International Journal of Industrial Engineering Computations 10 (2019) 421–442

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A study of a kanban based assembly line feeding system through integration of simulation and particle swarm optimization

Kaustav Kundu[a*], Matteo Rossini[a] and Alberto Portioli-Staudacher[a]

[a]*Politecnico di Milano, Italy*

C H R O N I C L E

Article history:
Received September 16 2018
Received in Revised Format October 7 2018
Accepted December 4 2018
Available online December 6 2018

Keywords:
Kanban
Assembly line
Simulation
Optimization
Supermarket
Part feeding

A B S T R A C T

With increase in differentiation and decreasing batch size of products, feeding the assembly line at regular intervals is considered to be a critical problem in today's manufacturing sector. Yet no clear solution has been developed for this problem; therefore, the main focus of this research is to discuss the different aspects of line feeding, the latest trend in literature, and to propose an innovative method to support solving the problem. A discrete event simulation model is developed and a mathematical model based on particle swarm optimization is used to support the simulation. The hybrid model is finally applied to practical situations. Results show how different settings of kanban influence the performance of the assembly line feeding system. The biggest novelty item is certainly the recognition of the trade-off between kanban size and number of kanban and the importance of investigating its behaviour during the design of the system.

1. Introduction

In recent years, an increase in the differentiation and decrease of batch size of final products has led to the diffusion of mixed-model assembly lines, in particular in certain sectors, such as automotive and white goods. In these scenarios the problem of feeding the line becomes especially complex. In addition, several companies nowadays are moving towards a line production configuration thanks to the increased flexibility of Industry 4.0 machines, and to the diffusion of lean management principles (Portioli-Staudacher & Tantardini, 2012). Lean management leads to carry fewer and fewer inventories, but too few inventory can lead to the entire factory becoming idle. On the other hand, large volumes of stocks stored near the line hide problems, can generate interferences with the work of the operators and decrease safety. Therefore, the fundamental problem to be solved consists in how to bring each component to the line only at the moment it is needed and in the quantity it is needed. The introduction of decentralized storage areas, called supermarkets, gave the possibility to have frequent deliveries of small batches of components (Emde & Gendreau, 2017). Supermarkets are located between the line and the central

* Corresponding author
E-mail: kaustav.kundu@polimi.it (K. Kundu)

2019 Growing Science Ltd.
doi: 10.5267/j.ijiec.2018.12.001

warehouse. Their objective, other than to reduce the refilling lead time, is to enable a change of stock keeping unit, from a full pallet to a bin or, ideally, to the single piece (Sali & Sahin, 2016). Moreover, to foster the continuous improvement efforts aimed at reducing inventory level, the kanban card system can be adopted: the movement of stocks towards the line is authorized only when the parts are needed. Kanban is the Japanese word for card. A kanban is usually a piece of paper, contained in a rectangular vinyl envelope and attached to a parts container (Vatalaro & Taylor, 2005). It is one of the popular lean techniques that is carried around the factory, either on its own or attached to components, by operators known as mizusumashi, which can be translated to whirligig or water spider (Monden, 2011).

Following the current trend in literature, the areas, where kanban is applicable, are: Production; Refilling: Supply chain; Variations. But it is noticed that little study has been done on line refilling in comparison with the other three categories. This is one of the reasons that made it become the focus of this research. The dimensioning of this kind of refilling system, albeit often carried out without adopting a specific technique, has recently attracted the attention of several researchers (Lolli et al., 2016; Sali & Sahin, 2016; Emde & Schneider, 2018), keen to provide a comprehensive design methodology. This work fits in this last description, aiming at expanding some of the gaps found in the existing literature. The research questions of this paper are:

RQ1 How and to what extent does kanban sizing improve the performance of assembly line feeding system?

RQ2 Is the proposed algorithm giving better performances than similar existing ones?

To answer these research questions, a discrete event simulation model is created in order to strengthen the bond between the developed methodology and real life. Recently, there are few studies (Roukya et al., 2019) where simulation along with optimization is proved to be efficient than other methods. Therefore, in conjunction, a meta-heuristic algorithm is written to find the values of each variable that minimizes the total cost function. This methodology is also tested through its application to the real case of a manufacturing company.

The remainder of this paper is organized as follows. Section 2 states theoretical background as well as methodology. Section 3 then outlines the simulation model followed by the algorithm. The results of the hybrid model to practical applications are presented in Section 4. Finally, the paper concludes with Section 5, where a discussion on future research directions is provided.

2. Theoretical background and Methodology

The line refilling is one of the major areas where kanban is extensively used. But the majority of this category's researches are quantitative. One of the reasons for the small number of theoretical papers could be the fact that this field uses many concepts, such as kanban, which have already been treated extensively in the traditional literature. The upcoming section presents a glimpse on different aspects of this topic.

2.1 Theoretical background

The central aspect in dimensioning a kanban system is the determination of the number of cards. The formula used by Toyota Motor Corporation to determine the number of kanban is called Toyota formula (Sugimori et al., 1977):

$$y = \frac{D\left(T_w + T_p\right)(1+\propto)}{a} \tag{1}$$

where,
y = number of kanban;
D = demand per unit of time;

T_w = waiting time of one kanban;
T_p= processing time of one kanban;
a = container capacity (not more than 10% of daily demand);
$\propto$ = safety factor. It is policy variable (although < 10%).

Despite the fact that it was created for the production system, it has also been used to solve the feeding dimensioning problem (Faccio et al., 2015; Faccio et al., 2013a; Faccio et al., 2013b). In the context of internal logistics however, its usage is limited by the fact that the optimal number of kanban solves only the trade-off between inventory and shortage costs, without considering a further cost item that is, handling costs (Lolli et al., 2016). Moreover, the parameters used are functions of other decision variables or other parameters (Faccio et al., 2013a), such as the number of water spiders. Water spider is an essential component required for line filling. Water spiders can operate according to either a fixed quantity, variable time (Lolli et al., 2016) or a fixed time, variable quantity policy (Hanson & Finnsgard, 2014). These two systems correspond respectively to the reorder point and periodic review policies (Monden, 2011) found in the traditional logistics literature. For an overview of the pros and cons of each solution, the reader may refer to Ballou (2004). This research utilizes the fixed time, variable quantity solution and does not investigate the differences with the other policy. This could however be the subject of future research.

