e}^c \leftarrow \bar{e}^c + \bar{e}^c_{local})_{\text{bf}}$
30: **return** numFailCluster, $\bar{e}$, $(\bar{e}^c)_{\text{bf}}$
31: **end if**
32: **end loop**
33: **end function**

This section will discuss the procedure for assigning equipment, shown in Algorithm 3. The process involves assigning pieces of equipment to tasks and thus substituting the clusters with the corresponding tasks in lines 4–6. The loop in line 8 attempts to assign pieces of equipment to as many tasks as possible. In lines 8–16, we assign pieces of equipment to the current set of tasks, and the best equipment is selected in line 11. The total processing time resulting from the current assignment is computed in lines 20–22.

If the assignment is valid, the function terminates. However, if the assignment is unsuccessful, the tasks in the last cluster are removed from the current set of tasks. The variable *numFailCluster* is incremented by one to indicate the cluster's removal to the cluster assignment procedure. Please note that the algorithm does not necessarily terminate since the equipment selected by the heuristic could result in a processing time violating the cycle time. To ensure termination, we can enforce faster equipment to be assigned when the assignment procedure fails for tasks of a single cluster.

Finally, a weighted sum heuristic algorithm is used, representing a cost-time-trade-off. First, a *cost score* vector $\bar{s}^c$ is calculated to estimate the expected cost of single tasks. The *cost score* normalizes the investment cost of every piece of equipment concerning the number of tasks it can perform. The computation differs for greenfield and brownfield instances due to different cost computation approaches, see Algorithms 4 and 5. Secondly, a *time score* vector $\bar{s}^t$ is calculated as the sum of time the piece of equipment needs to perform all assigned tasks. This measure is independent of the solution's total cost and is not directly related to the type, greenfield, or brownfield.

Once the two scores are computed, they are combined into a single score by the trade-off factor $\alpha \in [0, 1]$, as shown in Algorithms 4 and 5. This self-optimized trade-off parameter $\alpha$ introduces variance in the population and can encourage different equipment combinations, resulting in fewer stations. Please refer to Section "Heuristic" for more details.

## C Hyperparameter results

### C.1 Age parameter

This section performs a detailed analysis of the *age* hyperparameter. We evaluated the performance of the proposed GA algorithm for *age* values in {100, 200, 300, 400, 500, 600, 700, 800, 1000} to evaluate the robustness against different *age* criteria. The number of generations until the GA reaches convergence is also measured for the greenfield and brownfield cases. The values are provided in Table 5. For this evaluation, the GA is averaged over ten runs for every instance and every equipment alternative $r$ in {5, 10, 15, 20}.

**Algorithm 4** Greenfield equipment selection heuristic

**Require:** $r$, $\bar{pt}_{..}$, $ec_{.}$, $\alpha$
**function** BESTEQUIPMENT($t^{unassigned}$)
  $\bar{e}^t \leftarrow [0, \ldots, 0]^T$ ▷ tasks per equipment vector $\bar{e}^t$
  $\bar{s}^t \leftarrow [0, \ldots, 0]^T$ ▷ time score vector $\bar{s}^t$
  $\bar{s}^c \leftarrow [0, \ldots, 0]^T$ ▷ cost score vector $\bar{s}^c$
  **for** $i \in t^{unassigned}$ **do**
    **for** $j = 1, \ldots, r$ **do**
      **if** $\bar{pt}_{i,j} \geq 0$ **then** ▷ processing costs vector $\bar{pt}_{i,j}$ per task $i$ and equipment $j$
        $\bar{e}^t \leftarrow \bar{e}^t + 1$
        $\bar{s}^t_j \leftarrow \bar{s}^t_j + pt_{i,j}$
      **end if**
    **end for**
  **end for**
  **for** $j = 1, \ldots, r$ **do**
    **if** $\bar{e}^t_j > 0$ **then**
      $\bar{s}^c_j \leftarrow ec_j \,/\bar{e}^t$
      $\bar{s}^t_j \leftarrow \bar{s}^t_j \,/\bar{e}^t$
    **else**
      $\bar{s}^c_j \leftarrow (-1)$
      $\bar{s}^t_j \leftarrow (-1)$
    **end if**
  **end for**
  **for** $j = 1, \ldots, r$ **do**
    **if** $\bar{e}^t_j > 0$ **then**
      $\bar{s}^c_j \leftarrow \bar{s}^c_j \,/ \max \bar{s}^c$
      $\bar{s}^t_j \leftarrow \bar{s}^t_j \,/ \max \bar{s}^t$
    **end if**
  **end for**
  $\bar{s} \leftarrow [\infty, \ldots, \infty]^T$ ▷ score vector $\bar{s}$
  **for** $j = 1, \ldots, r$ **do**
    **if** $\bar{e}^t_j > 0$ **then**
      $s_j \leftarrow (1-\alpha) \cdot \bar{s}^c_j + \alpha \cdot \bar{s}^t_j$ ▷ weighted score per equipment $j$
    **end if**
  **end for**
  **return** $\arg\min_{1 \leq l \leq r} s_j$, s.t. $s_j \geq 0$
**end function**

