

# ULRR

## Embedding optimization with deterministic discrete event simulation for assignment of cross-trained operators: an assembly line case study

Item Type	Article
Authors	Dagkakis, Georgios;Rotondo, Anna;Heavey, Cathal
Citation	Computers and Operations Research;111, pp. 99-115
Publisher	Elsevier
Download date	2026-06-08 17:16:00
Item License	<a href="https://creativecommons.org/licenses/by-nc-sa/1.0/">https://creativecommons.org/licenses/by-nc-sa/1.0/</a>
Link to Item	<a href="https://hdl.handle.net/10344/8389">https://hdl.handle.net/10344/8389</a>

# Embedding Optimization with Deterministic Discrete Event Simulation for Assignment of Cross-trained Operators: An Assembly Line Case Study

Georgios Dagkakis<sup>a</sup>, Anna Rotondo<sup>b</sup>, Cathal Heavey<sup>c,\*</sup>

<sup>a</sup>*Nexedi SA, 147 Rue du Ballon, 59110 La Madeleine, Lille, France*

<sup>b</sup>*Irish Manufacturing Research, Aerodrome Business Park, Rathcoole, Ireland*

<sup>c</sup>*Enterprise Research Centre, School of Engineering, University of Limerick, Ireland.*

---

## Abstract

In manufacturing systems where productivity is constrained by operators' availability, cross-training strategies can be used to enable dynamic assignment of operators to workstations. However, finding an assignment approach that efficiently works under various system conditions is not trivial as, among other factors, the level of cross-training, the production duration and the initial conditions of the system can influence the assignment approach's performance.

To overcome this issue, in the case study presented in this paper, operator assignments have been modeled using a simulation-based optimization approach, with an "outer" optimizer that selects assignment-related parameters to simulate based on the system conditions, and an "inner" optimizer integrated with a simulation model that generates optimal assignments. For the case described here, which is modeled using a deterministic simulation model, the "outer" optimizer is an Ant Colony Optimizer (ACO) and the "inner" optimizer is a Binary Integer Programming (BIP) model. The ACO will select weights for the assignment objectives of the BIP multi-objective function so that throughput is maximized. The BIP is called by the system simulation model at fixed intervals or when a system status changes to assign operators to workstations. Results show that the simulation-based optimization approach generates higher throughput performance than static WIP-base assignment, especially when longer production duration are considered. The effects of cross-training and production duration on production throughput are also investigated. The simulation-optimization approach used can be abstracted to a framework where the "outer" and "inner" optimizers may be applied to different domains than the case study addressed here and can also be applied to stochastic simulation models.

*Keywords:* Operator Assignment, Simulation-Optimization, Ant Colony Optimization, Binary Integer Programming

---

## 1. Introduction

In labour-intensive production systems, it may happen that due to staffing decisions at the strategical level and/or operators' absenteeism the number of operators assigned to a system during a production shift is less than the number of workstations to be operated. Cross-training strategies can be used

---

\*Corresponding author.

*Email addresses:* [georgios.dagkakis@nexedi.com](mailto:georgios.dagkakis@nexedi.com) (Georgios Dagkakis), [anna.rotondo@imr.ie](mailto:anna.rotondo@imr.ie) (Anna Rotondo), [cathal.heavey@ul.ie](mailto:cathal.heavey@ul.ie) (Cathal Heavey)

to increase workforce flexibility and effectively address both short-term needs or sudden changes due to workforce shortage, such as an operator's absence or resignation (Abrams and Berge, 2010). Cross-trained workers generate flexible capacity, that is, workers can be allocated dynamically to where they are needed and when they are needed (Hopp and Oyen, 2004). However, the implementation of cross-training strategies in a production system requires the identification of operators' coordination policies that exploit an operator's flexibility in the most efficient manner based on both the system's characteristics and the business' objectives (Hopp and Oyen, 2004). Previous studies have demonstrated that the level of workforce flexibility affects the efficacy of an assignment approach (Uzun Araz and Salum, 2010; Downey and Leonard, 1992); more generally, productivity performance of a system is impacted by its settings and state changes. As a result, the use of static assignment approaches, which do not take any system state factor into account, will result in below optimal solutions. (Uzun Araz and Salum, 2010).

This paper will study the problem of assigning cross-trained operators to workstations of a unidirectional assembly line that operates in a medical device fabrication plant. Due to technological issues and strict production flow regulations, the line is not perfectly balanced. To reduce manufacturing costs, operators are cross-trained, and the number of operators assigned to the assembly line is less than the number of workstations where manual tasks are performed (Downey and Leonard, 1992). The main challenge addressed here is to develop an assignment approach that identifies the best balance of assignment objectives based on given system settings (i.e., initial system status, level of workforce flexibility, production duration) so that throughput is maximized. Within this context, the initial system status refers to the combination of line layout, Work In Progress (WIP) distribution and operators' availability. Workforce flexibility is intended as the level of skills possessed by each operator so that they can perform tasks at different workstations. Production duration is the time horizon considered for throughput maximization (i.e., operational shift, day, month, etc.).

To address this challenge a novel simulation-optimization approach is developed; this approach is based on a two-layer optimization framework that uses simulation to both evaluate different scenarios (i.e., alternative combinations of optimization objectives used for assignment decisions) and anticipate the system state at the decision points so that optimal assignment decisions can be made. In the "inner" layer, a multi-objective function is used by a binary integer programme (BIP) model, that is integrated with a simulation model, to assign operators to workstations; the weights on individual assignment objectives are selected using an Ant Colony Optimizer (ACO) as the "outer" optimizer. The BIP model is called at fixed time intervals or when system states change. Different system states executed in the simulation model may cause the BIP to execute at different non-fixed times (i.e., WIP build-up at workstations in the assembly line), thus allowing the ACO to select the best combination of weights for the multi-objective function. The efficacy of the suggested simulation-optimization approach in delivering optimal solutions is tested with respect to variations in both the level of workforce flexibility and different production duration. The simulation-optimization approach presented here can be used to address other problems (i.e., job sequencing and job routing), detailed later after introducing in Section 3 more details of the simulation-optimization framework.

The rest of the paper is as follows: in the following section, a brief review of relevant literature on the

concept of operator’s flexibility and corresponding assignment approaches is presented. In Section 3, we present a generic simulation-optimization framework for deterministic simulation. Then, in Section 4 we present the case study of an assembly line and in Section 5 we adopt the generic simulation-optimization framework to address the case study. Section 6 introduces the experimental plan with Section 7 presenting the results, which we also compare with a widely used heuristic for operator assignment (Downey and Leonard, 1992; Hopp et al., 2004). Finally, Section 8 concludes the paper with considerations on the work presented and future research directions.

## 2. Literature Review

Workforce flexibility is increasingly adopted as an operational strategy to generate flexible capacity within a company and secure its competitiveness and survival in extremely dynamic manufacturing environments (Abrams and Berge, 2010). Workforce flexibility can be achieved through cross-training, or multiskilling, that is a workforce development strategy through which employees are trained to broaden their skillsets and perform more than a single task (Abrams and Berge, 2010). A more flexible workforce fosters the realization of more robust (Abrams and Berge, 2010; Sennott et al., 2006) and efficient (Chen and Askin, 2006) production systems, increases workers’ motivation and responsibility (Crociet al., 2000), reduces workers’ alienation (Downey and Leonard, 1992), and provides many other benefits not necessarily related to production efficiency, such as quality improvement, learning-curve acceleration, improved organizational culture (Herzenberg et al., 2000). On the other hand, the increased productivity gained by reducing idle times may conceal potential reductions in workers’ efficiency due to deviations from skill specializations (De Bruecker et al., 2015; Abrams and Berge, 2010).

Considering the relevance of cross-training as a strategic means to achieving production agility (Malecki, 1996), Hopp and Oyen (2004) developed a framework for classifying workforce flexibility and evaluating its advantages from both a strategic and tactical perspective. Key elements are cross-training skill patterns and worker coordination policies. This last aspect concerns the assignment of operators to tasks over time and becomes critical when improving productivity. The minimization of the number of operators that allow the accomplishment of productivity targets is a common objective in flexible worker assignment problems (Dolgui et al., 2018; Inman and Blumenfeld, 2010). Alternatively, throughput maximization is pursued as an objective when the workforce size is set, and worksharing is allowed (Chen and Askin, 2006; Anuar and Bukchin, 2006). More generally, throughput maximization is a common objective in systems where productivity is constrained by the availability of both workstations and operators, namely, Dual Resource Constrained (DRC) systems. While DRC systems concurrently address operators’ assignment and job dispatching, dynamic operator assignments uniquely characterizes these systems since the number of available operators is always less than the number of machines operating in the system (Cesani and Steudel, 2005). The assignment strategies suggested in the DRC literature are similar to the assignment approach used by the supervisors managing the assembly line modeled here, which is based on the use of “when” and “where” rules. These rules are followed to ensure efficient and effective workforce re-assignment in real-time and maximize manufacturing productivity (Xu et al., 2011). “Where” rules concern the selection of machines to assign to different operators whereas

“when” rules define the time when re-assignment is allowed (Ammar et al., 2013). “Where” rules correspond with defining worker coordination policies; for instance, Hopp et al. (2004) compare nine heuristic worker coordination rules which are classified into static rules (i.e., they do not use information on the system status), queue-length-based rules (i.e., rules that use real-time information on the WIP levels) and workload-based rules (i.e., rules that use information on the cumulative processing times for the jobs in a queue). “When” rules are generally categorized into centralized and decentralized rules. Centralized rules allow an operator to be reassigned to another machine as soon as the current job is completed whereas decentralized rules allow operators to transfer when the queue at the machine currently assigned is empty (Ammar et al., 2013). Xu et al. (2011) argue that “when” rules are more important and have more effect on the overall system performance than the “where” rules.

The DRC problem also includes job dispatching rules used to determine the order in which jobs are processed at each machine. These include standard dispatching rules, such as earliest due date or first-come-first-serve, used along with ad-hoc rules, such as the forced pace rule or the 2Q rule (Kher, 2000), which are purposely developed to manage high priority orders. For the problem analysed here, job dispatching rules are not relevant since only one product type is processed in the assembly line modelled and a push production logic is adopted. Studies that have investigated the effect of rule combinations on systems in DRC environments suggest that the appropriate choice of these rules is situational, which means that no particular combination of assignment and dispatching rules globally outperform others.

Any change in the problem objectives, including the optimization of the production duration (Uzun Araz and Salum, 2010), the system configuration, the operator’s availability and efficiency in processing different tasks (Costa et al., 2013), the transfer costs between stations (Malhotra and Kher, 1994) will impact the choice of optimal rules to be implemented. Support systems, such as simulation-based adaptive control schemes, are used to identify the combination of rules that work best in given states of the system but the solution found can quickly become obsolete due to variations in the system and a new assessment is required (Ferjani et al., 2017; Uzun Araz and Salum, 2010). For this reason, simulation meta-models, such as artificial intelligence models, are an efficient alternative that can be used to speed up the decision process and select rules at each assignment point (Can and Heavey, 2012; Uzun Araz and Salum, 2010). Assignment points (or “when” rules) can be triggered by events that mark a system setting variation, or they can be time interval-based. For instance, Uzun Araz and Salum (2010) use simulation to analyze the impact of system settings and state variables (job tardiness allowance, average queue length, mean flow time, etc...) on performance measures. Then, to make the decision system more efficient for real-time they use an artificial neural network (ANN) model trained on the simulation results. Fuzzy logic is used to assemble the performance indices output into a global measure, so that multi-criteria decision making is enabled.