There are many quantitative methods, which are used for solving line feeding problems. Simulation, while being the most used modelling approach in literature about production (Kumar & Panneerselvam, 2007; Hao & Shen, 2008), has not been taken in great consideration. Lolli et al. (2016) modelled the system using the queue theory; simulation is then used to find the most cost effective solution, in terms of minimum number of water spiders required to avoid stock-out. Faccio et al. (2013b) created a procedure that first computes analytically the number of water spiders and kanban and then, through simulation, establishes the best delivery frequency. Moreover, one of the conclusions of the aforementioned research is that the high impact of the short-term loading variables like the tow train capacity and the refilling interval demonstrates that use of an analytical approach alone may be unable to obtain reliable results. This is one of the reasons that led to the adoption of a simulation model for this research. Emde et al. (2012), Emde and Boysen (2012a) and Emde and Boysen (2012b) created a framework for the design of a supermarket-based refilling system and then tackled four different problems in one of the three articles (Emde and Boysen (2012a) - problem 1, the decision regarding the number and the location of supermarkets; Emde and Boysen (2012b)- problems 2 and 3, vehicle and inventory routing; Emde et al. (2012)-problem 4, short-term loading).

Location of supermarkets had already been treated by Battini et al. (2009), who created a two-tier framework to jointly decide about the decentralization of parts' inventory and how they are fed to the line. A similar work has been done in Battini et al. (2010), where the choice is extended at each component's level. Caputo and Pelagagge (2011) arrived to a similar conclusion, stating that adopting the same feeding policy for all components may not be the most cost effective solution. The comparison of different feeding policies has been relatively well covered. Limere et al. (2012) proposed a mathematical model to help with the investigation and the selection of the best one. Golz et al. (2012) compared traditional kanban feeding with custom-scheduled tours: their solution performed slightly better than traditional kanban refilling, although being worse in terms of computational time. Sali et al. (2015) deepened the comparison with the addition of sequenced bins, which guarantee the best cost performance when dealing with large items and high component diversity. Faccio (2014) investigated the breakeven points of different feeding policies according to the variations in production mix.

Savino and Mazza (2015) investigated the implications of the line's layout on the refilling procedure, concluding that O-shaped layouts would require a lower amount of components' stocks to function. Faccio et al. (2013a), utilizing a cost minimization model, demonstrated how the application of the classical kanban number calculation (e.g. using the Toyota formula) can be unable to bring about substantial results without an integrated approach. Moreover, they advised for further research on the

supermarket topic, since it is regarded as an emerging research topic. Faccio et al. (2015) devised an analytical cost model with the novelty of distinguishing among parts and assembly lines, overcoming the simplifying hypothesis assuming a constant lead time for all parts. Mathematical models are very used as well. Choi and Lee (2002) combined one with an algorithm in order to generate the optimal refilling schedules for an automotive assembly line. Satoglu and Sahin (2013) created a mathematical model to bring components to the line. This solution includes the pre-determination of the quantities to be shipped in each tour. A real case application is also shown and the most noticeable impact is the considerable reduction of the time the line is idle, waiting because of a shortage. Finally, also queue theory can be used to solve this kind of issue: Gamberini et al. (2013) modelled the system using this technique and then based their solving approach on the application of the Erlang-C function, computing the probability that a kanban had to wait before refilling.

Different articles used different variables in order to assess the performance of line feeding systems. Among the variables, the number of kanban and number of water spiders are included into this work because of their almost universal utilization in literature. Service level has not been picked since it is considered as a performance measure. This is done to potentially increase the scope of application of the model: in a scenario where all costs have similar values, it may be useful to evaluate the trade-off between stock-out allowed and (handling + inventory) costs. Refilling interval, as used by Faccio et al. (2013b), is defined as the minimum interval between a vehicle trip and the following one. Therefore, for analytical purposes it can be seen as a consequence of number of water spiders.

One of the novelty items of this research is the consideration of kanban size alongside number of kanban, which has never been done in feeding literature, although suggested by Battini et al. (2015).

It can also be seen how it is fairly standard in literature to consider the three measures: inventory cost, stock-out cost and handling cost. Some articles do not contemplate the possibility of stock-out (service level must be 100%) and thus do not need to measure stock-out costs. The only one that proposed a different classification is Faccio et al. (2013b), who uses jointly workstation utilization and number of tours/day to represent handling costs. It can be assumed that the reason behind this choice is the desire to measure performances independently from monetary values. This hypothesis is backed by the fact that also the other two measures are considered as quantities. Since one of the novelty items of this research is the consideration of the dimensions of intra-station buffer as parameters, it becomes necessary to introduce a new performance measure: the WIP cost. Just as in refill literature, inventory and stock-out costs are the most popular. A larger overview on this aspect is given by Kumar and Panneerselvam (2007). Their conclusions are represented in Fig. 1, where measures have been clustered in categories created in order to facilitate the comparison with refill literature.

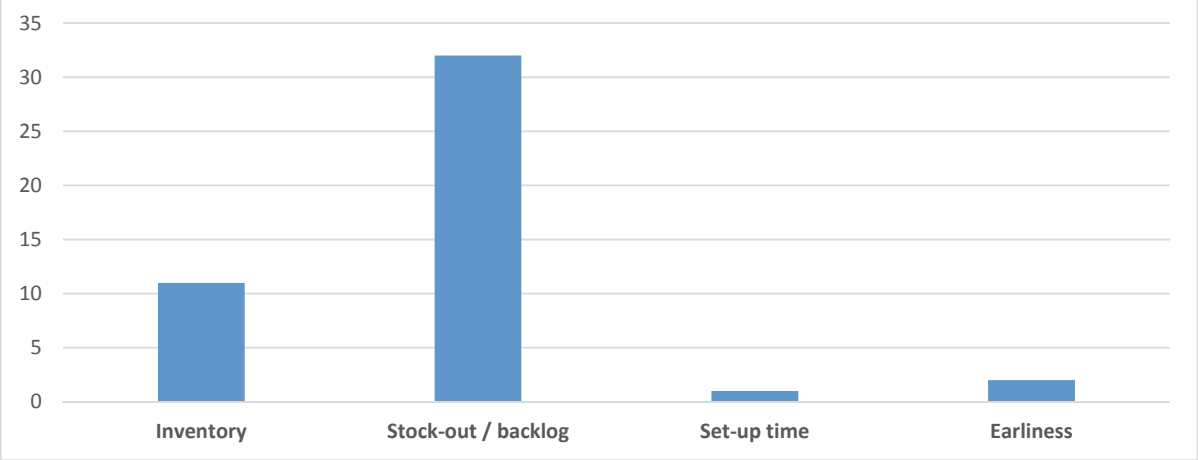

Fig. 1. Performance measures used in production literature and their frequency of appearance. Source of the data: Kumar and Panneerselvam (2007).