**Algorithm 5** Brownfield equipment selection heuristic

**Require:** $r$, $t_{..}$, $d_{.}$, $pc_{.}$, $sc_{.}$, $ec_{.}$, $\alpha$
**function** BESTEQUIPMENT($t^{unassigned}$, $\hat{e}$)
  $\bar{e}^t \leftarrow [0, \ldots, 0]^T$ ▷ tasks per equipment vector $\bar{e}^t$
  $\bar{s}^t \leftarrow [0, \ldots, 0]^T$ ▷ time score vector $\bar{s}^t$
  $\bar{s}^c \leftarrow [0, \ldots, 0]^T$ ▷ cost score vector $\bar{s}^c$
  **for** $i \in t^{unassigned}$ **do**
    **for** $j = 1, \ldots, r$ **do**
      **if** $\bar{pt}_{i,j} \geq 0$ **then** ▷ processing costs vector $\bar{pt}_{i,j}$ per task $i$ and equipment $j$
        $\bar{e}^t \leftarrow \bar{e}^t + 1$
        $\bar{s}^t_j \leftarrow \bar{s}^t_j + \bar{pt}_{i,j}$
      **end if**
    **end for**
  **end for**
  **for** $j = 1, \ldots, r$ **do**
    **if** $\bar{e}^t_j > 0$ **then**
      **if** $d_j - \hat{e}_j > 0$ **then** ▷ amount of equipment $j$ in the existing line as $d_j$
        $\bar{s}^c_j \leftarrow pc_j + sc_j \bar{e}^t$
      **else**
        $\bar{s}^c_j \leftarrow pc_j + ec_j \bar{e}^t$
      **end if**
      $\bar{s}^t_j \leftarrow \bar{s}^t_j \,/\bar{e}^t$
    **else**
      $\bar{s}^c_j \leftarrow (-1)$
      $\bar{s}^t_j \leftarrow (-1)$
    **end if**
  **end for**
  **for** $j = 1, \ldots, r$ **do**
    **if** $\bar{e}^t_j > 0$ **then**
      $\bar{s}^c_j \leftarrow \bar{s}^c_j \,/ \max \bar{s}^c$
      $\bar{s}^t_j \leftarrow \bar{s}^t_j \,/ \max \bar{s}^t$
    **end if**
  **end for**
  $\bar{s} \leftarrow [\infty, \ldots, \infty]^T$ ▷ score vector $\bar{s}$
  **for** $j = 1, \ldots, r$ **do**
    **if** $\bar{e}^t_j > 0$ **then**
      $s_j \leftarrow (1-\alpha) \cdot \bar{s}^c_j + \alpha \cdot \bar{s}^t_j$ ▷ weighted score per equipment $j$
    **end if**
  **end for**
  **return** $\arg\min_{1 \leq l \leq r} s_j$, s.t. $s_j \geq 0$
**end function**

The performance is compared based on the total costs $C$ and the computational runtime $T$ against the IP baseline.

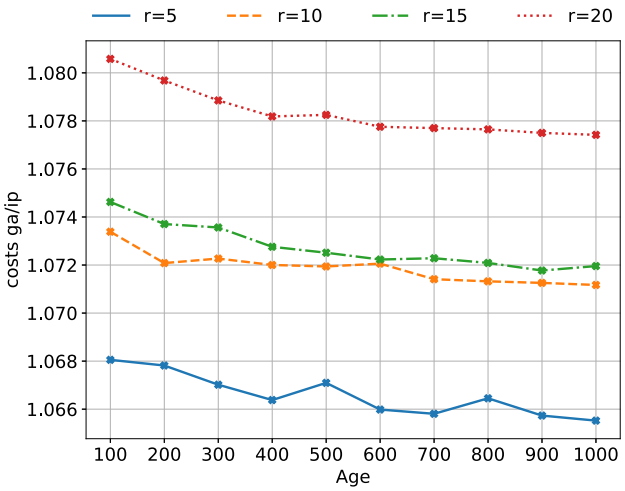
It becomes apparent in Fig. 5c and 5d that the algorithm scales linearly with the increase in *age*. A significant increase in performance is observed for the greenfield instances up to *age* values of 400 and 500, with a performance drop for the $r = 5$ instances, see Fig. 5a. To allow for more optimization generations while sacrificing computational runtime, an *age* value of 500 is appropriate. All other instances ($r = \{10, 15, 20\}$) showcase a shallow performance decrease after the value 500, which indicates a sweet spot between early termination and computation time reduction since every increase in age directly correlates with more computational runtime.

The brownfield performance is relatively constant towards changes in the *age* value, and the same linear scaling can be seen in Figs. 5c and 5d. As a result, a significantly lower *age* value could be chosen. However, this would decrease the greenfield performance. An *age* value of 500 was chosen to ensure sufficient generations to highlight the algorithm's capabilities on the total cost while maintaining its computation time reduction benefit,The maximum convergence generations ranging from 203.8 ($r = 5$) to 328.6 ($r = 20$) for the greenfield and 224.5 ($r = 5$) to 263.4 ($r = 20$) for the brownfield case confirm this selection. Together with the findings of Appendix C Section "Population size parameter", this *age* value holds the highest potential for scalability in the RALBP-TT problem setting. The *age* hyperparameter comparison results are provided in Table 6.

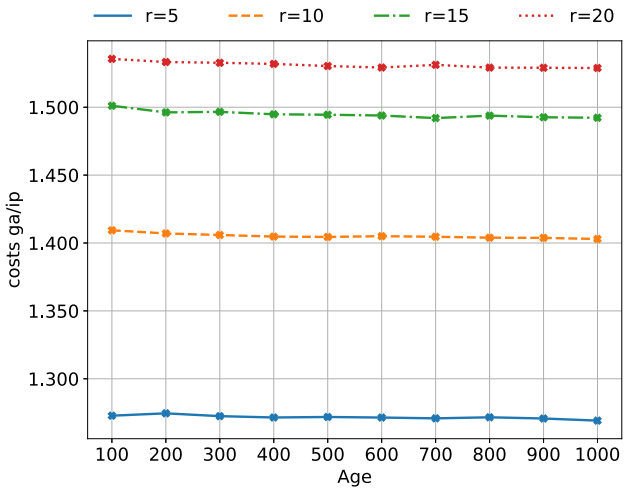
**Table 5** Average minimum, maximum, and mean number of generations until the GA reaches convergence.