Uzun Araz and Salum (2010) highlight the importance of having dynamic scheduling rules (Ouelhadj and Petrovic, 2009) and variable length assignment intervals, allowing dynamic allocation of operators to workstations based on the current system state. One approach that attempts to dynamically allocate operators is to develop a set of operator assignment rules that can be selected dynamically based on the current system state (Xu et al., 2011). However, the definition of a limited set of rules from which the

optimal ones are to be chosen proves per se a limiting factor (Uzun Araz and Salum, 2010).

In this study, we eliminate the need of having a set of pre-defined static rules and aim at enhancing the allocation of the operators' assignment logic by implementing a BIP approach in which a weighted multi-objective is allowed to adapt to the production's current settings using an ACO. In BIP, model objectives and constraints used are inspired by sensible assignment practices observed in the real assembly line in this study, these practices are also commonly found in the literature (Hopp et al., 2004). The BIP model is embedded into a simulation framework and is called using a time interval-based approach combined with an event-based approach (Uzun Araz and Salum, 2010). In the next section we introduce the simulation-optimization concept and present a generic simulation-optimization framework for deterministic simulation that can be used to address dual optimization problems. Its specific implementation to the case study is detailed in section 5.

### 3. Optimization Framework for Deterministic Simulation

Figueira and Almada-Lobo (2014) taxonomize the interaction between simulation and optimization. The most typical use of simulation-based optimization consists of using simulation for solution evaluation (Figueira and Almada-Lobo, 2014). In typical implementations simulation is used as a black box that evaluates objectives and constraints for a particular input (Amaran et al., 2014) and is coupled with an optimizer, for example metaheuristics such as genetic algorithms (Hegazy, 1999; Azzaro-Pantel et al., 1998) or ACO (Brailsford et al., 2007).

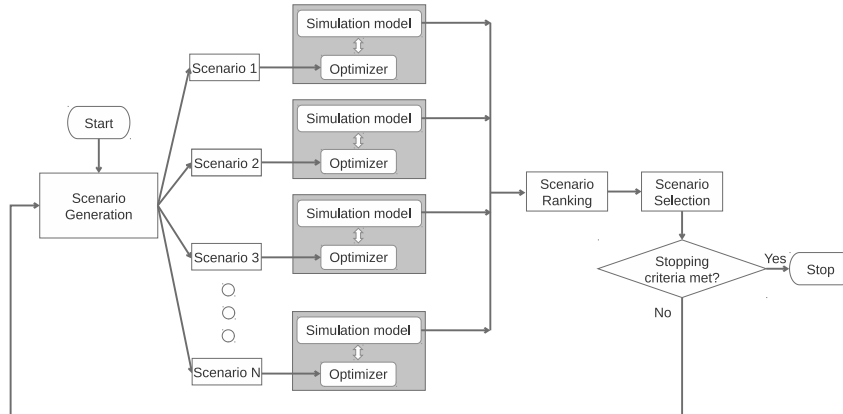


Figure 1: Generic framework of Dual Simulation Optimization (DualSimOpt).

A generic schema of the the Dual Simulation Optimization (DualSimOpt) framework is shown in Figure 1. It consists of *Scenario Generation*, which we term the “outer” optimizer and an *Optimizer* that integrates with the *Simulation model*, which we term the “inner” optimizer. The *Scenario Generation* is responsible for generating potential scenarios (see Figure 1), whose evaluation requires the availability of prediction models or tools that are able to anticipate the effects that systems' settings have on systems' performances; for this reason, simulation proves a suitable scenario-based evaluator as it can capture the complex and dynamic relationships between input and output variables. In its most straightforward implementation, this “outer” optimization module can consist of a full enumeration approach (i.e. all possible scenarios are investigated and the best solution is selected) or design of experiments

approaches whereby scenarios are developed based on the experimental plan most appropriate for the analysis’s purposes. Alternatively, in cases that enumeration can lead to huge scenario spaces, the scenario generation could consist of evolutionary optimizer, founded on population-based search logic, with a feedback loop that uses the simulation results intelligently to guide the search through the solution space. The “inner” *Optimizer* that integrates with the *Simulation model* will optimize at different decision points simulated within the simulation model. Algorithm 1 presents the general steps of the framework shown in Figure 1.

---

**Algorithm 1:** Dual simulation optimization framework

---

Initialize Scenario Generation

**while** *not Stopping Criteria met* **do**

    Update Scenario Generation based on the results of previous loop

    Generate  $N$  Scenarios using Scenario Generation

**for**  $i$  **to**  $N$  **do**

        Start simulation of Scenario  $i$

**for**  $t$  **to** *SimulationLength* **do**

            Simulate Scenario  $i$

**if** *Decision Point* **then**

                | Call Optimizer

**end**

**end**

        Store result of Scenario  $i$

**end**

    Rank Scenarios 1 ...  $N$

**end**

Return the best ranked Scenario and stop

---

The simulation-optimization approach developed here can be viewed as combining two hierarchical structures identified by [Figueira and Almada-Lobo \(2014\)](#), namely Optimization with Simulation-based iterations, which we denote as the “outer” optimizer, and Simulation with Optimization-based iterations, which we denote as the “inner” optimizer. The Scenarios in Figure 1 may relate to different configurations that would be simulated. For instance, the framework can be used to address a flexible job-shop scheduling problem where two decision levels must be solved, i.e., job sequencing and job routing. The “outer” optimizer can generate alternative sequencing solutions and the “inner” optimizer can make use of heuristics or an alternative optimization approach to make routing decisions within a simulation run. Other examples where this framework can be applied is scheduling optimization in batching tools and hierarchical production planning in supply chains, which are also characterized by dual optimization problems (i.e., sequencing and scheduling or assignment). It could also be used to optimize multi-objective functions, where a scalarization method converts a multi-objective problem into a parameterized single-objective ([Hunter et al., 2019](#)), but where the parameters of the single-objective problem are also optimized using the framework proposed here. This article will present an application of the framework to optimize the assignment of cross-trained operators to an assembly line, with the

case study described in the next section.

#### 4. Case Study

The case study presented here is inspired by a problem statement experienced at various manual assembly lines operating in a medical device fabrication company. In these lines, productivity is constrained by the availability of both workstations (i.e., some workstations are shut down during specific shifts) and operators (Xu et al., 2011). The number of operators allocated to a production line is generally less than the number of machines operating. Cross-training of operators is used to achieve production targets. The aim of the staffing policy at the company involved in this study is based on a “3 tasks to 3 operators” concept, whereby, it will try and train operators on up to 3 tasks. However, to train operators on tasks is an expense and, as we will see in examples later, some operators may only be trained on one task. At the beginning of each shift, the production supervisor assigns operators to workstations based on the number of operators and machines available. A rule of thumb generally adopted at the company is that in case of operator shortage, the supervisor assigns operators to the first workstations in the line so that WIP is built at the intermediate workstations. Towards the end of the shift or during the following shift, operators are moved to the last workstations to re-balance the line’s WIP. The implementation of this rule and its effect on production are also dependent on the supervisors’ experience and his/her promptness in making assignment decisions based on the line status. The assembly line operates for two 9 hour shifts from 7:00-16:00 and 16:00-01:00 and one shift on Fridays from 7:00-14:00 with a 1-hour break divided between different periods within a shift. The objective here is to develop a decision support tool able to generate optimal operator assignment schedules to maximize throughput for a shift or a given production duration.

Figure 2 reports the schematic layout of a pilot assembly line as modeled in Discrete Event Simulation (DES). The parallel machines (or sub-lines) that operate in the system are considered equivalent from both a performance and training requirement perspective so that an operator trained on a particular task can perform that task at any machine available in the workstation. A workstation is intended here as the set of machines where the same operational task is performed, i.e., St6M0 and St6M1 perform the same task. The two sublines between SB and B7 are disjoint (i.e. no batch can be shared or moved across the lines). Only sub-batch processing is allowed across the sequential machines (in a subline) and a “line clearance” policy (i.e., physical segmentation of a line with respect to batches in order to avoid cross-batch contamination) is also implemented. The other two processes across which batch splitting is allowed are the last two in the line (i.e., St7 and St8) where a routing buffer (B12) rather than a standard buffer is used. This buffer routes sub-batches of the same batch towards the same machine (either S8M0 or S8M1).

Processing times observed in the line are reported in Table 1. Units are processed in batches of 80; however, sub-batches are also defined at each station to allow for more processing flexibility (i.e., an operator can leave a station after completing a sub-batch). The processing times in the company are all assumed to be deterministic, even though times will experience a low level of variability. This however,

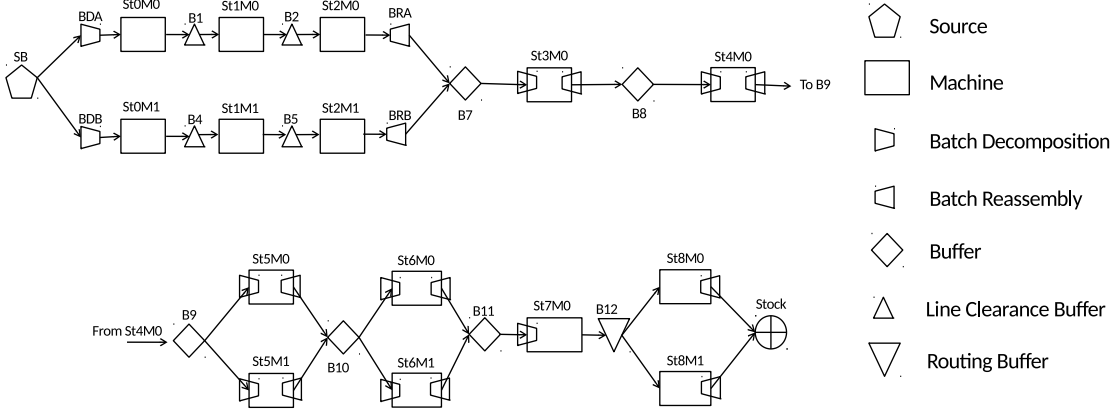


Figure 2: Layout of a pilot assembly line.

Table 1: Processing times.

Station	St0	St1	St2	St3	St4	St5	St6	St7	St8
Unit Processing Time (PT) [min]	0.77	0.82	0.8	0.2	0.4	0.22	0.82	0.52	1
Sub-Batch Size [unit]	20	20	20	80	40	80	10	10	10
(sub)Batch PT [min]	15.4	16.4	16	16	16	17.6	8.2	5.2	10

is a requirement by the company as it is used to measure the performance of each shift. This results in a deterministic simulation model which will not require any replications.