First of all, it's immediately possible to notice how the use of inventory and stock-out costs is widespread. Two measures are exclusive of this field of research, set-up time and earliness. The latter, in accordance with JIT principles, can be seen as the dual of stock-out costs. Moreover, according to Ohno (1988), parts arriving earlier than needed go against the principles of waste elimination. In any case, one can conclude that their usage is only case-specific, given the very few number of times they've been utilized. Moreover, recently Emde and Schneider (2018) pointed out that companies are more focusing on just-in-time solution rather than keeping the inventory for a longer period. They proposed a solution to this problem by building a mathematical model whose objective is to reduce the number of vehicles and total route duration. But they assumed that the service times at the stations or refilling time are independent of the number of bins. To the best of our knowledge, there is no study which considered the dependency of the refilling time on the number of bins. In view of this statement, this study considers that refilling time depends on the number of bins. The methodology used in this paper is discussed in the next section.

2.2 Methodology

As mentioned previously, simulation is chosen to solve this problem for two main reasons: it is the most used in production literature; Kumar and Panneerselvam (2007), Hao and Shen (2008), Faccio et al. (2013b) concluded that the sole analytic approach may be unable to obtain reliable results. In particular, discrete-event simulation (DES) is adopted as the method of choice. Examples exist in JIT-kanban literature that support this choice. For instance Albino et al. (1995) used DES to validate their Markovian model. Another case of DES application to simulate a production line is given by Hao and Shen (2008). The approach followed in this research pairs the discrete event simulation with a meta-heuristic search algorithm, what is described as a hybrid technique (Aghajani et al., 2016). Heuristic algorithms are methods used to find a solution to complex problems: they sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time. Meta-heuristics are an evolution of these algorithms, since they combine the basics of traditional heuristics with higher level frameworks that are able to guide the search and make more efficient the exploration of the solutions' space (Blum & Roli, 2003, Toncovicha et al., 2019). In JIT-kanban literature, the usage of said algorithms to solve complex problems is quite widespread: some examples are in Widyadana et al. (2010) and Houand Hu (2011). Hybrid approaches have also been used to dimension productive systems (Bowden et al., 1996); Azadeh et al., 2010; Aghajani et al., 2016; Rouky et al., 2019). The most widely used algorithms for meta-heuristic optimization are genetic algorithm (GA), tabu search (TS) and simulated annealing (SA) (Kumar and Panneerselvam, 2007). A relatively new method, particle swarm optimization (PSO), is starting to be used also in the JIT-kanban field, as can be seen in Aghajani et al. (2016) and Cao and Li (2013). PSO was first introduced by Eberhart and Kennedy (1995). It is based on socio-psychological principles: a population of particles that, interacting between each other, improves the problem's solution over time (Kennedy, 2011; Pradeepmon et al., 2018). Aghajani et al. (2016) compared the performance of PSO and SA: their findings show that PSO is able to reach solutions closer to the optimum than SA's and thus is better suited for large problems. This reason, combined with its novelty in comparison with other methods, led to the choice of PSO for this work. While the algorithm in this research is written without following a specific piece of code, the two formulas used to update particles are the ones that Kennedy (2011) described as the most common type of implementation. The values of the two parameters are chosen in order to keep particles under control, in accordance with the findings in Clerc and Kennedy (2002). Originally, in fact, the particle swarm needed the imposition of a maximum velocity, so that particles would not assume unrealistic values. This has been made unnecessary by properly tuning the two aforementioned parameters (Kennedy, 2011). Looking at the population, two main types of PSO exist: gbest and lbest. The former considers each particle as linked with all the other ones, while in the latter each particle has its own different neighbourhood (Eberhart and Kennedy, 1995). Because of the way it's built, the gbest can be prone to end up in a local optimum. On the other hand, the lbest spreads slower, but sub-populations may search diverse regions of the search space in parallel. This increases the probability to end up near the global optimum (Kennedy, 2011). Because of this reason, lbest is chosen for this research. Finally, the number of particles is set in

accordance with the range [10; 100] defined by Kennedy (2011); after a series of trials, it is found that a population of size 40 would yield the highest performance in terms of speed of convergence to the solution.

3. Simulation model and algorithm

In this section, simulation model is described followed by the algorithm.

3.1 Simulation Model

3.1.1 Assumptions

For the model to be considered valid, the following assumptions must be made.

1. Kanbans are considered empty according to the bottom of container rule, meaning that the kanban card is made available for the water spider's pick-up as soon as the container is emptied. This solution is chosen as it represents a good trade-off between the system's inventory level and the possibility of visually controlling stocks (Vatalaro & Taylor, 2005).

2. The line is divided into independent blocks; each block is resupplied independently from the others. A similar assumption is used in Emde and Boysen (2012b).

3. Congestion (in terms of traffic of operators) is not an issue in the factory. This is coherent with most of the articles in literature, since they do not mention this issue.

4. The assembly operator sees that the container is empty only when collecting a new component (Hobbs, 2003).

5. Express kanban are prioritized according to the order the water spider encounters them. This aspect is not mentioned in literature. It is however logical to assume that, in a situation of complete stock-out, components needed upstream are the most urgent, since without them the downstream stages would be idle anyway.

6. All service times are exponentially distributed, in order to reflect the fact that being late is much more probable than being early. This assumption is backed by a similar one done by Karmarkar and Kekre (1989).