| Objective | r | Avg | Min | Max |
|---|---|---|---|---|
| Greenfield | 5 | 82.04 | 23.80 | 203.80 |
| | 10 | 125.05 | 37.81 | 303.43 |
| | 15 | 118.75 | 33.72 | 293.86 |
| | 20 | 138.67 | 40.62 | 328.60 |
| Brownfield | 5 | 87.66 | 27.69 | 224.46 |
| | 10 | 106.13 | 27.64 | 260.17 |
| | 15 | 103.07 | 26.90 | 249.94 |
| | 20 | 111.07 | 31.92 | 263.43 |

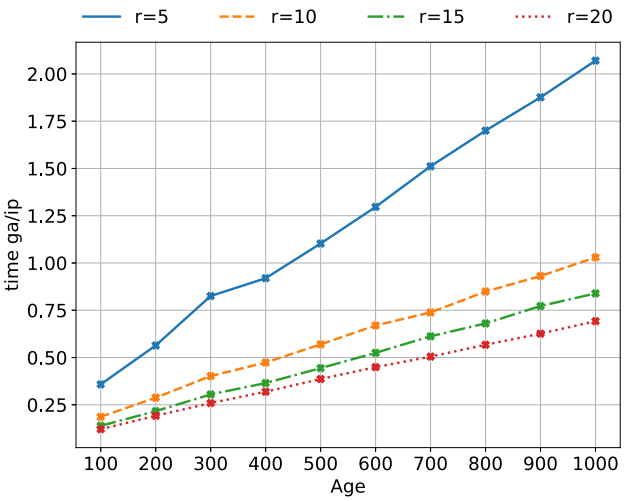
The number of generations for the greenfield and brownfield objectives is shown for all equipment alternatives



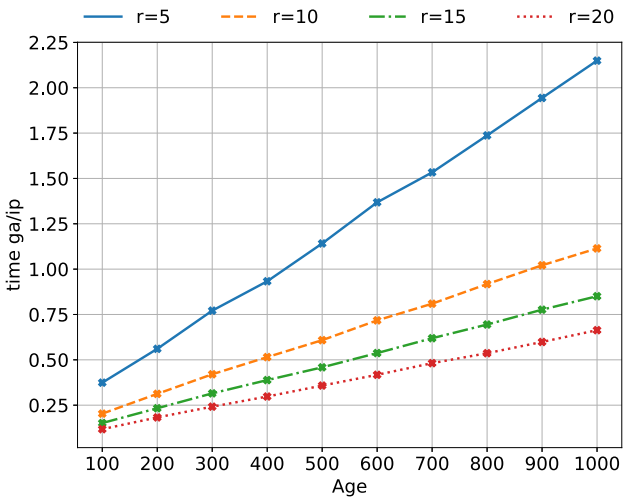

(a) Greenfield cost ratio comparison.




(b) Brownfield cost ratio comparison.




(c) Greenfield time ratio comparison.




(d) Brownfield time ratio comparison.

**Fig. 5** Cost and time comparison plots for the hyperparameter *age*. Compared are the *avg* GA values of ten runs over the IP values for each instance, averaged for each equipment alternative $r$ in {5, 10, 15, 20}. The hyperparameter *age* values are shown on the x-axis. The metrics ratio is shown on the y-axis. Depicted metrics ratios include total costs $C$ for greenfield and brownfield in Fig. 5a and 5b, and computational runtime $T$ in Fig. 5c and 5d

**Table 6** Experiment results for the evaluation metrics total costs $C$, and computational runtime $T$ on the hyperparameter *age* for problem instance with all available equipment alternatives and the greenfield and brownfield objective

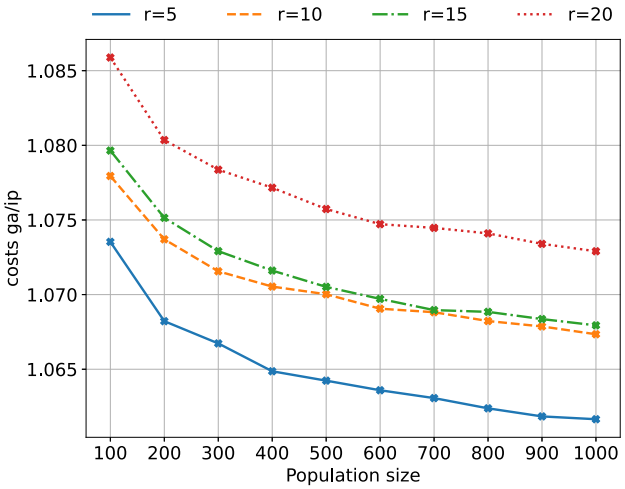
| | | r | Age values | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
| Greenfield | $C^{ga}/C^{ip}$ | 5 | 1.068 | 1.068 | 1.067 | 1.066 | 1.067 | 1.066 | 1.066 | 1.066 | 1.066 | **1.066** |
| | | 10 | 1.073 | 1.072 | 1.072 | 1.072 | 1.072 | 1.072 | 1.071 | 1.071 | 1.071 | **1.071** |
| | | 15 | 1.075 | 1.074 | 1.074 | 1.073 | 1.073 | 1.072 | 1.072 | 1.072 | **1.072** | 1.072 |
| | | 20 | 1.081 | 1.080 | 1.079 | 1.078 | 1.078 | 1.078 | 1.078 | 1.078 | 1.077 | **1.077** |
| Greenfield | $T^{ga}/T^{ip}$ | 5 | **0.358** | 0.564 | 0.825 | 0.919 | 1.103 | 1.297 | 1.512 | 1.700 | 1.876 | 2.070 |
| | | 10 | **0.186** | 0.288 | 0.402 | 0.474 | 0.570 | 0.670 | 0.739 | 0.849 | 0.931 | 1.029 |
| | | 15 | **0.139** | 0.216 | 0.305 | 0.365 | 0.444 | 0.525 | 0.612 | 0.680 | 0.772 | 0.839 |
| | | 20 | **0.121** | 0.192 | 0.259 | 0.319 | 0.386 | 0.449 | 0.505 | 0.568 | 0.626 | 0.692 |
| Brownfield | $C^{ga}/C^{ip}$ | 5 | 1.273 | 1.275 | 1.272 | 1.271 | 1.272 | 1.271 | 1.271 | 1.272 | 1.271 | **1.269** |
| | | 10 | 1.409 | 1.407 | 1.406 | 1.405 | 1.404 | 1.405 | 1.405 | 1.404 | 1.404 | **1.403** |
| | | 15 | 1.501 | 1.496 | 1.497 | 1.495 | 1.494 | 1.494 | **1.492** | 1.494 | 1.493 | 1.492 |
| | | 20 | 1.536 | 1.533 | 1.533 | 1.532 | 1.530 | 1.529 | 1.531 | 1.529 | 1.529 | **1.529** |
| Brownfield | $T^{ga}/T^{ip}$ | 5 | **0.374** | 0.561 | 0.771 | 0.933 | 1.142 | 1.368 | 1.533 | 1.737 | 1.944 | 2.149 |
| | | 10 | **0.203** | 0.312 | 0.420 | 0.515 | 0.608 | 0.717 | 0.809 | 0.918 | 1.021 | 1.114 |
| | | 15 | **0.151** | 0.233 | 0.315 | 0.388 | 0.459 | 0.537 | 0.619 | 0.695 | 0.776 | 0.851 |
| | | 20 | **0.118** | 0.183 | 0.242 | 0.298 | 0.358 | 0.417 | 0.482 | 0.537 | 0.599 | 0.663 |