## 5. Dual Simulation Optimization: Case Study

To address the case study we use the specific framework shown in Figure 3 derived from the generic one shown in Figure 1. This framework integrates simulation with optimization, where a DES model interfaces with an “inner” BIP optimizer, and where an “outer” ACO optimizer evaluates different scenarios, with each ant representing weights (i.e., scenario) for each objective that is simulated. The BIP model is described in Section 5.2 while the ACO algorithm is described in Section 5.3.

In this case study, simulation is used to predict the system status (i.e., variables such as WIP distribu-

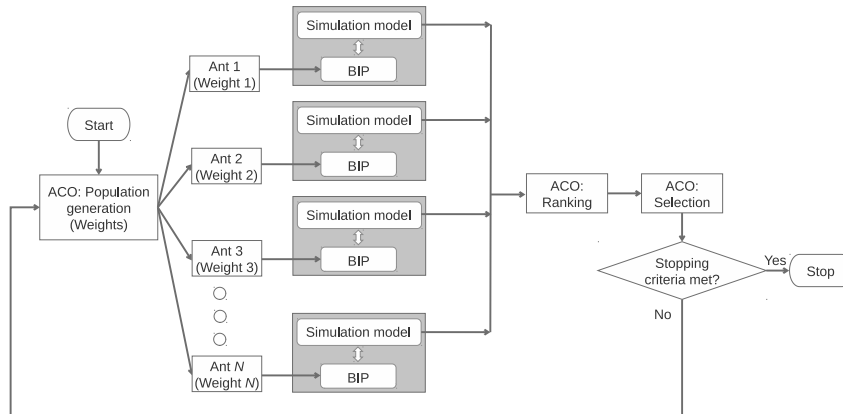


Figure 3: DualSimOpt for case study.

tion and throughput) at given points in time when operator assignment is required. Operator assignment is performed with a multi-objective BIP model which is called by the simulation model and obtaining from it all updated data required to make assignment decisions. The “outer” optimizer, which is based on ACO is responsible for generating potential scenarios (in this case an ant) and that is optimized using an “inner” optimizer which consists of simulation interfaced with a BIP model.

### 5.1. Simulation Model

The simulation model of the assembly line can be used to efficiently solve the operator assignment problem and assist the line supervisor in making re-assignment decisions that can minimize the impact of operators or machines unavailability on productivity and maximize throughput. ManPy was used to model the assembly line (Dagkakis et al., 2015). ManPy is an Open Source (OS) simulation library (<http://www.manpy-simulation.org/>) written in Python, that uses the SimPy software (<http://simpy.readthedocs.org/>) to implement a process-based simulation worldview. The simulation model of the case study is presented in Dagkakis et al. (2014). As the simulation model is deterministic replications are not required. Realistic initial WIP levels are set at the start of the simulation model.

### 5.2. Binary Integer Programming Model

The first optimization aspect concerns the operators’ assignment decision at a specific point in time; this problem has been formulated as a multi-objective BIP problem where the binary optimization variables consist of the assignment variables  $x_{ij}$  (i.e., operator  $i$  is assigned to machine  $j$  if  $x_{ij}$  is 1) and different objectives are taken into account. Following a multi-attribute utility approach (Morrice et al., 1998), the objectives are firstly normalized and then grouped in a combined metric calculated as a weighted sum so that the objective function can be formulated as:

$$\max f(x_{ij}) = \sum_k w_k * obj_k(x_{ij}) \quad (1)$$

where  $w_k$  represents the weight of the  $k^{th}$  objective ( $obj_k(x_{ij})$ ) and each objective is a function of the assignment variables. The objectives considered in this formulation include:

- **Obj<sub>1</sub> - Maximize the number of assigned operators:** this is to avoid operators left idle;
- **Obj<sub>2</sub> - Prioritize assignment of operators to machines with highest WIP:** this is inspired by both the current assignment practice adopted at the company and heuristic approaches suggested in the DRC literature (Xu et al., 2011);
- **Obj<sub>3</sub> - Minimize unnecessary operators’ transfers:** this is to avoid operators that are re-assigned to different machines when there is no change in the operator/line availability configuration. This objective was suggested by production management at the modeled assembly lines to minimize hand-over delays (which are not captured by the model); a source of production variability within a batch.

- **Obj<sub>4</sub> - Support uniform distribution of operators across workstations:** if possible, operators should be assigned in order to guarantee that at least one machine per workstation is operational. This is to support production flow continuity across the line.
- **Obj<sub>5</sub> - Prioritize assignment to machines that have been idle for long:** this is to avoid situations whereby an operator is repeatedly assigned to the same machine if he/she can operate machines that have been idle for a long time. This objective is specifically thought for the assembly sublines where it may happen that a batch remains incomplete for a long time due to WIP levels that are not as high as at other processes in the line.

Assuming that  $m$  machines and  $n$  operators are available for assignment, relevant constraints applied to this problem can be formalized as follows:

$$\sum_{j=1}^m x_{ij} \leq 1 \quad \forall i = 1, 2, \dots, n \quad (2)$$

$$\sum_{i=1}^n x_{ij} \leq 1 \quad \forall j = 1, 2, \dots, m \quad (3)$$

Equation 2 imposes that an operator can be assigned to one machine at most and Equation 3 suggests that a machine can be operated by at most one operator.

The assignment of operators to machines is also constrained in terms of operator's skills. Training constraints are implicitly applied to the definition of decision variables as an assignment variable  $x_{ij}$  is initialized only if operator  $i$  is trained on the task performed at machine  $j$ .

The *Obj<sub>1</sub>* objective is given by:

$$Obj_1 = \sum_{i,j} \frac{x_{ij}}{n} \quad (4)$$

Besides information on operators' skills and machines and available operators at the assignment decision time, the *Obj<sub>2</sub>* objective requires information on the WIP level at all machines. Thus *Obj<sub>2</sub>* can be formally expressed as:

$$Obj_2 = \frac{\sum_{i,j} WIP_j \times x_{ij}}{\sum_{k=1}^m WIP_k} \quad (5)$$

where  $WIP_j$  represents the work in process at machine  $j$ .

For the definition of *Obj<sub>3</sub>*, dummy variables,  $\Delta Ass_{ij}$ , are introduced that represent assignment changes with respect to the previous assignment plan, where  $\Delta Ass_{ij}$  will be equal to 0 if the assignment of operator  $i$  with respect to machine  $j$  is not modified by the assignment solution, otherwise it will be 1.  $\Delta Ass_{ij}$  is calculated using the following two symmetrical inequality constraints:

$$\Delta Ass_{ij} \geq x_{ij}^* - x_{ij} \quad (6)$$

$$\Delta Ass_{ij} \geq x_{ij} - x_{ij}^* \quad (7)$$

The constraints above are introduced to force  $\Delta Ass_{ij}$  to assume positive values and  $x_{ij}^*$  is the equivalent of  $x_{ij}$  in the previous assignment plan.  $Obj_3$  can therefore be expressed as:

$$Obj_3 = \sum_{i,j} \frac{\Delta Ass_{ij}}{2 \times Ass\#} \quad (8)$$

where  $Ass\#$  represents the number of assigned operators in the previous assignment plan. If no previous assignment plan exists or it is not meaningful for the definition of a new plan (i.e., new shift starts),  $Obj_3$  is automatically neglected in the objective function.

Also  $Obj_4$  requires the introduction of dummy variables,  $\Delta Station_{hk}$ , through which the difference between the ratio of operators assigned to two generic stations,  $h$  and  $k$ , is calculated as:

$$\Delta Station_{hk} \leq \sum_{i \in n, j \in St_h} \frac{x_{ij}}{\#St_h} - \sum_{i \in n, j \in St_k} \frac{x_{ij}}{\#St_k} \quad (9)$$

$$\Delta Station_{hk} \leq \sum_{i \in n, j \in St_k} \frac{x_{ij}}{\#St_k} - \sum_{i \in n, j \in St_h} \frac{x_{ij}}{\#St_h} \quad (10)$$

where  $St_k$  and  $\#St_k$  represent the set of machines and the number of machines that belong to station  $k$ , respectively. The number of operators assigned to a station is normalized with respect to the number of machines operating in the station in order to keep  $\Delta Station_{hk}$  in the interval  $[-1, 0]$  and avoid that the minimization of  $\Delta Station_{hk}$  between two stations characterized by substantially different capacity leads to solutions where available operators are deliberately not allocated to the higher capacity station in order to keep  $\Delta Station_{hk}$  minimal. The two symmetrical inequalities are used to force  $\Delta Station_{hk}$  to assume negative values as this variable should be minimized (i.e., be reduced to 0).  $Obj_4$  can then be expressed as:

$$Obj_4 = \sum_{h,k} \Delta Station_{hk} \quad (11)$$

Finally,  $Obj_5$  is calculated based on the latest time ( $LastTime_j$ ) at which a machine  $j$  has been assigned an operator throughout the simulated time.  $LastTime_j$  is normalized as follows:

$$LastAss_j = 1 - \frac{LastTime_j}{MaxAss} \quad \forall j = 1, 2, \dots, m \quad (12)$$

where  $MaxAss$  represents the latest assignment time across all machines. If the sum of  $LastAss_j$  across all machines ( $LastAss$ ) is null (i.e., equal to 0),  $Obj_5$  is neglected as all machines are equivalent in the last assignment. On the contrary, when  $\sum_j LastAss_j$  is greater than 0, there exist machines in the assembly line that have been left idle with no operator for longer times than other machines.  $Obj_5$  is calculated as follows:

$$Obj_5 = \sum_{i,j} \frac{LastAss_j \times x_{ij}}{\sum_j LastAss_j} \quad (13)$$

Last assignment times further back in the past (i.e., small values of  $LastTime_j$ ) correspond with higher values of  $LastAss_j$ ; which means that  $Obj_5$  should be maximized in order to favor solutions that avoid keeping machines characterized by some level of WIP idle for long times (i.e., see  $Obj_2$ ).

Although the binary mathematical model can work independently of the simulation model, its integration within the simulation module enables the generation of good assignment schedules for an entire

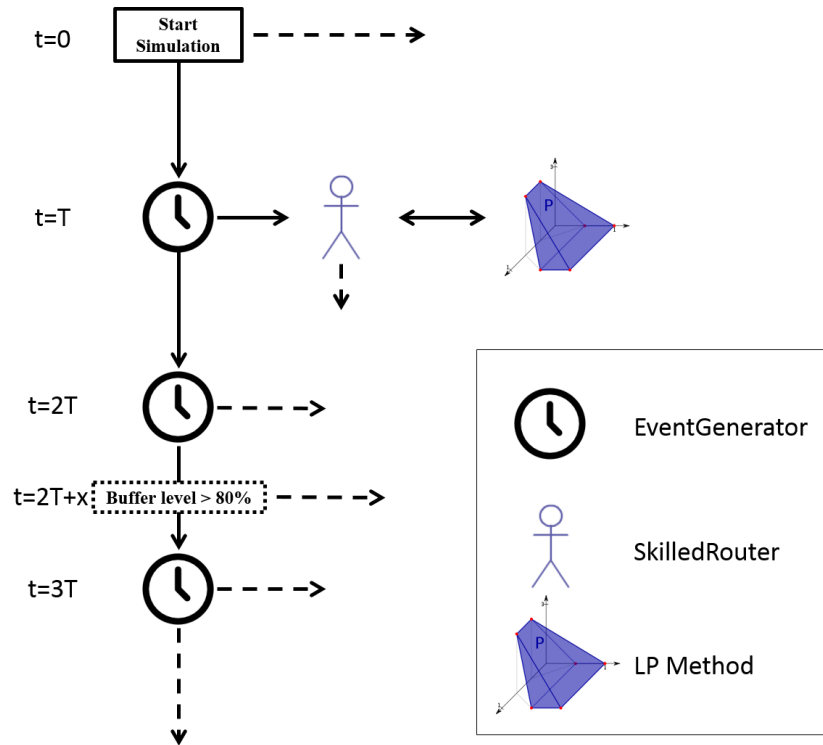


Figure 4: Interaction between ManPy model and binary optimization.

production shift by using the simulation model capability of predicting operator assignments to machines and its effect on the movement of WIP through the assembly line.