7. Collection/distribution of bins in the line can only be done by one water spider at a time. This assumption is needed for modelling purposes; as will be shown later though, it does not affect the optimality of the solution.

8. One kanban strictly corresponds to one bin. This assumption is the most common in refill literature (see for instance Lolli et al., 2016).

9. Scrapped products are sent to a different station, where they are reworked and ultimately completed. This is an established practice in the manufacturing industry (Heng et al., 2017).

10. The refilling time increases with decrease in the size of the bins. In this research, a smaller size bin implies that the water spider, each tour, will pick up more kanban. This produces an increment in service times. This assumption is also used by Lolli et al. (2016): they stated that the water spider's service time depends on the number of kanban, while it's independent from the size of the SKU.

11. Water spiders circulate in the supermarket according to a traversal policy (Koster et al., 2007).

The following is a list of all the notations used in this research.

Decisional variables

- nrkbpn = number of kanban for part n

- nrws = number of water spiders
- szbipn = size of the kanban for part n

Parameters

- (*ID, k*) = component ID of piece k
- ct = fake cycle time of the line [sec]
- d_{rack} = depth of a rack in the supermarket [m]
- l_{aisle} = length of an aisle in the supermarket [m]
- $l_{extra\ bit}$ = length of one of the pieces of supermarket not travelled by the water spider [m]
- $l_{extra\ tot}$ = total length of the pieces of supermarket not travelled by the water spider [m]
- $l_{first\text{-}smk}$ = distance between the supermarket and the first station visited by the water spider [m]
- $l_{last\text{-}smk}$ = distance between the supermarket and the last station visited by the water spider [m]
- n_{aisles} = number of aisles in the supermarket
- $n_{b/in}$ = number of movements of the water spider between the stations
- n_{cart} = cart capacity [components]
- n_{comp} = number of different components in the system
- $n_{w/in}$ = number of movements of the water spider within the single station
- spd = speed of the water spider [m/s]
- $t_{b/in}$ = time spent by the water spider in a movement between stations[sec]
- $t_{extra\ tot}$ = total extra time included in the approximation [sec]
- t_{line} = total time the water spider spends in the line [sec]
- $t_{linebins}$ = total time the water spider spends managing bins in the line[sec]
- t_{linefx} = fixed unitary loading/unloading time for a bin in the line [sec]
- $t_{linetravel}$ = total time spent by the water spider travelling along the line [sec]
- $t_{linevar}$ = variable unitary loading/unloading time for a bin in the line [sec]
- t_{rfl} = total time needed to a water spider to refill all the bins [sec]
- t_{rflfx} = fixed unitary loading/unloading time for a bin in the supermarket [sec]
- t_{rflvar} = variable unitary loading/unloading time for a bin in the supermarket [sec]
- t_{sim} = total time of the simulation [sec]
- t_{smk} = total time the water spider spends in the supermarket [sec]
- $t_{smktravel}$ = total time spent by the water spider travelling in the supermarket [sec]
- $t_{w/in}$ = time spent by the water spider in a movement within the station [sec]
- t_{wup} = time required for the initialization of the model [sec]
- timereq = time required by a water spider for a complete trip
- w_{aisle} = width of an aisle in the supermarket [m]
- w_{smk} = total width of the supermarket [m]

Costs

- $c_{backlog}$ = total backlog cost [unit/h]
- c_{sp} = unitary holding cost for component p [unit/(h*pc)]
- $c_{salaryws}$ = hourly cost of a water spider [unit/(h*operator)]
- c_{stocks} = total holding costs for the components [unit/h]
- c_{totws} = total cost of water spiders [unit/h]
- $c_{unitarybacklog}$ = unitary backlog cost [unit/(h*unit)]
- c_{wip} = total Work In Progress cost [unit/h]
- c_{wipn} = unitary cost for the WIP in stage n [unit/(h*pc)]

Particle swarm

- $\propto$= Particle swarm parameter
- β = Particle swarm parameter
- b_{pv} = value of variable v in the best solution of particle p

- bn_{pv} = value of variable v in the best solution of particle p's neighbourhood
- p_{pv} = current value of variable v in particle p
- v_{pv} = current velocity of variable v in particle p

3.1.2 General Overview

This model is intended to simulate a mixed-model assembly line. In accordance with assumption number 2, the whole line can be seen as divided into blocks by buffers; each block is resupplied independently from the others, meaning that a water spider can serve only stations belonging to the same block. This plant has a central supermarket, where it is assumed congestion is not an issue, following assumption number 3; for instance, there are no interferences between water spiders when moving or retrieving components. Water spiders operate according to a fixed time, variable quantity policy. They move parts through a cart on wheels, by either pushing or pulling it. At each station, bins are put in inclined racks, made of two shelves: the top one is inclined towards the operator and receives full bins; the bottom one has instead the opposite inclination, facilitating the collection of empty ones (Faccio et al. 2013a; Battini et al., 2013). The boxes are moved from the top to the bottom shelf by the operator, who notices the empty bin only when he is looking to retrieve a component from it (in accordance with assumption 4). It is also assumed that all assembly operations are already divided between stations in order to even the workload among all operators. The stations taken into consideration use eight components: for the sake of simplicity, they are identified by a number from 1 to 8. Two variants of the same product are assembled, differing only in the parts they are made of. Below is presented the composition of both variants; numbers represent the IDs of the components, while horizontal lines indicate a change of station.

$$1°2 - 3°4 - 5°6$$
$$1°7 - 3°8 - 5°6$$

3.2. Algorithm

The size of the problem at hand leads to the decision to write a meta-heuristic algorithm. The other viable alternative is a full factorial approach (Lolli et al., 2016; Hou & Hu, 2011). In the case presented in this research this would have meant performing billions of experiments. As explained in section 2, among the many meta-heuristic algorithms used in literature, the Particle Swarm Optimization (PSO) method is selected for this research. The PSO algorithm used in this work consists of 40 particles used to explore the n-dimensional space (where n is the number of the problem's variables) in order to find the optimal solution. Particles are organized in rows and columns. Each of them has 4 neighbours, following a von Neumann neighbourhood of range = 1 (Toffoli & Margolus, 1987). In order to provide each particle with a neighbour, the population is wrapped in toroidal shape. Each particle is associated with a specific position in the n-dimensional space. This location changes each iteration, based both on the particle's speed and on the influence of its neighbours. The objective function through which each particle is evaluated is made up of the sum of all the relevant costs. Below a mock of the algorithm can be found, written in pseudo-code.

```
{
model initialization
for n = 1: (number of iterations)
      for  y = 1 : ( number of particles)
            insertparticles'variables in the model
            run the model
            compute objective function
                  if result < ( particle's best )
                      update particles' best
                  end
       end
       if ( number of iterations) < (maximum number of iterations)
            update particles' positions
       end
end
}
```

For easiness of use, as well as clarity in the presentation, the algorithm is structured on several levels. Each action (or set of actions sometimes) has been enclosed in a sub-program, which can be called and run whenever required.