Shown are the averaged $avg$ GA to IP greenfield and brownfield ratios. Highlighted are the best results in each row

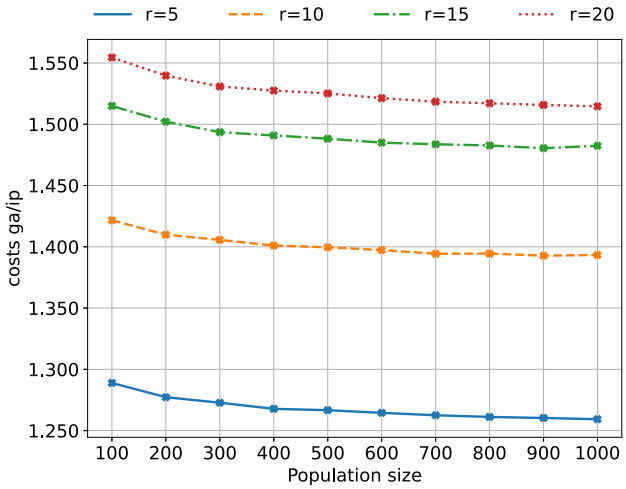
**Table 7** Experiment results for the evaluation metrics total costs $C$, and computational runtime $T$ on the hyperparameter *population size* for problem instance with all available equipment alternatives and the greenfield and brownfield objective

| | | r | Population size values | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
| Greenfield | $C^{ga}/C^{ip}$ | 5 | 1.074 | 1.068 | 1.067 | 1.065 | 1.064 | 1.064 | 1.063 | 1.062 | 1.062 | **1.062** |
| | | 10 | 1.078 | 1.074 | 1.072 | 1.071 | 1.070 | 1.069 | 1.069 | 1.068 | 1.068 | **1.067** |
| | | 15 | 1.080 | 1.075 | 1.073 | 1.072 | 1.071 | 1.070 | 1.069 | 1.069 | 1.068 | **1.068** |
| | | 20 | 1.086 | 1.080 | 1.078 | 1.077 | 1.076 | 1.075 | 1.074 | 1.074 | 1.073 | **1.073** |
| Greenfield | $T^{ga}/T^{ip}$ | 5 | **0.761** | 0.940 | 1.305 | 1.687 | 2.030 | 2.405 | 2.719 | 3.151 | 3.495 | 3.939 |
| | | 10 | **0.226** | 0.400 | 0.569 | 0.742 | 0.910 | 1.104 | 1.294 | 1.521 | 1.655 | 1.818 |
| | | 15 | **0.186** | 0.330 | 0.447 | 0.594 | 0.719 | 0.855 | 1.002 | 1.147 | 1.281 | 1.453 |
| | | 20 | **0.155** | 0.273 | 0.384 | 0.502 | 0.610 | 0.736 | 0.854 | 0.999 | 1.123 | 1.242 |
| Brownfield | $C^{ga}/C^{ip}$ | 5 | 1.289 | 1.277 | 1.273 | 1.268 | 1.267 | 1.265 | 1.263 | 1.261 | 1.260 | **1.259** |
| | | 10 | 1.422 | 1.410 | 1.406 | 1.401 | 1.400 | 1.397 | 1.394 | 1.394 | **1.393** | 1.393 |
| | | 15 | 1.515 | 1.502 | 1.494 | 1.491 | 1.488 | 1.485 | 1.484 | 1.483 | **1.480** | 1.482 |
| | | 20 | 1.554 | 1.540 | 1.531 | 1.528 | 1.525 | 1.521 | 1.518 | 1.517 | 1.516 | **1.515** |
| Brownfield | $T^{ga}/T^{ip}$ | 5 | **0.640** | 1.001 | 1.341 | 1.753 | 2.115 | 2.461 | 2.832 | 3.190 | 3.564 | 3.958 |
| | | 10 | **0.245** | 0.435 | 0.620 | 0.824 | 0.988 | 1.181 | 1.382 | 1.575 | 1.776 | 1.958 |
| | | 15 | **0.185** | 0.326 | 0.468 | 0.617 | 0.748 | 0.896 | 1.044 | 1.188 | 1.345 | 1.474 |
| | | 20 | **0.144** | 0.254 | 0.365 | 0.475 | 0.579 | 0.693 | 0.807 | 0.929 | 1.039 | 1.160 |