The BIP model has been coded using PuLP, a Python library that offers pre-defined methods to model a binary mathematical model and generates Mathematical Programming System (MPS) files that are solved by solvers, such as GLPK, COIN CLP/CBC and CPLEX.

The ManPy model was formulated in order to use the binary mathematical model as a black box. A custom ManPy object built called *SkilledOperatorRouter* was developed for the needs of the case. This checks the state of the system and if assignment of operators is required it invokes the binary mathematical model sending all the information it needs about the state, i.e., the available machines and operators, the previous assignments and the weights.

There are three cases when the assignment is triggered as the model runs (see Figure 4):

- At given intervals:** This simulates the model, where the line supervisor regularly checks the state of the system and decides if an assignment is needed. In the simulation model, this is achieved with a generic ManPy object called *EventGenerator*. The common behavior of *EventGenerator* is to be activated at given simulated time intervals and invokes a method. In this specific model, an instance of *EventGenerator* was added in order to invoke the *SkilledOperatorRouter*. The triggering frequency was set to 40 minutes of simulated time, after consultations with industrial engineers working on the assembly line, but we also experimented with this value, in order to see how alternative intervals of evaluation affect the systems performance. From these experiments, a 40 minute assignment interval was deemed a good compromise between throughput performance and operators' transfer minimization. Considering the sub-batch processing times observed in the

assembly line (Table 1), a 40 minute assignment interval would also allow an operator to complete at least two (sub)batches when assigned to a station. Therefore, the resulting assignment control policy is between a purely centralized control strategy, where operators are forced to leave their current station as soon as they complete a (sub)batch, and a decentralized strategy, where operators cannot leave their current station until the upstream buffer is emptied. Unless specified otherwise, all experiments reported in this paper are based on a 40 minute assignment interval.

- **Model status changes:** There are two status changes that the model uses to call the *SkilledOperatorRouter* method. First, when an upstream buffer is 80% full to prevent machine blockage. Second, when a machine is starved for more than 30 minutes, this triggering event is considered in order to be more responsive to situations when the production flow is blocked at a certain point in the line.
- **Certain events:** When a shift changes, reflecting what happens in the real system.

When the assignment is triggered, a two-step procedure is applied. In the first step, machines that are not blocked are filtered and sent to the BIP model. This is to avoid operators being assigned to machines where production is constrained and prioritize the assignment of operators to machines where they can immediately start processing items (i.e. there is sufficient capacity in the downstream buffer). After the first assignment, if there is any operator left idle (i.e., the number of available operators is greater than the number of unblocked machines), a second assignment is performed using idle operators and all machines left unattended from the first assignment (i.e., blocked machines).

The BIP model returns a Python dictionary that describes the optimal assignment based on the current system’s status. This has the form of:

```
{ 'Operator_Id': 'Machine_Id' }
```

Then, the *SkilledOperatorRouter* imposes the assignment in the model. This may not happen instantaneously, since some operators may need to end their operation in process before they get moved to the next station.

### 5.3. Ant Colony Optimization Algorithm

The “outer” optimizer in Figure 3, which is based on ACO, concerns the definition of optimal weights for the objective function of the BIP model. The BIP solution quality is obviously dependent on the weights assigned to the various objectives; this is also because the objectives considered are conflicting in terms of their impact on productivity. As the assignment plan of the operators to the machines should promote a right balance between productivity and implementation performance and considering the experimental evidence that objective weights’ performance is closely related to the system’s status, an approach that optimizes the weights for given system’s conditions has been developed. Optimization is performed by means of an ACO algorithm inspired by the Min-Max Ant System optimization logic (Stützle and Hoos, 2000). ACO is an iterative algorithm inspired by the biological behaviour of real ant colonies. The study of real ants attracted the interests of biologists due to the ability of these almost

blind animals to find the shortest path from their colony to feeding sources (Dorigo et al., 1996). It was found that pheromone trails are laid by the ants while in search for food and the intensity of the pheromone, which reduces in time, is the criterion followed to make decisions on route directions. This search behaviour has been adapted to an algorithmic decision process based on the idea that solutions preferred by previous ants are chosen with higher probability. As a result, general-purpose optimization algorithms (or meta-heuristics) have been derived. An ant is an agent that constructs a possible solution to a combinatorial optimization problem. A population of ants consist of a set of  $n$  ants, which correspond to  $n$  possible solutions, where  $n$  is also called population size. When the algorithm starts (i.e., first cycle), generally a random population is created. In the following cycles, information from the previous cycles are used to update the pheromone values and improve the ants' performance. For the case study discussed here, an ant represents a combination of weights to be used for the BIP objective function. During each ant cycle, ants select a weight for each objective so that the BIP problem is correctly defined. The ants are then simulated and the corresponding results are ranked based on productivity performance; the throughput obtained at the end of the simulation run is used as a performance metric. The generation of the new population of ants entails a pheromone update mechanism based on an elitist logic whereby the best two ants from the current cycle are used to update the weight selection probability for each objective. The pheromone values are updated so that the probability of selecting weights carried by the best ants is reinforced; an evaporation rate is applied to regulate the relevance of a previous weights' probability during the pheromone updating mechanism.

The representation of the selection probability for an objective's weight is made by means of a list of weights that is initialized to a limited set of user-defined values. Indeed, the end-users define the lower and upper bound for an objective weight and a discrete step is applied to generate the value domain for each objective. For instance, if a lower bound of 0 is applied to  $Obj_1$  and its upper bound is set to 1, a step of 0.1 would lead to the following initial list of weights for  $Obj_1$ :

$$[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$$

More generally, if all weights present the same value intervals (which is not the case in the implementation of the optimization algorithm presented in Section 7), the solution space will be given by the combination of weight values represented by the following dictionary (WeightDict):

$$\begin{aligned} Obj_1 &: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], \\ Obj_2 &: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], \\ Obj_3 &: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], \\ Obj_4 &: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], \\ Obj_5 &: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] \end{aligned}$$

The weight selection mechanism consists of randomly selecting a weight from the weight list of each objective. For instance, an ant could be represented by the following dictionary:

$$Obj_1 : 0.2, Obj_2 : 0.3, Obj_3 : 1.0, Obj_4 : 0.7, Obj_5 : 0.3$$

which suggests that a solution characterized by  $Obj_1$  equal to 0.2,  $Obj_2$  equal to 0.3, etc... will be assessed.

The updating mechanism consists of adding the weights associated with the best two ants to the objective weight list. For instance, if the two best-performing ants in terms of associated throughput are:

$$Obj_1 : 0.2, Obj_2 : 0.3, Obj_3 : 0.1, Obj_4 : 0.7, Obj_5 : 0.3$$

and

$$Obj_1 : 0.4, Obj_2 : 0.2, Obj_3 : 0.1, Obj_4 : 0.6, Obj_5 : 0.1$$

the weights dictionary will be updated, with new additions underlined, to the following:

$$Obj_1 : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, \underline{0.2}, \underline{0.4}],$$

$$Obj_2 : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, \underline{0.3}, \underline{0.2}],$$

$$Obj_3 : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, \underline{0.1}, \underline{0.1}],$$

$$Obj_4 : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, \underline{0.7}, \underline{0.6}],$$

$$Obj_5 : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, \underline{0.3}, \underline{0.1}]$$

The updated dictionary implies that the weight values of the best performing ants will have increased probability of being selected in the following generation.

This updating mechanism is equivalent to the use of the following probability function:

$$P(w_j) = \frac{\tau(w_j)}{\sum_i \tau(w_i)} \quad (14)$$

where  $w_j$  represents the weight of the  $j^{\text{th}}$  objective and  $\tau(w_j)$  the pheromone value that is updated according to Equations 15 and 16.

$$\tau(w_j) = (1 - \rho)\tau(w_j) + \Delta\tau(w_j) \quad (15)$$

where  $\rho$  is the pheromone evaporation rate and is set equal to 0 in this case;  $\Delta\tau$  is the amount of reinforcement given to a particular pheromone value and is defined as follows:

$$\Delta\tau(w_j) = \begin{cases} 1 & \text{if } w_j \text{ is selected in the best solutions} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

The initial pheromone values are set to 1 for each weight.

A *best-so-far* elitist logic is used so that the ants that perform best since the first cycle are kept in the solution pool. Upon reaching the termination criteria (i.e., simulation of a pre-defined number of cycles or solution convergences), the procedure is concluded by selecting the best solution, that is the combination of BIP weights that generate assignment schedules characterized by the highest throughput.

This ACO approach is used as the “outer” optimization module in the optimization framework in Figure 3 as it enables the generation of scenarios that define optimization parameter settings (see Algorithm 2). The ACO approach was implemented in Python as an algorithm that is set to call the ManPy model each time it needs to evaluate a design. In this case ACO sends to ManPy the same simulation model changing only the weights that *SkilledOperatorRouter* will use when calling the BIP model. This algorithm receives as user input the number of generations (*numberOfGenerations*) and the number of ants

for each generation (*NumberOfAntsPerGenerations*) and executes algorithmic steps that are typical of an evolutionary algorithm approach (i.e., generation, evaluation, selection) until a terminating condition is met or the full cycle of predefined generations expires.

---

**Algorithm 2:** Ant colony algorithm

---

```

for  $i$  to NumberOfGenerations do
  for  $j$  to NumberOfAntsPerGeneration do
    | Select Ants Randomly from WeightDict and add them to AntList
  end
  for  $j$  to NumberOfAntsPerGeneration do
    | Ant = AntList[j]
    | Invoke ManPy for Ant
    | Save Results (i.e., detailed schedule and assignment plan) In Attribute of Ant
    | Calculate Ant Score (i.e., throughput) and save it in Attribute of Ant
  end
  sort AntList based on Score (descending)
  for  $j$  to length of numberofAntsThatCarryTheirPheromone do
    | Update WeightDict with the weights of Ant[j]
  end
  Check Terminating Condition
end

```

---

## 6. Experimental Setup

Experiments have been conducted to demonstrate the efficacy of the optimization framework using the case study. The experimental plan has been progressively developed to validate the underlying assumption in this study that objectives' weights affect throughput and that the optimal weight combination is affected by the system state. Considering that the shortage of operators and the availability of a flexible workforce are the main reasons why assignment policies are requested in the system modelled, the experiments conducted aim at exploring the impact of workforce flexibility level on throughput.