Some clarification is needed on the unitary WIP cost. Focusing on cadded-stage1, which represents the items in the first buffer (between stations 1 and 2), the parameter can be seen as made up of two parts: the first is tied to the value of the components added in the stage, while the second represents the cost of the assembly operator during the cycle time required him to assemble the pieces. In order to represent the fact that multiple variants are being assembled on the same line, the first part of the formula accounts for the value of all the parts used by the operator, weighted accordingly with the production mix. The operator's cost is then multiplied by (0.0011/8), which is the average common hourly holding charge (Lolli et al., 2016). The logic behind the computation of caddedstage2 is exactly the same, with of course a change in the component types.

Finally, c_{wip1} is obtained by adding the value of the piece coming from upstream, multiplied by the average common hourly holding charge, to cadded-stage1 previously computed. c_{wip2} is obtained similarly, but also including in the sum cwip1. Lastly, the number of kanban is checked again in conjunction with the corresponding kanban size, in order to prevent the number of parts to be higher than the limit.

The computation of the maximum number of water spiders allowed is based on three conditions:

- Time spent in the line. Assumption 7 states that, due to the design of the model, no more than one water spider can be collecting or distributing bins in the line at the same time. Thus the maximum number of water spiders is given by timereq/t_{line}. Given the time penalty applied for smaller sizes of the bins, as explained in section 3.1, the computation is done considering the worst case scenario, where every component's kanban size is the lowest available.
- Distribution delay. Since only one block of this type is present in the model, all components have to leave before the parts carried by the next water spider arrive. The formulas are the same ones that define the service time in the distribution delay block, accounting for $t_{linebins}$, as well as $t_{w/in}$ and $t_{b/in}$. Also in this case the worst case scenario is picked.
- SMK blocks. Following the same principle as above, parts carried by different water spiders must not be mixed. The formula accounts for the whole t_{smk}; the result is divided by 10, which is the current number of SMK blocks in the model.

The application of the hybrid model to practical applications is discussed in the next section.

4. Practical applications

4.1 Source of the data

A lot of effort was put into searching the literature for sensible values for the main parameters. The main problem encountered was the fact that many articles that include practical cases omit purposefully either part or all of the data used as input. Moreover, talking about costs, some values were discarded because either outdated or in another currency with no time reference to track the exchange rate, and thus not completely comparable with the others. The ratio of the different costs has been used in order to allow for a better comparison between the different cases. Lolli et al. (2016) presented the mean daily stocks holding cost. Assuming one 8-hour shift per day, the c_s (mean daily stocks holding cost) is computed by making the average of different components and dividing the result by both the number of working days per year and the number of daily working hours, both included in the text. The same procedure is applied to obtain $c_{salaryws}$. Regarding $c_{unitarybacklog}$, it is obtained by multiplying the average stock-out cost per component by the average number of components per product. The article provides the yearly stocks holding cost percentage, alongside the value of each of the components considered. Thus, c_s is computed

by multiplying the holding charge by the average value of components; the result is then divided by the number of working days in a year and by the number of working hours per day. $c_{salaryws}$'s computation is immediate, given that the yearly cost is included. In this case the cost of backlog, given the data available, has been computed as the margin lost because of the missed sale. First of all, for each of the product's variant the production cost is computed using both the bill of materials and the components' values. Then, $c_{unitarybacklog}$ is obtained by making the average, across all product variants, of the difference between the model's final price and its production cost.

Lolli et al. (2016) used a service time that includes both refilling and travel times. This parameter is considered to be dependent only on the number of kanban carried by the water spider. The assumption that refilling time is the only one that changes was needed to compute the desired parameters. The authors provided a table with service times for incremental number of kanban. By computing the average difference between these values it is possible to obtain bin refilling time. To obtain the unitary value, the average SKU size is computed by dividing the coverage time of an inline SKU by the average part consumption rate. $t_{smktravel}$ is instead found by subtracting this item from the lowest service time value. In article by Faccio (2015), piece refilling time is computed by dividing the given bin refilling time by the average bin size, provided as well in the article. In order to compute $t_{smktravel}$, the only useful element provided is the total distance travelled by a water spider during a complete tour. Assuming that the length of the path within the supermarket is half of the total, $t_{smktravel}$ is found by dividing it by the water spider's speed. The computation of Piece refilling time used in the article by Faccio (2013a) is the same used in Faccio (2015). The computation of $t_{smktravel}$ is more elaborate, having at disposal only the total distance travelled and a scaled picture of the water spider's route. The ratio between the part of the route in the supermarket and the one outside is obtained by counting the length of the paths in the image, using pixels as the unit of measure. With the ratio and the total length in meters, it is easy to obtain the length of the supermarket path and ultimately $t_{smktravel}$, dividing it by the speed of the water spider. For the detailed data, the readers are referred to the articles by Lolli et al. (2016) and Faccio (2015). Three experimental factors, kanban size, number of water spiders and t_{rflvar} are chosen. The experimental design is shown in Table 1.

Table 1
Ranges for different experimental factors

	Kanban size	Number of water spiders	t_{rflvar}
Range	5,10,15,20,25,30,35,40,45,50	2,3,4,5	2,4,5

Considering all the possible combinations of different experimental factors, the total number of experiments considered in this paper is 120. The performance measure used here is the inventory or stock levels. After initial testing, the size of the swarm is set at 40 particles and the number of iterations is fixed to 150. During each iteration, particles explore the space, taking into account also the position of each of their neighbours. Their objective is to minimize the total cost function, which includes the costs of inventory, stock-out, handling and work in process. The simulation is run for a period of 7200 seconds with warmup period of 253 seconds. Each scenario is replicated for 40 times in order to obtain more generalised results. The results are discussed in the next section.