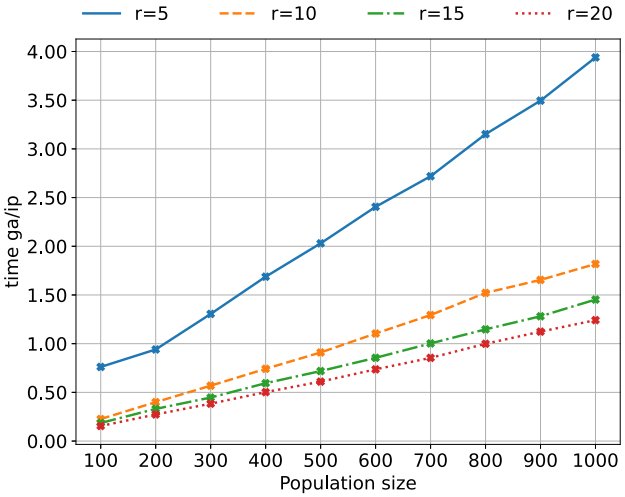
Shown are the averaged $avg$ GA to IP greenfield and brownfield ratios. Highlighted are the best results in each row



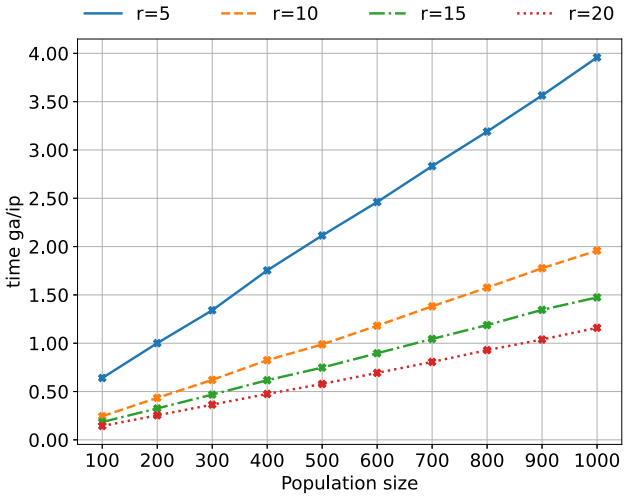

(a) Greenfield cost ratio comparison.




(b) Brownfield cost ratio comparison.




(c) Greenfield time ratio comparison.




(d) Brownfield time ratio comparison.

**Fig. 6** Cost and time comparison plots for the hyperparameter *population size*. Compared are the *avg* GA values of ten runs over the IP values for each instance, averaged for each equipment alternative $r$ in $\{5, 10, 15, 20\}$. The hyperparameter *population size* values are shown on the x-axis. The metrics ratio is shown on the y-axis. Depicted metrics ratios include total costs $C$ for greenfield and brownfield in Figs. 6a and 6b, and computational runtime $T$ in Figs. 6c and 6d

## C.2 Population size parameter

Next, a detailed analysis of the *population size* hyperparameter is performed. Since the GA operates on populations with a fixed population size, this many population triples $(\underline{u}, \underline{w}, \underline{q})$ are considered at once.

The experiments on the *population size* hyperparameter showcases the significant correlation between computation time and increased *population size*. The increased computation time is, at worst, $3.94\times$ $(r = 5)$ slower than the IP baseline for the greenfield objective and $3.96\times$ $(r = 5)$ for the brownfield objective. In comparison with the *age* hyperparameter scaling, which is $2.07\times$ $(r = 5)$ for the greenfield and $2.15\times$ $(r = 5)$ brownfield objective, the *population size* has a more relevant influence on the computational runtime. The *population size* hyperparameter comparison results are provided in Table 7.

An issue comes from evaluating the greenfield total costs $C$ results, shown in Fig. 6a. The results show a performance increase for all equipment alternatives $r$ with increased *population size*. This result may be explained by the fact that the greenfield objective requires only sequential planning from scratch without incorporating existing assembly lines. Thus, increasing the *population size* expands the search space, leading to better overall costs. These findings are only valid for the greenfield objective. In contrast, an increase in *population size* has a less relevant influence on the total costs $C$ for the brownfield objective, shown in Fig. 6b. As a result, the

*population size* is set to 300 to exploit the runtime benefits of lower *population sizes*. The decrease in computational runtime is compensated with an increased *age* parameter due to the better scaling in runtime and performance benefits for a higher number of equipment alternatives.

## References


Albus, M., & Huber, M. F. (2023). Resource reconfiguration and optimization in brownfield constrained robotic assembly line balancing problems. *Journal of Manufacturing Systems, 67*, 132–142. https://doi.org/10.1016/j.jmsy.2023.01.001

Albus, M., & Seeber, C. (2021). Linear optimization for dynamic selection of resources in constrained assembly line balancing problems. *Procedia CIRP, 104*, 134–139. https://doi.org/10.1016/j.procir.2021.11.023

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solutionapproaches. *International Journal of Production Economics, 142*(2), 259–277. https://doi.org/10.1016/j.ijpe.2012.10.020

Baybars, I. (1986). Survey of exact algorithms for the simple assembly line balancing problem. *Management Science, 32*(8), 909–932. https://doi.org/10.1287/mnsc.32.8.909

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715. https://doi.org/10.1016/j.ejor.2004.07.023