### 6.1. Objective Function Weights

Range values for each objective weight have been inputted to the ACO algorithm so as to set relative relevance of objectives according to the suggestions provided by the industrial practitioners familiar with the assembly line (Table 2). Industrial engineers working at the real plant have suggested that the assignment of operators to machines with highest WIP (*Obj<sub>2</sub>*) is considered critical to guarantee production continuity. On the contrary, although frequent operator transfers should be discouraged to avoid hand-over delays not captured by the model, *Obj<sub>3</sub>* is considered less critical. Hence, a lower average weight has been set for *Obj<sub>3</sub>*. Likewise, the assignment of operators to machines that have been idle for a long time (*Obj<sub>5</sub>*) is an objective specifically developed to avoid WIP stagnation at parallel machines

and is not deemed fundamental to ensure WIP flow; hence, the lower range values. The increment used is 0.1.

Table 2: User-defined objectives' weights.

	<i>Obj</i> <sub>1</sub> : Max no assigned of operators	<i>Obj</i> <sub>2</sub> : Assign operators highest WIP	<i>Obj</i> <sub>3</sub> : Min operator transfers	<i>Obj</i> <sub>4</sub> : Uniform distribution of operators	<i>Obj</i> <sub>5</sub> : Assign machines longest idle time
Min value	0.5	1.5	0	0.5	0
Max value	1.5	2.5	1	1.5	1

## 6.2. Workforce Flexibility

As the focus of the case study is to facilitate the assembly line supervisor in the task of assigning cross-trained operators to workstations, the efficacy of the optimization framework has been investigated under different levels of workforce flexibility. The effect of operator's flexibility on productivity performance has attracted researchers' attention (Costa et al., 2013). Vairaktarakis (2003) investigate the value of resource flexibility in the context of a resource-constrained job assignment problem, which broadly covers various scheduling problems, whereas Cesani and Steudel (2005) addressed a similar problem in a cellular manufacturing system. A review of the literature on worker's flexibility in DRC systems is presented in Xu et al. (2011); most studies reviewed conclude that incremental flexibility (i.e., flexibility level of 2, that is all workers can perform two tasks) provides the majority of system performance improvement. In this study, the value of flexibility is analyzed regardless of cost implications since a well-defined cross-training policy is in place at the company that inspired this work (hence, a multi-trained workforce is already available). Operator's flexibility is analyzed to evaluate its impact not only on throughput but also on the efficacy of the assignment approach suggested; in particular, we also assess the value of multi-objective optimization against simpler assignment patterns (see Section 7.4) based on the level of operator's flexibility considered.

Although it is current practice at the pilot company to try and train each operator for up to three tasks, it is also common to have standard assignment patterns that do not fully exploit the cross-skill levels available during a production shift. For instance, a standard assignment configuration for the line given in Figure 2 consists of having 11 operators statically assigned to a machine and one operator moving to support St3, St4, and St5. This configuration, which is characterized by low workforce flexibility (LF) will be used as a reference result in the experimental plan (see Table 3). By controlling the exploitable skills as input parameters of the simulation model, the end-user can also control the impact of workforce flexibility on system performance. Additional skills across the workforce available during a shift have been introduced to generate another two experimental operator flexibility levels, medium flexibility (MF) and high flexibility (HF). The resulting skill matrices (i.e., skills applicable to each operator) for the three flexibility levels are reported in Table 3; the LF flexibility level presents 14 skills in total, MF 16 skills, and HF 19 skills. Skills are counted as the total number of workstations to which operators can be assigned

(i.e., they have received training and assignment is allowed). Note that in Table 3 St0 represents St0M0 and St0M1, etc. (see Figure 2).

Table 3: Operator cross-trained.

Operator	Cross-Trained		
	LF	MF	HF
Op_1	St0	St0	St0
Op_2	St0	St0	St0, St4
Op_3	St1	St1	St1
Op_4	St1	St1, St5	St1, St5
Op_5	St2	St2	St2
Op_6	St2	St2	St2
Op_7	St3, St4, St5	St3, St4, St5	St3, St4, St5
Op_8	St6	St6	St5, St6
Op_9	St6	St6	St6
Op_10	St7	St3, St7	St3, St7
Op_11	St8	St8	St5, St8
Op_12	St8	St8	St8

### 6.3. Selection of Ant Colony Parameters

Preliminary experiments have been run to establish the ACO parameters to use in the experimental plan. The objective of this sensitivity analysis was to find a combination of ant population size and number of generations that would robustly lead to optimal results in reasonable computational times. Full combination of four values (i.e., 5, 10, 15, and 20) for both ant population size and number of generations have been considered leading to 16 parameter combinations to simulate and a total number of ants (population size  $\times$  number of generations) ranging between 25 and 400. It is worth noting that different combinations of parameters can equal the same total number of ants (i.e.  $5 \times 20$ ,  $10 \times 10$  and  $20 \times 5$  have all the same total number of 100 ants). As the results from the ACO will be stochastic, five replications were run. Examining the MF flexibility, the average computational times across the five replications is shown in Figure 5, with similar profiles for the other flexibility. That is, there is a large increase in computational time as the total number of ants (population size  $\times$  number of generations) go from 100 to 150. Therefore, ideally the total number of ants for experimentation with the three flexibility scenarios should be kept below 100. However, further analysis is required to investigate how the different parameter combinations perform in terms of their ability to find the optimum.

To do this we used a measure which is the optimal ratio (*OptRatio*) of ants that maximize throughput (*MaxAnts*) with respect to the total number of ants (*TotalAnts*), giving:

$$OptRatio = \left( \left( \frac{MaxAnts}{TotalAnts} \right) \times 100 \right).$$

This last metric provides an indication of the level of optimality robustness with experiments ran for the three flexibility scenarios (i.e., LF, MF and HF). The results obtained showed very high optimality

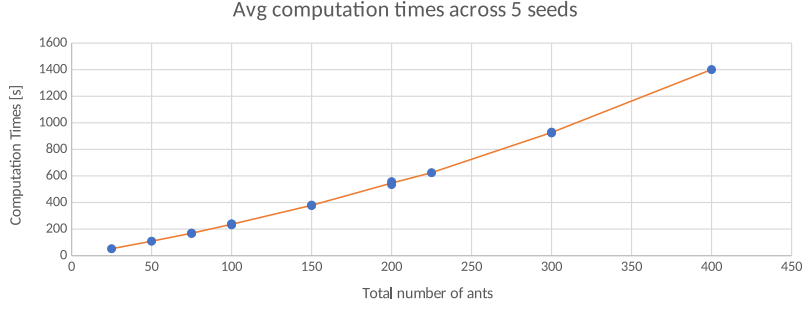


Figure 5: Average computational times across five replications for the MF scenario.

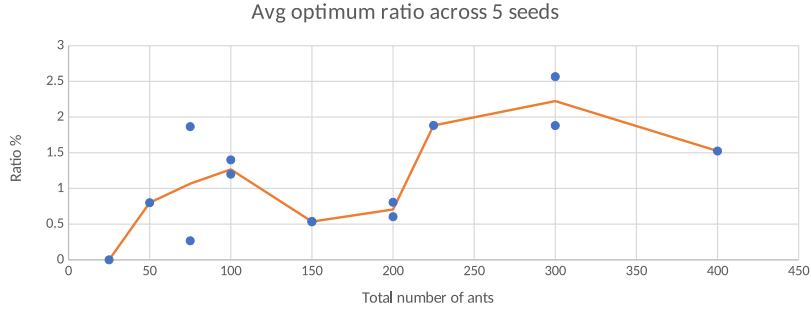


Figure 6: Average ratio of optimal ants across five replications for the MF scenario.

robustness for the HF scenario with a *OptRatio* greater than 92% across all parameter combinations, i.e., 92% of ants maximized throughput. The LF scenario also showed good ACO performance with all parameter combinations generating optimal results with *OptRatio* greater than 4% across all parameter combinations. For the MF scenario, the results for the *OptRatio* are given in Figure 6, which reports the average *OptRatio* across five replications. The blue dots represent the values observed for each parameter combination averaged across the five replications, whereas the orange line plots the average values across parameter combinations characterized by the same total number of ants. The results in Figure 6 show that 100 ants present the best performance across the parameter combinations characterized by higher computational efficiency. Therefore, we choose the total number of ants of 100 for experimentation with the three flexibility scenarios. The total number of ants of 100, can be created three times by the combination of number of generations by ants per generation of 10, 20 and 5. For the total number of ants of 100 in Figure 6, 1.4% was obtained for of  $10 \times 10$  (i.e., number of generations by ants per generation) and 1.2% for  $20 \times 5$  and  $5 \times 20$ . Using this information, we choose the number of generations of 10 and the ants per generation of 10 for the ACO experiments.

## 7. Results

The effectiveness of the optimization framework in addressing the assignment of operators is illustrated using a case study. The targets set by management within the company are deterministic values based on throughput. While simulation provides the capability of inclusion of variability within its models, we will carry out the optimization using standard times used within the company. The inclusion of variability within the assembly line for optimization, while an interesting topic, is not of interest to

the company as it uses standard times for tasks and it also sets performance goal using a deterministic value, i.e., throughput.

### *7.1. Evaluation of Low, Medium and High Workforce Flexibility*

In the real assembly line, assignment decisions are usually taken by supervisors to maximize the number of units produced during a shift (i.e., shift throughput); based on this, experiments have been initially run for an entire production shift of 9 hours. A realistic distribution of WIP across the workstations has been used to set initial system conditions, with standard break times for operators (i.e. breakfast, lunch, etc) modeled. As detailed in Section 6.3, for the ACO algorithm the population size and the number of ants in each population have been both set to 10. In order to analyze the impact of weights on throughput, the performance of each ant is also outputted by the model per single shift.

The distribution of throughput obtained across the 100 simulated ants is reported in Figure 7 for flexibility levels LF, MF, and HF, respectively. These graphs represent histograms and report relative frequencies of throughput values obtained at the end of the ACO procedure organized in small bins of size 30.