4.2 Results

The goal of this section is to show the functioning of the proposed algorithm: how it balances between the values of the different variables in order to pursue the optimal solution. First of all, the trade-off between kanban size (t_{rflvar}) and kanban number is considered. Unitary kanban size would be the ideal solution, since it allows for higher sensitivity. In a real life scenario though, this solution presents some problems, which are not easily quantifiable, and in particular very high number of kanbans (and containers) to manage. As explained in assumption 10, this research locates this complexity in the refilling of the bins in the supermarket.

In the interest of investigating the pattern of this trade-off, the best solution for each size of the kanban has been computed for different sizes of the penalty: that is for different values of t_{rflvar}. In this work best solution is the one with the lowest inventory level that manages to avoid stock-out. For the sake of better highlighting the difference between the different solutions, the same kanban size is used for every component. The graph below presents the results of this analysis. Please notice that the axis values represent the reduction of the inventory level in comparison with the base case, the one with kanban size of 50 pieces. For further clarity, all the mentions of inventory level, or stocks level, in this section refer to the maximum number of stocks in the system.

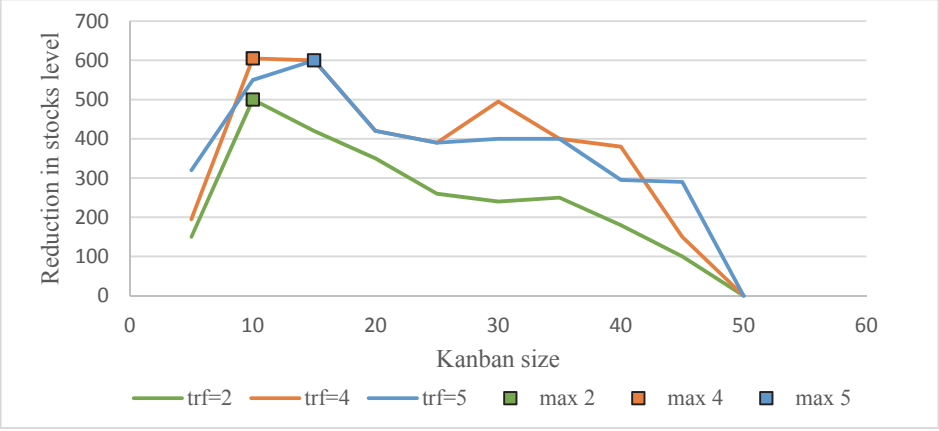

Fig. 2. Reduction in stocks level changing kanban size, for different values of t_{rflvar} (here simply called trf).

The key conclusion that can be deduced from the Fig. 2 is that, increasing t_{rflvar} (and thus the penalty), the kanban size of the best solution increases. It is possible to notice how the system seems to work better with a t_{rflvar} of 4, since water spiders' saturation is higher and there is also the biggest savings in terms of stocks. The first part represents the fact that, increasing t_{rflvar} from 4 to 5, it is necessary to add another water spider to the system, thus reducing the overall saturation. Since with t_{rflvar}= 4 there are more stocks in the system than with t_{rflvar}=2, it is plausible that also the number of stocks reduced will be higher in the first scenario. Considering instead t_{rflvar}= 5, a possible explanation may be given by the fact that the overall refilling time is so big that it prevents a further reduction of the inventory level. The second trade-off in the determination of the best solution is the one between level of stocks and number of water spiders. Intuitively, a higher number of water spiders should allow the system to work with a reduced number of stocks.

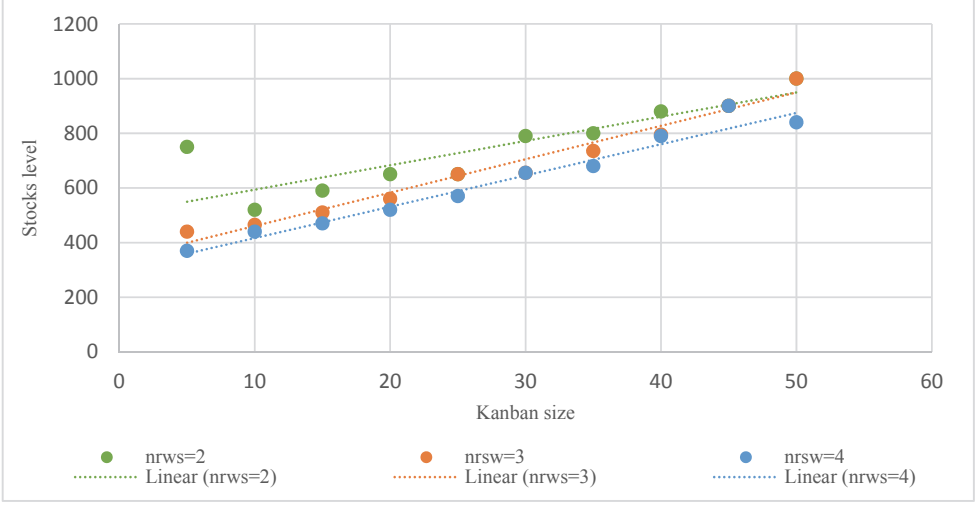

Fig. 3. Stocks level for different kanban sizes and water spiders, when t_{rflvar} = 2