Blum, C. (2008). Beam-ACO for simple assembly line balancing. *INFORMS Journal on Computing, 20*(4), 618–627. https://doi.org/10.1287/ijoc.1080.0271

Borba, L., Ritt, M., & Miralles, C. (2018). Exact and heuristic methods for solving the robotic assembly line balancing problem. *European Journal of Operational Research, 270*(1), 146–156. https://doi.org/10.1016/j.ejor.2018.03.011

Boysen, N., Fliedner, M., & Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics, 111*(2), 509–528. https://doi.org/10.1016/j.ijpe.2007.02.026

Boysen, N., Schulze, P., & Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research, 301*, 797–814. https://doi.org/10.1016/J.EJOR.2021.11.043

Bukchin, J., Dar-El, E. M., & Rubinovitz, J. (2002). Mixed model assembly line design in a make-to-order environment. *Computers and Industrial Engineering, 41*(4), 405–421. https://doi.org/10.1016/s0360-8352(01)00065-1

Bukchin, J., & Rubinovitz, J. (2003). A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions (Institute of Industrial Engineers), 35*(1), 73–85. https://doi.org/10.1080/07408170304429

Bukchin, J., & Tzur, M. (2000). Design of flexible assembly line to minimize equipment cost. *IIE Transactions (Institute of Industrial Engineers), 32*(7), 585–598. https://doi.org/10.1080/07408170008967418

Capacho, L., Pastor, R., Guschinskaya, O., & Dolgui, A. (2006). Heuristic methods to solve the alternative subgraphs assembly line balancing problem. In: IEEE International Conference on Automation Science and Engineering (CASE), pp. 501–506. https://doi.org/10.1109/COASE.2006.326932.

Chakravarty, A. K., & Shtub, A. (1986). A cost minimization procedure for mixed model production lines with normally distributed task times. *European Journal of Operational Research, 23*(1), 25–36. https://doi.org/10.1016/0377-2217(86)90211-0

Chutima, P. (2022). A comprehensive review of robotic assembly line balancing problem. *Journal of Intelligent Manufacturing, 33*(1), 1–34. https://doi.org/10.1007/s10845-020-01641-7

Çil, Z. A., Mete, S., & Agpak, K. (2017). Analysis of the type ii robotic mixed-model assembly line balancing problem. *Engineering Optimization, 49*(6), 990–1009. https://doi.org/10.1080/0305215X.2016.1230208

Çimen, T., Baykasoğlu, A., & Demirkol Akyol, S. (2022). A detailed review and analysis of assembly line rebalancing problems. *Assembly Automation, 42*(6), 742–760. https://doi.org/10.1108/AA-02-2022-0031

Daoud, S., Chehade, H., Yalaoui, F., & Amodeo, L. (2014). Solving a robotic assembly line balancing problem using efficient hybrid methods. *Journal of Heuristics, 20*(3), 235–259. https://doi.org/10.1007/s10732-014-9239-0

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In: Proc. of IJCAI-85, pp. 162–164.

Dolgui, A., & Ihnatsenka, I. (2009). Branch and bound algorithm for a transfer line design problem: Stations with sequentially activated multi-spindle heads. *European Journal of Operational Research, 197*(3), 1119–1132. https://doi.org/10.1016/j.ejor.2008.03.028

Erel, E., Sabuncuoglu, I., & Sekerci, H. (2005). Stochastic assembly line balancing using beam search. *International Journal of Production Research, 43*(7), 1411–1426. https://doi.org/10.1080/00207540412331320526

Falkenauer, E. (2005). Line balancing in the real world. *Proceedings of the International Conference on Product Lifecycle Management PLM, 5*, 360–370.

Fisel, J., Arslan, A., & Lanza, G. (2017). Changeability Focused Planning Method for Multi Model Assembly Systems in Automotive Industry. *Procedia CIRP, 63*, 515–520. https://doi.org/10.1016/j.procir.2017.03.148

Gansterer, M., & Hartl, R. F. (2018). One-and two-sided assembly line balancing problems with real-world constraints. *International Journal of Production Research, 56*(8), 3025–3042. https://doi.org/10.1080/00207543.2017.1394599

Gao, J., Sun, L., Wang, L., & Gen, M. (2009). An efficient approach for type ii robotic assembly line balancing problems. *Computers and Industrial Engineering, 56*(3), 1065–1080. https://doi.org/10.1016/j.cie.2008.09.027

Goldberg, D.E., Lingle, R. (1985). Others: Alleles, loci, and the traveling salesman problem. In: Proceedings of an International Conference on Genetic Algorithms and Their Applications, vol. 154, pp. 154–159. Lawrence Erlbaum Hillsdale, NJ

Greinacher, S., Overbeck, L., Kuhnle, A., Krahe, C., & Lanza, G. (2020). Multi-objective optimization of lean and resource efficient manufacturing systems. *Production Engineering, 14*, 165–176. https://doi.org/10.1007/S11740-019-00945-9/FIGURES/7

Gupta, H., Kumar, A., & Wasan, P. (2021). Industry 4.0, cleaner production and circular economy: An integrative framework for evaluating ethical and sustainable business performance of manufacturing organizations. *Journal of Cleaner Production, 295*, 126253. https://doi.org/10.1016/j.jclepro.2021.126253

Hamzadayi, A., & Yildiz, G. (2012). A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model u-lines with parallel workstations and zoning constraints. *Computers and Industrial Engineering, 62*(1), 206–215. https://doi.org/10.1016/j.cie.2011.09.008

Huo, J., Wang, Z., Chan, F. T. S., Lee, C. K. M., & Strandhagen, J. O. (2018). Assembly line balancing based on beam ant colony optimisation. *Mathematical Problems in Engineering*. https://doi.org/10.1155/2018/2481435