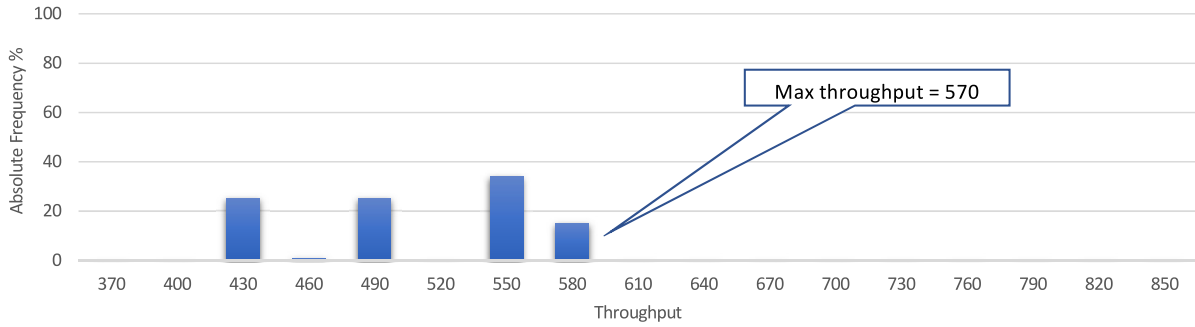
Figure 7 suggests that the choice of optimization weights is not trivial and the risk of making poor assignment decisions is possible at all levels of flexibility. However, it is evident that higher flexibility level not only generates higher throughput (i.e., 580 units, 810 units and 840 units, for LF, MF and HF, respectively) but also makes the assignment process more robust to objective weight variations. The average throughput obtained across the 100 ants increases from 501.2 units for the LF flexibility level, to 761.1 units for the MF flexibility level and 833.3 units for the HF flexibility level. Conversely, the likelihood of having smaller throughput reduces as the flexibility increases, with a standard deviation of 50.9 units for LF, 43.3 units for MF and 34.4 units for HF. The results obtained are in line with what is reported in the literature on the effect of incremental flexibility; the highest throughput increase is obtained between LF and MF whereas only a small throughput increment is observed when an additional flexibility is introduced (i.e., the HF flexibility level).

While analyzing the assignment plans, it was also interesting to note that, for the LF flexibility level, assignment decisions are made in the model loosely following the logic currently implemented at the company, that is, operator Op\_7 (see Table 3) cyclically moves between St3, St4, and St5 whereas all other operators work at the same machine for the entire shift. On the contrary, for the higher flexibility levels, no clear assignment pattern emerged; for these levels, the skill matrix cannot be immediately translated into logical assignment dynamics and an evaluation mode like simulation or queuing network becomes a fundamental tool to identify the best assignment solutions, due to the dynamic nature of WIP movement within the assembly line.

### *7.2. Sensitivity of Selected Weights*

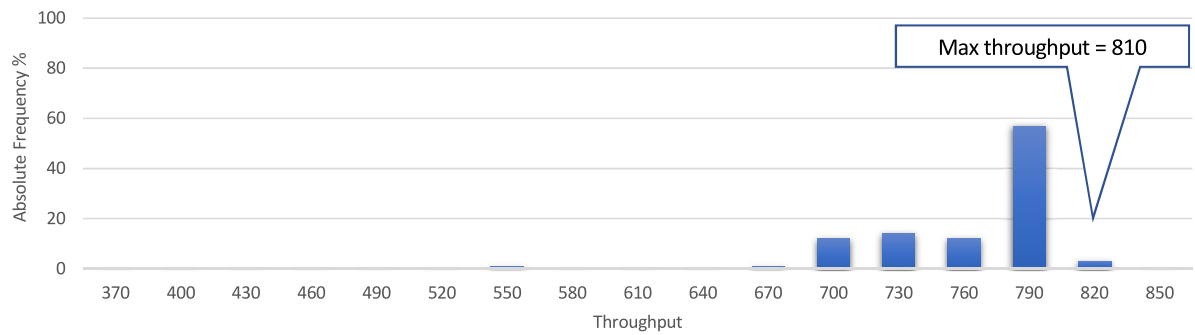
It is evident the results show that different ants, that is different combination of objective weights, have a significant effect on throughput. In particular, the MF flexibility level appears to be highly sensitive to the selected weights, for instance, Table 4 reports three different ants characterized by

### Low flexibility - 1 Shift



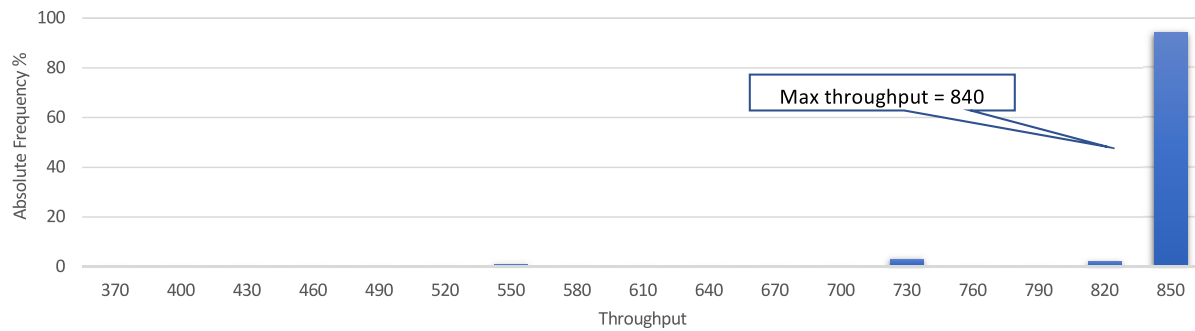
(a) Throughput distribution for LF flexibility level.

### Medium flexibility - 1 Shift



(b) Throughput distribution for MF flexibility level.

### High flexibility - 1 Shift



(c) Throughput distribution for HF flexibility level.

Figure 7: Throughput distribution for LF, MF and HF flexibility levels for 1 shift.

the same value for  $Obj_2$  (Prioritize assignment of operators to machines with highest WIP) and  $Obj_4$  (Support uniform distribution of operators across workstations) and the corresponding throughput. The lower throughput for Ant C is evidently due to the lower value of  $Obj_3$  (Minimize unnecessary operators' transfers) and the higher value of  $Obj_5$  (Prioritize assignment to machines that have been idle for long). This last objective enforces the assignment of operators to machines that may not have high WIP but have been idle for a long time in order to guarantee production continuity and avoid WIP stagnation. This objective has been specifically introduced for stations with parallel machines and for both sub-lines where it may happen that partial batches remain waiting to be completed for long times; hence, it is more suitable for longer production duration (i.e. 4 shifts) when WIP balance becomes fundamental to guarantee steady throughput or for higher flexibility levels where more frequent movements (i.e., lower weight values for  $Obj_3$ ) can be used to ensure faster production flow.

Table 4: Impact of weights on throughput

Solution	Obj <sub>1</sub> : Max no assigned of operators	Obj <sub>2</sub> : Assign operators highest WIP	Obj <sub>3</sub> : Min operator transfers	Obj <sub>4</sub> : Uniform distribution of operators	Obj <sub>5</sub> : Assign machines longest idle time	Throughput
Ant A	1.3	1.5	0.9	1.1	0.1	810
Ant B	1.3	1.5	0.6	1.1	0.1	790
Ant C	0.5	1.5	0.3	1.1	0.5	700

This is confirmed when the results for the HF flexibility level are considered. Table 5 reports the average weights for the five objectives across the “best” ants identified by ACO for both MF and HF. In absolute terms, the maximization of WIP remains the most relevant objective for both flexibility levels. However, moving from MF to HF,  $Obj_3$  tends to decrease its optimal value whereas  $Obj_5$  tends to increase it. This pattern on the relative relevance of  $Obj_3$  and  $Obj_5$  confirms what was previously noted, that is promoting operators' transfers and ensuring that all machines in the production line are regularly operated on is more effective when the workforce is more flexible. This happens because, at each assignment point, there exist various options (i.e., various cross-trained operators) for assigning an unattended machine and this can make the optimal choice smarter in the sense that the operator whose transfer will cause less disruption to line productivity can be chosen.

Table 5: Average weights for optimal ants.

Flexibility level	Obj <sub>1</sub> : Max no assigned of operators	Obj <sub>2</sub> : Assign operators highest WIP	Obj <sub>3</sub> : Min operator transfers	Obj <sub>4</sub> : Uniform distribution of operators	Obj <sub>5</sub> : Assign machines longest idle time
MF	0.70	1.57	0.90	1.23	0.10
HF	0.91	1.84	0.51	1.01	0.56

Further experiments have been conducted to verify whether weights that are optimal (/worst) for a flexibility level also prove optimal (/worst) for the other flexibility levels. A combination of weights (i.e., ants) have been selected among the best and worst ants for MF and HF and have been simulated

Table 6: Optimality of weights across different flexibility levels.  
(a) LF and HF (b) LF and MF

Flexibility Level	Throughput		Flexibility Level	Throughput	
	MF_Worst	MF_Best		HF_Worst	HF_Best
LF	540	480	LF	490	540
HF	840	840	MF	790	720

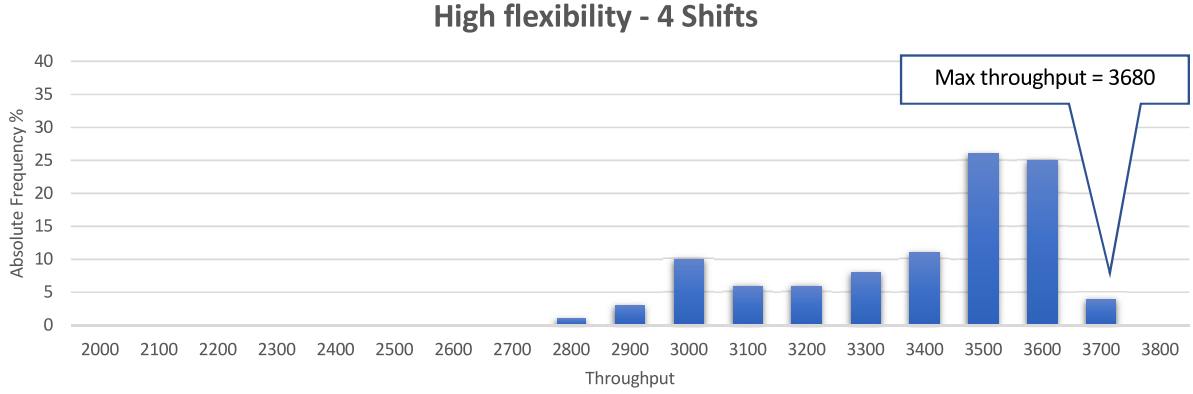


Figure 8: Throughput distribution for HF\_4S flexibility level.

for LF, MF, and HF. The results obtained are reported in Table 6. Interestingly, the worst weight combination for MF generates the highest throughput observed for HF (i.e., 840 units). Moreover, for the LF flexibility level, higher throughput is obtained when the worst weight combination for MF is applied with respect to the case when the MF best weight combination is used. Similarly, the worst set of weights observed for HF generates higher throughput for MF with respect to the case when the best weight combination for HF is applied.

### 7.3. Impact on Longer Production Duration

The impact of applying the simulation-optimization framework has also been investigated by simulating the same flexibility levels (LF, MF, and HF) for longer production duration (4 shifts which we label with 4S, giving LF\_4S, MF\_4S, HF\_4S). It is assumed that the same set of operators is available for the four simulated shifts so as to focus the analysis on the impact of production duration on throughput. The results confirm that the introduction of additional skills in the available workforce determines a significant increase in average throughput also for the 4 shift (i.e., 45.7% for MF\_4S and 57.3% for HF\_4S with respect to LF\_4S). Results for the flexibility level of HF\_4S is shown in Figure 8.

However, an optimal choice of weights for 4 shifts can boost productivity, and higher throughput can be obtained. For instance, Table 7 shows the worst, best and average throughput values obtained with the DualSimOpt results for the 4S flexibility levels and the corresponding Max Expected value which is estimated by multiplying the maximum throughput for the corresponding one shift flexibility level by four. The 8.3% best throughput increase obtained for the HF\_4S flexibility level with respect to the 1 shift suggests that the longer run perspective can impact the way shift productivity is managed (i.e.,

Table 7: DualSimOpt throughput ranges for 4S production duration.