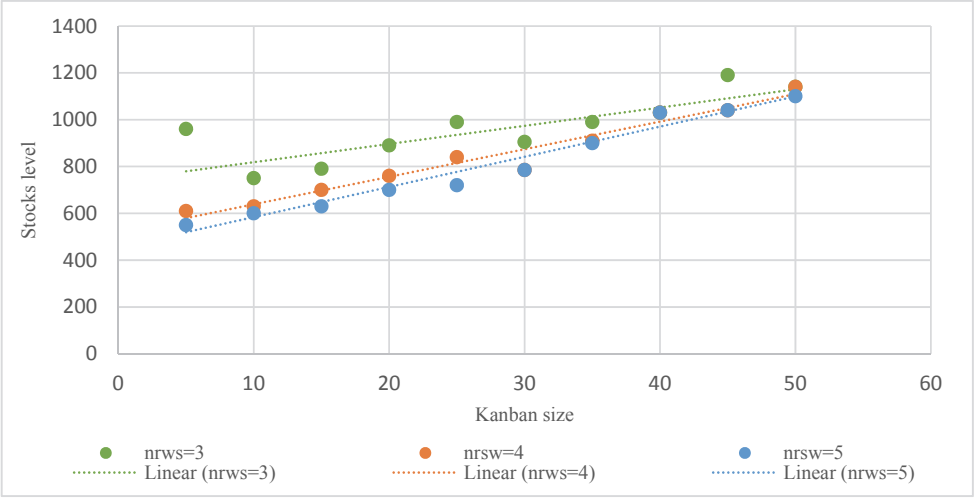

Fig. 4. Stocks level for different kanban sizes and water spiders, when $t_{rflvar} = 4$

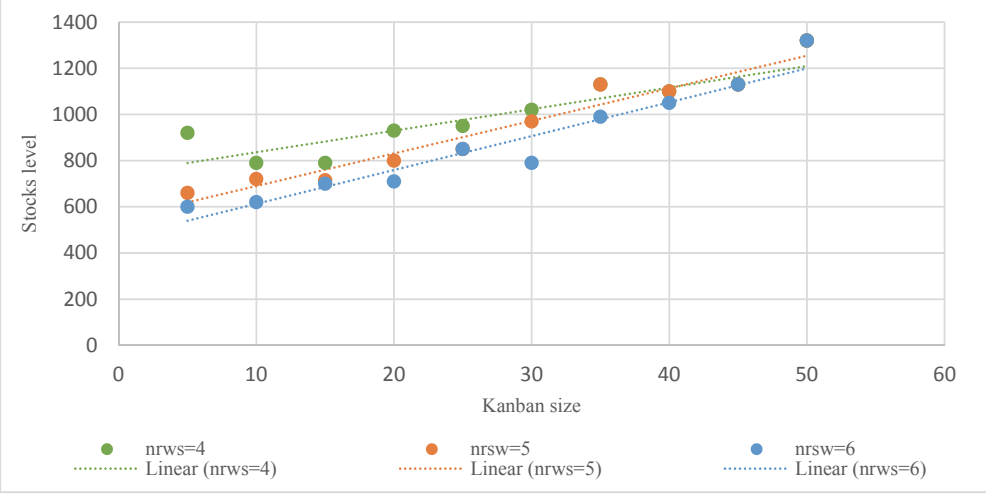

Fig. 5. Stocks level for different kanban sizes and number of water spiders, when $t_{rflvar} = 5$

The theory is confirmed by these results shown in Figs. (3-5). It is noticeable how, the larger the kanban, the lower is the difference between the curves.

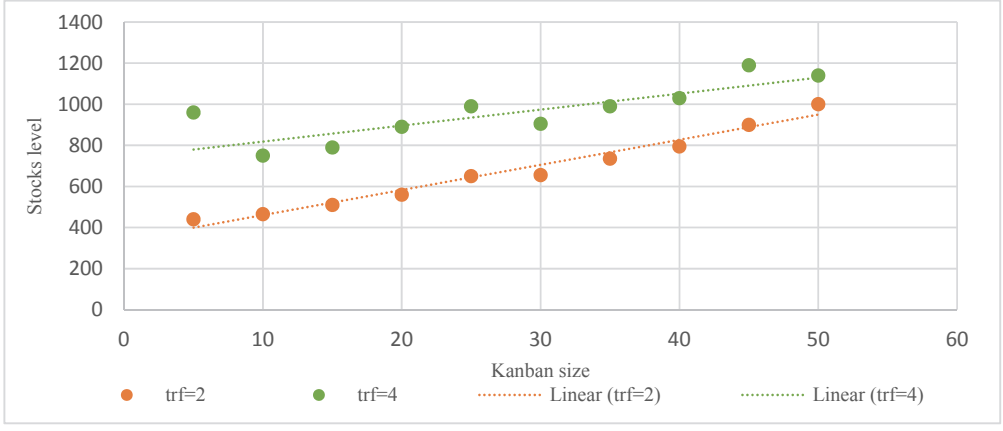

Fig. 6. Stocks level for different kanban sizes and t_{rflvar}, for 3 water spiders

This behaviour can be explained with the fact that large kanban imply less flexibility in the reduction of inventory level, since the removal of even one kanban can have a big impact on the system. The following graphs contain the same results just presented, but organized in order to compare different values of t_{rflvar}.

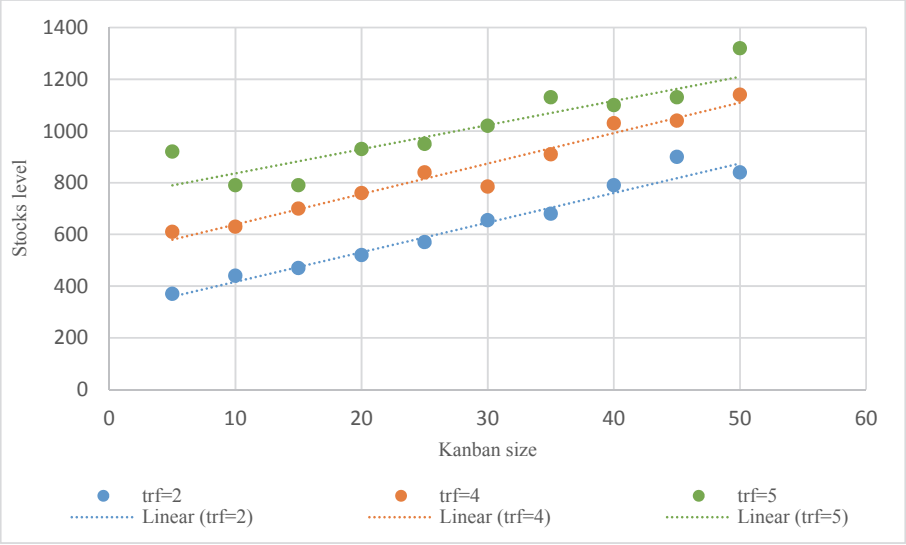

Fig. 7. Stocks level for different kanban sizes and t_{rflvar}, for 4 water spiders