Javaid, M., Haleem, A., Singh, R. P., & Suman, R. (2021). Substantial capabilities of robotics in enhancing industry 4.0 implementation. *Cognitive Robotics, 1*, 58–75. https://doi.org/10.1016/j.cogr.2021.06.001

Ji, W., & Wang, L. (2019). Industrial robotic machining: a review. *International Journal of Advanced Manufacturing Technology, 103*(1–4), 1239–1255. https://doi.org/10.1007/s00170-019-03403-z

Kammer Christensen, M., Janardhanan, M. N., & Nielsen, P. (2017). Heuristics for solving a multi-model robotic assembly line balancing problem. *Production & Manufacturing Research, 5*(1), 410–424. https://doi.org/10.1080/21693277.2017.1403977

Koren, Y. (2010). *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems* (Vol. 80). Hoboken, NJ: John Wiley & Sons. https://doi.org/10.1002/9780470618813

Koren, Y., & Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems, 29*(4), 130–141. https://doi.org/10.1016/j.jmsy.2011.01.001

Kucukkoc, I., Buyukozkan, K., Satoglu, S. I., & Zhang, D. Z. (2019). A mathematical model and artificial bee colony algorithm for the lexicographic bottleneck mixed-model assembly line balancing problem. *Journal of Intelligent Manufacturing, 30*(8), 2913–2925. https://doi.org/10.1007/s10845-015-1150-5

Kucukkoc, I., & Zhang, D. Z. (2016). Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines. *International Journal of Advanced Manufacturing Technology, 82*(1–4), 265–285. https://doi.org/10.1007/s00170-015-7320-y

Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*(3), 811–825. https://doi.org/10.1016/j.ejor.2004.07.030

Li, Z., Janardhanan, M. N., & Ponnambalam, S. G. (2021). Cost-oriented robotic assembly line balancing problem with setup times: Multi-objective algorithms. *Journal of Intelligent Manufacturing, 32*(4), 989–1007. https://doi.org/10.1007/s10845-020-01598-7

Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (2018). Mathematical model and metaheuristics for simultaneous balancing and sequencing of a robotic mixed-model assembly line. *Engineering Optimization, 50*(5), 877–893. https://doi.org/10.1080/0305215X.2017.1351963

Li, Z., Kucukkoc, I., & Tang, Q. (2020). A comparative study of exact methods for the simple assembly line balancing problem. *Soft Computing, 24*(15), 11459–11475. https://doi.org/10.1007/s00500-019-04609-9

Li, Y., Li, Z., & Saldanha-da-Gama, F. (2021). New approaches for rebalancing an assembly line with disruptions. *International Journal of Computer Integrated Manufacturing*. https://doi.org/10.1080/0951192X.2021.1925967

Lopes, T. C., Sikora, C. G. S., Molina, R. G., Schibelbain, D., Rodrigues, L. C. A., & Magatão, L. (2017). Balancing a robotic spot welding manufacturing line: An industrial case study. *European Journal of Operational Research, 263*(3), 1033–1048. https://doi.org/10.1016/j.ejor.2017.06.001

Lotter, B., & Wiendahl, H.-P. (2012). *Montage in der Industriellen Produktion: Ein Handbuch Für die Praxis* (p. 501350). Heidelberg: Springer.

Makssoud, F., Battaïa, O., & Dolgui, A. (2014). An exact optimization approach for a transfer line reconfiguration problem. *International Journal of Advanced Manufacturing Technology, 72*(5–8), 717–727. https://doi.org/10.1007/s00170-014-5694-x

Makssoud, F., Battaïa, O., Dolgui, A., Mpofu, K., & Olabanji, O. (2015). Re-balancing problem for assembly lines: New mathematical model and exact solution method. *Assembly Automation, 35*(1), 16–21. https://doi.org/10.1108/AA-07-2014-061

Morrison, D. R., Sewell, E. C., & Jacobson, S. H. (2014). An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset. *European Journal of Operational Research, 236*(2), 403–409. https://doi.org/10.1016/j.ejor.2013.11.033

Napoleone, A., Pozzetti, A., Macchi, M., & Andersen, R. (2021). Time to be responsive in the process industry: a literature-based analysis of trends of change, solutions and challenges. *Production Planning and Control*. https://doi.org/10.1080/09537287.2021.1942282

Nicosia, G., Pacciarelli, D., & Pacifici, A. (2002). Optimally balancing assembly lines with different workstations. *Discrete Applied Mathematics, 118*(1–2), 99–113. https://doi.org/10.1016/S0166-218X(01)00259-1

Nilakantan, J. M., Nielsen, I., Ponnambalam, S. G., & Venkataramanaiah, S. (2017). Differential evolution algorithm for solving RALB problem using cost- and time-based models. *International Journal of Advanced Manufacturing Technology, 89*(1–4), 311–332. https://doi.org/10.1007/s00170-016-9086-2

Nilakantan, J. M., & Ponnambalam, S. G. (2016). Robotic u-shaped assembly line balancing using particle swarm optimization. *Engineering Optimization, 48*(2), 231–252. https://doi.org/10.1080/0305215X.2014.998664

Nilakantan, J. M., Ponnambalam, S. G., Jawahar, N., & Kanagaraj, G. (2015). Bio-inspired search algorithms to solve robotic assembly line balancing problems. *Neural Computing and Applications, 26*(6), 1379–1393. https://doi.org/10.1007/s00521-014-1811-x

Oesterle, J., & Amodeo, L. (2016). Comparison of multiobjective algorithms for the assembly line balancing design problem. *IFAC-PapersOnLine, 49*(12), 313–318. https://doi.org/10.1016/j.ifacol.2016.07.623

Oesterle, J., Amodeo, L., & Yalaoui, F. (2019). A comparative study of multi-objective algorithms for the assembly line balancing and equipment selection problem under consideration of product design alternatives. *Journal of Intelligent Manufacturing, 30*(3), 1021–1046. https://doi.org/10.1007/s10845-017-1298-2