Flexibility Level	4 Shifts			Single Shift
	Worst Solution	Best Solution	Average	Max Expected
LF_4S	2120	2340	2231.2	2280
MF_4S	2900	3410	3211.1	3240
HF_4S	2800	3640	3348.5	3360

accumulating WIP at the end of the line when one shift is run versus balancing WIP across different shifts to ensure a higher throughput over the four shifts).

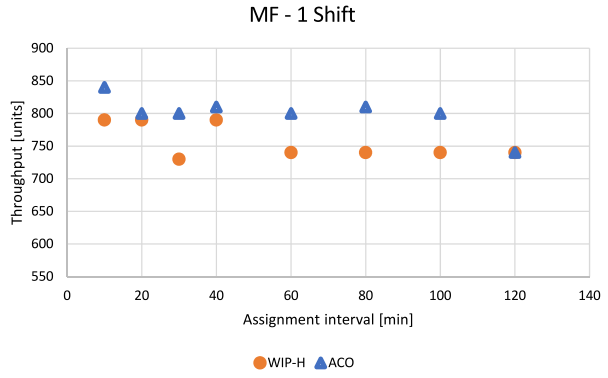
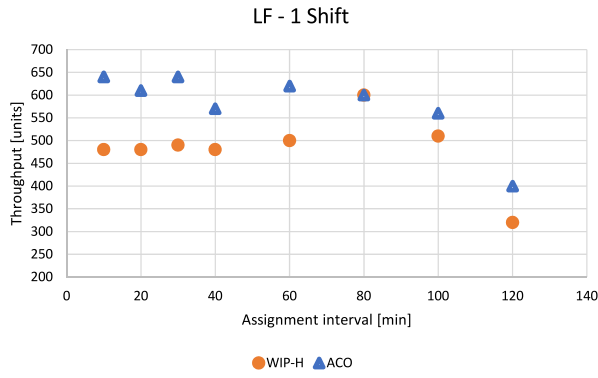
As a result of these experiments, it can be confirmed that the optimality of objectives weights is dependent on both the production duration and flexibility level and applying weights that are found to be optimal for a specific flexibility level to another flexibility level could lead to poor results, hence the relevance of the optimization approach.

#### 7.4. Comparison Against WIP-Based Heuristic

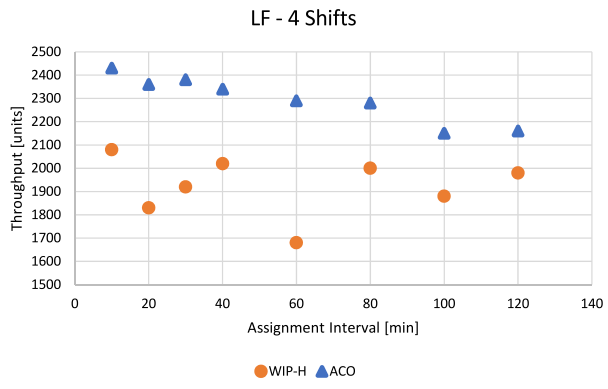
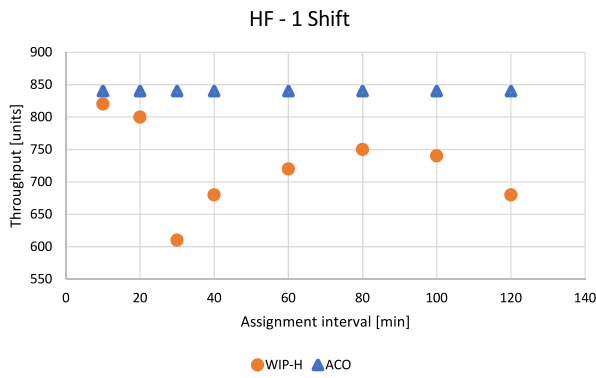
Experiments have been run to assess the efficacy of the DualSimOpt in comparison to a simpler WIP-based heuristic. Variants of the WIP-based heuristic (WIP-H) have been used in [Downey and Leonard \(1992\)](#); [Hopp et al. \(2004\)](#). The logic of the WIP-H algorithm used is:

1. The algorithm loops through the machines to identify one with the highest WIP (*maxWIP* station). In case of a tie, the one closer to the end of the production line is chosen;
2. If there is an idle operator that can handle the *maxWIP* station, he/she is assigned and the process finishes, otherwise the algorithm continues to the third step;
3. The system finds the machine with the minimum WIP (*minWIP* station), which currently occupies an operator who can operate the *maxWIP* station, then, the operator is moved from a *minWIP* station to a *maxWIP* station.

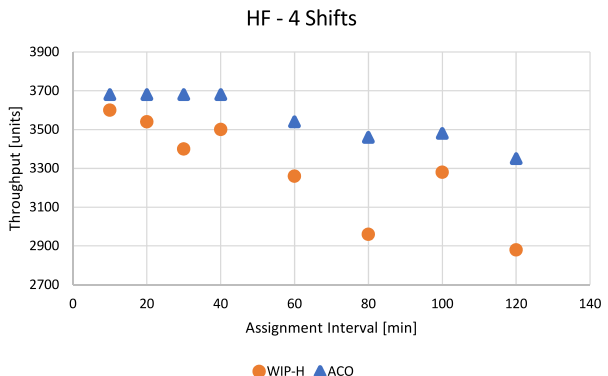
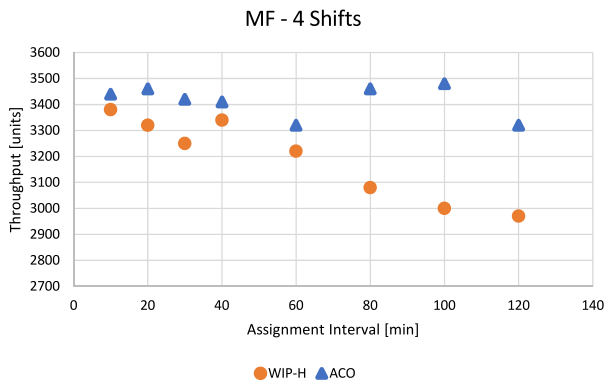
When comparing the WIP-H or DualSimOpt optimizing methods the assignment interval (see Figure 4 in Section 5.2) affects obtained solutions. Figure 9 presents a comparison of WIP-H versus DualSimOpt over different assignment intervals for single and four shifts. As previously mentioned, a 40 minute assignment interval, which is used in all other experiments reported in this paper, was deemed a good compromise between throughput performance and frequent operator transfers, also considering that inefficiencies due to frequent operator transfers (i.e., travel times, hand-over operations, set-up times, learning and forgetting effects, etc...) are not captured by the simulation model. For this assignment interval (i.e., 40 minutes) the single shift experiment results show that WIP-H always generates lower throughput than DualSimOpt for LF (19%), MF (3%), and HF(24%). For longer production duration of four shifts the efficacy of DualSimOpt for LF (16%), MF (2%) and HF (5%) can be observed. Comparing the two methods using other assignment intervals for single shifts (see Figures 9a-9c) the DualSimOpt maximizes throughput, with the exception of assignment interval of 80 (for LF) and 120 (for MF) where both values equate. For 4 shift comparisons (see Figures 9d-9f) in all instances DualSimOpt out performs WIP-H, and the magnitude of difference is largest for the lower flexibility level of LF.



(a) Comparison of WIP-H versus DualSimOpt for LF flexibility level and single shift. (b) Comparison of WIP-H versus DualSimOpt for MF flexibility level and single shift.



(c) Comparison of WIP-H versus DualSimOpt for HF flexibility level and single shift. (d) Comparison of WIP-H versus DualSimOpt for LF flexibility level for 4 shifts.



(e) Comparison of WIP-H versus DualSimOpt for MF flexibility level for 4 shifts. (f) Comparison of WIP-H versus DualSimOpt for HF flexibility level for 4 shifts.

Figure 9: Comparison WIP-H versus DualSimOpt over different assignment intervals.

Figure 9 demonstrates clearly that the assignment interval has a major effect on the outputs of both methods. The results for the WIP-H method varies, especially for HF one shift (Figure 9c) and LF four shifts (Figure 9d). These particular results have been investigated in detail but no errors could be observed. The results for the DualSimOpt method show that a longer production perspective can generate productivity improvements, especially when the production strategies used support a uniform distribution of WIP in the line. The possibility of experimenting with different assignment scenarios (i.e., different combinations of objective weights generated by the ACO) combined with the availability of more assignment options (i.e., higher flexibility), justifies the better performance of DualSimOpt.

It is also worth noting that the additional operator flexibility generates additional complexity for the combinatorial assignment problem that simple heuristics, such as the WIP-H rule, may not be able to tackle. DualSimOpt results are more robust as throughput performance do not degrade with higher flexibility levels. DualSimOpt also considerably improves throughput with respect to WIP-H for the low flexibility level, LF, in both 1 shift and 4 shifts. The reason for this result can be traced back to nature of the WIP-H rule which relies on a greedy WIP-based assignment approach. In the LF flexibility level, Op\_7 (see Table 3) should ideally be cyclically rotated across Stations 3, 4 and 5 to ensure continuous WIP flow and avoid starvation at the downstream workstations. However, the WIP-H rule might not allow Op\_7 transfers to the following workstation until a certain level of WIP is reached at that workstation. This can cause transfer delays and potential blockage and starvation issues with consequent impact on throughput performance.

In terms of computational performance, WIP-H obviously outperforms DualSimOpt. All experiments were run on a web application using cloud simulation. In this context, a precise evaluation of computational times is quite challenging as connection issues and server availability have a major impact on the time required to generate results. Approximately, for 1 shift simulation, WIP-H required less than 1 minute against 10-15 minutes observed for DualSimOpt. The assignment interval simulated also considerably impacts computational times. For the 4 Shift simulation, computational times typically observed were approximately 1 minute for WIP-H and 30-50 minutes for DualSimOpt. Conducting experiments on local processors would probably allow a fairer comparison of the computational efficiency of the two approaches. Moreover, the population-based nature of ACO would allow for parallel computing and this could drastically reduce the computational times observed for the DualSimOpt framework proposed here.

## 8. Discussion, Conclusions and Future Work

In this paper, we have developed a flexible assignment approach based on the integration of simulation modeling, BIP and ACO. The use of a BIP model whose objectives are inspired by practical assignment logic enables flexible assignment decisions based on the system “current” status, which is foreseen by the simulation model. Although the multi-objective BIP formulation proves effective in assigning operators to workstations, it does not react to changes in the system when objective weights are static. In order to overcome this, an “outer” optimization layer has been introduced. This consists of an ACO implementation that effectively searches through the solution domain (i.e., combination of objective function weights) to make the solution approach more flexible. The findings in this study reinforce the statement that a dynamic assignment strategy is fundamental to guarantee optimal assignment of operators to workstations (Uzun Araz and Salum, 2010).

In Section 7.1 we have experimented with different flexibility levels of operators on workstations with the goal of optimizing throughput. The results show that an increase in operator’s flexibility (in terms of the number of stations that an operator can operate) not only leads to an increase in throughput but also make the throughput gains more robust to optimization weights variations. Likewise, in Section 7.3, the results obtained show that a longer production duration should be adopted to increase productivity.

In general, the weighted BIP problem has proven to be effective in enhancing decision-making on the assignment of the operations and thus boosting productivity. However, various weights of the BIP that are optimal for a given flexibility level, may not be optimal when applied to another flexibility level. (see Section 7.2). This justifies the use of the “outer” ACO algorithm. The approach suggested has also been compared with an assignment model based on WIP (see Section 7.4). Results were presented for a single and four shifts across different assignment intervals. In all cases the DualSimOpt outperformed the WIP-H heuristic except for two instances where the results equate for the single shift production duration. Considerable throughput increase is achieved using DualSimOpt when a longer production duration of four shifts is used.

Considering its well-defined modular structure, the DualSimOpt framework developed here can be abstracted to a framework which could be used to solve multi-objective optimization problems where objective functions are created as weighted sums of various objectives and a system requires being evaluated based on variables dependent on system status. This is especially valid when preliminary investigations show that the solution quality is highly sensitive to objective weights and the weights optimality is also dependent on the particular case investigated as a small variation of system configuration and/or simulation settings can considerably impact system performance. More generally, this optimization approach can be used for optimizing systems that require simulation in order to capture complex dynamic elements and present decision processes that require problem-specific solutions (i.e., within the problem). Based on this, the framework could be used to address a flexible job-shop scheduling problem where two decision levels must be solved, i.e., job sequencing and job routing. The “outer” optimizer can generate alternative sequencing solutions and the “inner” optimizer can make use of heuristics or an alternative optimization approach to make routing decisions within a simulation run. Other examples where this framework can be applied is scheduling optimization in batching tools and hierarchical production planning in supply chains, which are also characterized by dual optimization problems (i.e., sequencing and scheduling or assignment).

There are several limitations and avenues for research that exceed the scope of this paper and will be subject to future work. For instance, the case study dictated that the simulation model developed should be deterministic. However, the sole deterministic view of the system can lead to misleading conclusions if variability of a system is important. A stochastic evaluation of the explored model is fundamental to identify robust solutions, especially when the system modeled is subjected to variability. Technically, if the data is adequate, it is relatively easy to manipulate a DES model in order to parameterize it with stochastic attributes. Nonetheless, considering that optimization is also involved in the solution approach, two issues arise:

- How should ranking and selection of stochastic realization of the problem be conducted?
- How can the optimizer handle the overhead of stochastic evaluation in order to remain efficient with regard to computational resources?

Indeed, having an ACO set up of  $G$  number of generations with  $N$  ants requires  $G \times N$  DES simulation models to be run in order to find a sub-optimal solution. Stochasticity will introduce a

further level of complexity since each scenario will have to be run multiple times (i.e.,  $R$  repetitions) in order to explore the effects of stochasticity and utilize a valid representative solution (i.e., average across repetitions). This makes the problem  $R$  times bigger and this computational overhead may inhibit the use of simulation-optimization. In order to reduce the problem size without neglecting the effects of stochasticity, the solution proposed by [Juan et al. \(2015\)](#) may be effective. They develop a simheuristic approach that avoids the full stochastic evaluation of each single scenario based on the consideration that solutions that perform well in the deterministic case have a higher likelihood to perform well in the stochastic counterpart. As a result, the stochastic effects are analyzed only for a selected sub-set of the evaluated solution space, that is the sub-set of sub-optimal deterministic solutions. Adaptations of the simheuristic approach to the DualSimOpt proposed here will be developed in the future.

Other possible directions for future work include:

- The way ACO is implemented, with various ants to be simulated within a generation, is amenable to parallel computing and cloud resources.
- The combination of stochastic simulation with alternative population-based optimization approaches could be explored.
- Applications of the simulation optimization approach developed to other use cases should be investigated to provide evidence that the approach is amenable to abstraction to a valid simulation optimization framework.

## Acknowledgements

The research leading to these results was supported from funding from the European Union Seventh Framework Programme (FP7-2012-NMP-ICT-FoF) under grant agreement no: 314364, the Electronic Component Systems for European Leadership Joint Undertaking which receives support from the European Union's Horizon 2020 research and Innovation Program [737459, 2017] (Productive 4.0, productive40.eu) and the financial support of Science Foundation Ireland (SFI) [16/RC/3918, co-funded by the European Regional Development Fund].

## References

- Abrams, C., Berge, Z., 2010. Workforce cross training: a re-emerging trend in tough times. *Journal of Workplace Learning* 22 (8), 522–529.
- Amaran, S., Sahinidis, N. V., Sharda, B., Bury, S. J., 2014. Simulation optimization: a review of algorithms and applications. *4OR* 12 (4), 301–333.
- Ammar, A., Pierreval, H., Elkosentini, S., 2013. Workers assignment problems in manufacturing systems: A literature analysis. In: *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*. IEEE, pp. 637–643.

- Anuar, R., Bukchin, Y., 2006. Design and operation of dynamic assembly lines using work-sharing. *International Journal of Production Research* 44 (18-19), 4043–4065.
- Azzaro-Pantel, C., Bernal-Haro, L., Baudet, P., Domenech, S., Pibouleau, L., 1998. A two-stage methodology for short-term batch plant scheduling: discrete-event simulation and genetic algorithm. *Computers & Chemical Engineering* 22 (10), 1461–1481.
- Brailsford, S. C., Gutjahr, W. J., Rauner, M. S., Zeppelzauer, W., 2007. Combined discrete-event simulation and ant colony optimisation approach for selecting optimal screening policies for diabetic retinopathy. *Computational Management Science* 4 (1), 59–83.
- Can, B., Heavey, C., 2012. A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models. *Computers & Operations Research* 39 (2), 424–436.
- Cesani, V. I., Steudel, H. J., 2005. A study of labor assignment flexibility in cellular manufacturing systems. *Computers & Industrial Engineering* 48 (3), 571–591.
- Chen, J., Askin, R. G., 2006. Throughput maximization in serial production lines with worksharing. *International Journal of Production Economics* 99 (1-2), 88–101.
- Costa, A., Cappadonna, F., Fichera, S., 2013. A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology* 69 (9-12), 2799–2817.
- Croci, F., Perona, M., Pozzetti, A., 2000. Work-force management in automated assembly systems. *International Journal of Production Economics* 64 (1-3), 243–255.
- Dagkakis, G., Papagiannopoulos, I., Heavey, C., aug 2015. ManPy: an open-source software tool for building discrete event simulation models of manufacturing systems. *Software: Practice and Experience*.  
URL <http://doi.wiley.com/10.1002/spe.2347>
- Dagkakis, G., Rotondo, A., Papagiannopoulos, I., Heavey, C., Geraghty, J., Young, P., Holland, R., 2014. From COTS Simulation Software to an Open-source Platform: A Use Case in the Medical Device Industry. *Procedia CIRP* 25, 283–292.  
URL <http://www.sciencedirect.com/science/article/pii/S2212827114010713>
- De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E., 2015. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research* 243 (1), 1–16.
- Dolgui, A., Kovalev, S., Kovalyov, M. Y., Malyutin, S., Soukhal, A., 2018. Optimal workforce assignment to operations of a paced assembly line. *European Journal of Operational Research* 264 (1), 200–211.
- Dorigo, M., Maniezzo, V., Colnani, A., et al., 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics* 26 (1), 29–41.

- Downey, B. S., Leonard, M., 1992. Assembly line with flexible work-force. *The International Journal of Production Research* 30 (3), 469–483.
- Ferjani, A., Ammar, A., Pierreval, H., Elkosantini, S., 2017. A simulation-optimization based heuristic for the online assignment of multi-skilled workers subjected to fatigue in manufacturing systems. *Computers & Industrial Engineering* 112, 663–674.
- Figueira, G., Almada-Lobo, B., 2014. Hybrid simulation–optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory* 46, 118–134.
- Hegazy, T., 1999. Optimization of resource allocation and leveling using genetic algorithms. *Journal of construction engineering and management* 125 (3), 167–175.
- Herzenberg, S. A., Alic, J. A., Wial, H., 2000. *New rules for a new economy: Employment and opportunity in postindustrial America*. Cornell University Press.
- Hopp, W. J., Oyen, M. P., 2004. Agile workforce evaluation: a framework for cross-training and coordination. *IIE Transactions* 36 (10), 919–940.
- Hopp, W. J., Tekin, E., Van Oyen, M. P., 2004. Benefits of skill chaining in serial production lines with cross-trained workers. *Management Science* 50 (1), 83–98.
- Hunter, S. R., Applegate, E. A., Arora, V., Chong, B., Cooper, K., Rincón-Guevara, O., Vivas-Valencia, C., Jan. 2019. An Introduction to Multiobjective Simulation Optimization. *ACM Transactions on Modeling and Computer Simulation* 29 (1), 1–36.  
URL <http://dl.acm.org/citation.cfm?doid=3309768.3299872>
- Inman, R. R., Blumenfeld, D. E., 2010. Assembly line team sizing with absenteeism. *International Journal of Production Research* 48 (22), 6537–6558.
- Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., Figueira, G., 2015. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives* 2, 62–72.
- Kher, H. V., 2000. Examination of worker assignment and dispatching rules for managing vital customer priorities in dual resource constrained job shop environments. *Computers & Operations Research* 27 (6), 525–537.
- Malecki, E. J., 1996. Technology, competitiveness, and flexibility: Constantly evolving concepts. In: *The transition to flexibility*. Springer, pp. 15–32.
- Malhotra, M. K., Kher, H. V., 1994. An evaluation of worker assignment policies in dual resource-constrained job shops with heterogeneous resources and worker transfer delays. *The International Journal of Production Research* 32 (5), 1087–1103.

- Morrice, D. J., Butler, J., Mullarkey, P. W., 1998. An approach to ranking and selection for multiple performance measures. In: Proceedings of the 30th conference on Winter simulation. IEEE Computer Society Press, pp. 719–726.
- Ouelhadj, D., Petrovic, S., Aug. 2009. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling* 12 (4), 417–431.  
URL <http://link.springer.com/10.1007/s10951-008-0090-8>
- Sennott, L. I., Van Oyen, M. P., Iravani, S. M., 2006. Optimal dynamic assignment of a flexible worker on an open production line with specialists. *European Journal of Operational Research* 170 (2), 541–566.
- Stützle, T., Hoos, H. H., 2000. MAX–MIN ant system. *Future generation computer systems* 16 (8), 889–914.
- Uzun Araz, Ö., Salum, L., 2010. A multi-criteria adaptive control scheme based on neural networks and fuzzy inference for DRC manufacturing systems. *International Journal of Production Research* 48 (1), 251–270.
- Vairaktarakis, G. L., 2003. The value of resource flexibility in the resource-constrained job assignment problem. *Management Science* 49 (6), 718–732.
- Xu, J., Xu, X., Xie, S., 2011. Recent developments in dual resource constrained (DRC) system research. *European Journal of Operational Research* 215 (2), 309–318.