Ogan, D., & Azizoglu, M. (2015). A branch and bound method for the line balancing problem in u-shaped assembly lines with equipment requirements. *Journal of Manufacturing Systems, 36*, 46–54. https://doi.org/10.1016/j.jmsy.2015.02.007

Oliveira, F. S., Vittori, K., Russel, R. M. O., & Travassos, X. L. (2012). Mixed assembly line rebalancing: A binary integer approach applied to real world problems in the automotive industry. *International Journal of Automotive Technology, 13*(6), 933–940. https://doi.org/10.1007/s12239-012-0094-4

Öztürk, C., Tunali, S., Hnich, B., & Örnek, M. A. (2013). Balancing and scheduling of flexible mixed model assembly lines. *Constraints, 18*(3), 434–469. https://doi.org/10.1007/s10601-013-9142-6

Pereira, J., Ritt, M., & Vásquez, Ó. C. (2018). A memetic algorithm for the cost-oriented robotic assembly line balancing problem. *Computers and Operations Research, 99*, 249–261. https://doi.org/10.1016/j.cor.2018.07.001

Piller, F.T. (2001). Kundenindividuelle Massenproduktion (Mass Customization), pp. 200–266. Deutscher Universitätsverlag, Wiesbaden. https://doi.org/10.1007/978-3-322-92337-0_7

Psarommatis, F., & Kiritsis, D. (2019). Identification of the Inspection Specifications for Achieving Zero Defect Manufacturing. *IFIP Advances in Information and Communication Technology, 566*, 267–273. https://doi.org/10.1007/978-3-030-30000-5_34

Psarommatis, F., Prouvost, S., May, G., & Kiritsis, D. (2020). Product Quality Improvement Policies in Industry 4.0: Characteristics, Enabling Factors, Barriers, and Evolution Toward Zero Defect manufacturing. *Frontiers in Computer Science, 2*, 26. https://doi.org/10.3389/fcomp.2020.00026

Psarommatis, F., Sousa, J., Mendonça, J. P., & Kiritsis, D. (2022). Zero-defect manufacturing the approach for higher manufacturing sustainability in the era of industry 4.0: a position paper. *International Journal of Production Research, 60*(1), 73–91. https://doi.org/10.1080/00207543.2021.1987551

Richtlinie, V. D. I. (1990). *VDI Richtlinie 2860: Montage-und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole*. Düsseldorf: VDI-Verlag.

Rubinovitz, J., Bukchin, J., & Lenz, E. (1993). Ralb - a heuristic algorithm for design and balancing of robotic assembly lines. *CIRP Annals - Manufacturing Technology, 42*(1), 497–500. https://doi.org/10.1016/S0007-8506(07)62494-9

Rubinovitz, J., & Levitin, G. (1995). Genetic algorithm for assembly line balancing. *International Journal of Production Economics, 41*(1–3), 343–354. https://doi.org/10.1016/0925-5273(95)00059-3

Scholl, A., Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. In: European Journal of Operational Research, vol. 168, pp. 666–693. https://doi.org/10.1016/j.ejor.2004.07.022

Scholl, A., Fliedner, M., & Boysen, N. (2010). Absalom: Balancing assembly lines with assignment restrictions. *European Journal of Operational Research, 200*(3), 688–701. https://doi.org/10.1016/j.ejor.2009.01.049

Sewell, E. C., & Jacobson, S. H. (2012). A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing, 24*(3), 433–442. https://doi.org/10.1287/ijoc.1110.0462

Touzout, F. A., & Benyoucef, L. (2019). Multi-objective sustainable process plan generation in a reconfigurable manufacturing environment: exact and adapted evolutionary approaches. *International Journal of Production Research, 57*(8), 2531–2547. https://doi.org/10.1080/00207543.2018.1522006

Vilà, M., & Pereira, J. (2014). A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers and Operations Research, 44*, 105–114. https://doi.org/10.1016/j.cor.2013.10.016

Wiendahl, H.-P., Große-Heitmeyer, V., Mühlenbruch, H., Keunecke, L., & Geiger, M. (2004). Variantenbeherrschung in der Montage Konzept und Praxis der Flexiblen Produktionsendstufe, pp. 1–320. Springer, Heidelberg

Yang, C., Gao, J., & Sun, L. (2013). A multi-objective genetic algorithm for mixed-model assembly line rebalancing. *Computers and Industrial Engineering, 65*(1), 109–116. https://doi.org/10.1016/j.cie.2011.11.033

Yelles-Chaouche, A.R., Gurevsky, E., Brahimi, N., & Dolgui, A. (2020). Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. International Journal of Production Research (2020) https://doi.org/10.1080/00207543.2020.1813913

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., & Ardakani, H. D. (2012). Type ii robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. *Journal of Manufacturing Systems, 31*(2), 139–151. https://doi.org/10.1016/j.jmsy.2011.10.002

Zhang, Y., Hu, X., Wu, C. (2018). Heuristic algorithm for type ii two-sided assembly line rebalancing problem with multi-objective. In: MATEC Web of Conferences, vol. 175, p. 03063. https://doi.org/10.1051/matecconf/201817503063.

Zhang, Y., Hu, X., & Wu, C. (2018). A modified multi-objective genetic algorithm for two-sided assembly line re-balancing problem of a shovel loader. *International Journal of Production Research, 56*(9), 3043–3063. https://doi.org/10.1080/00207543.2017.1402136

Zhang, Z., Tang, Q., Li, Z., & Zhang, L. (2019). Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems. *International Journal of Production Research, 57*(17), 5520–5537. https://doi.org/10.1080/00207543.2018.1530479