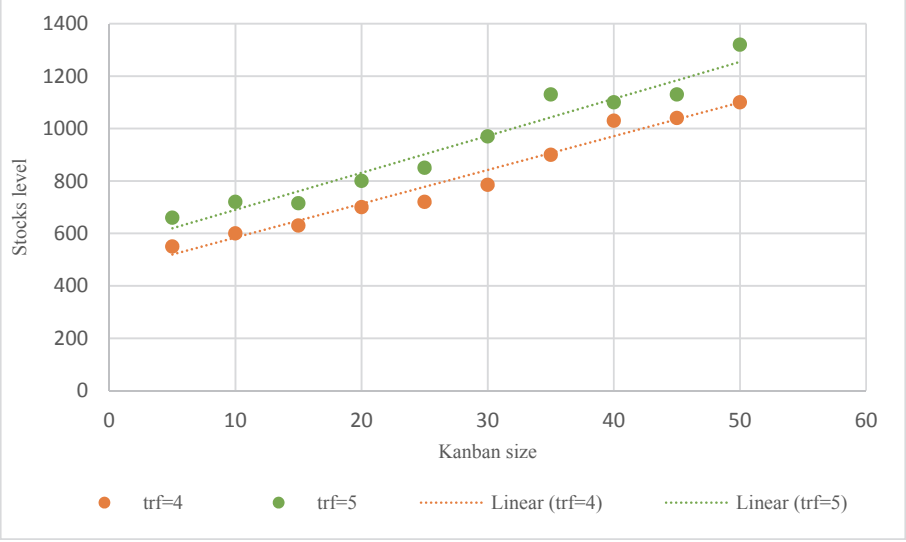

Fig. 8. Stocks level for different kanban sizes and t_{rflvar}, for 5 water spiders

As expected, Figs. (6-8) confirm the fact that higher scenarios with higher refilling time require more stocks to function properly. Moreover, as noticed changing the number of water spiders, the reduction in the level of stocks, when reducing the refilling time, is lower in scenarios that feature larger kanban sizes. It is noticeable the fact that, the lower t_{rflvar}, the higher is the impact of adding or removing a water spider to the system. An explanation could be the fact that the change in water spider has a bigger impact in systems with a lower inventory level which, as shown above, is directly correlated with the entity of the refilling time. The last set of experiments of this section is conducted focusing on the parameter buffer size, which is a novelty item of this work, since no similar article has dealt with it before. The objective here is to understand if and how the size of buffers influences the design of the refilling system. In order to do this, the parameter in question has been varied in conjunction with the proposed variables. As done

before, the best solution in each scenario has been chosen as the one with the lowest inventory level that manages to avoid stockout (and thus has a value of the objective function comparable with the other ones). Fig. 9 shows the results obtained when acting on the number of water spiders.

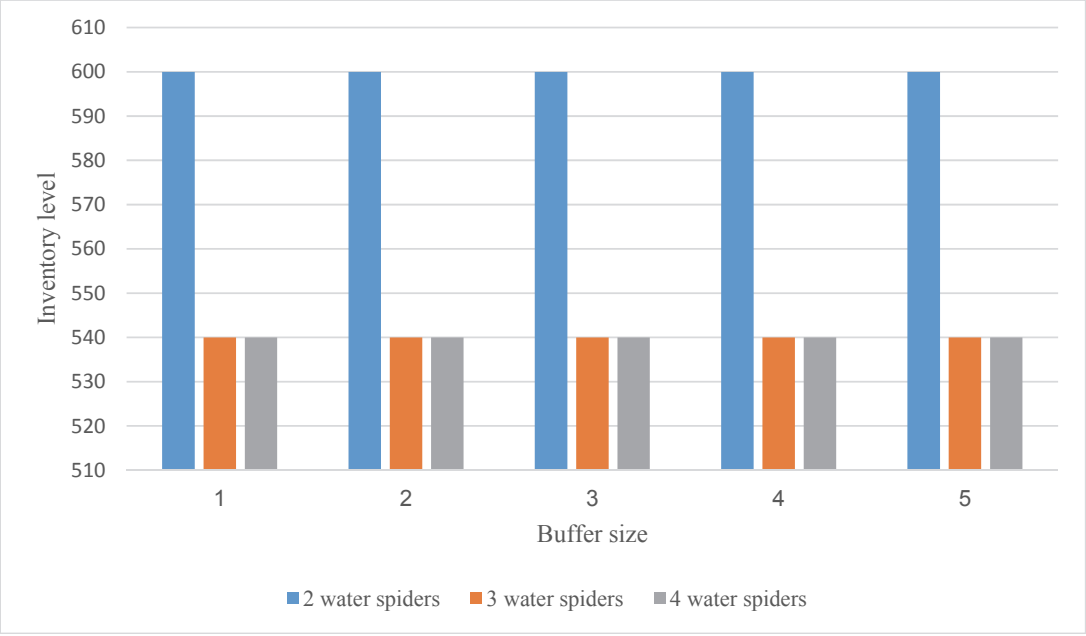

Fig. 9. Inventory level for different buffer sizes and number of water spiders

Besides the reduction in inventory obtained when passing from two to three water spiders, the size of the buffer seems to have no impact at all on the behaviour of the objective function. The same conclusion can be reached when looking at the way inventory level changes, when both buffer size and kanban size are modified. ANOVA is performed on several parameters in order to understand their significance and their impact on the total cost function. The usage of this methodology is quite common in literature: a nice example of its application is found in Battini et al. (2009). This work's ANOVA has been created trying to include as many parameters that were considered by Battini et al. (2009) as possible. Results show that the two with the biggest influence are the capacity of the cart and the length of the supermarket's aisles. The width of the supermarket and its distance from the line are also found to have a moderate impact on the cost function. The ANOVA supports what is already seen in Fig. 9: the size of the buffer has a very small impact on the objective function, and thus on the variables' values.

4.3 Real case application

The methodology described in this research has also been applied to a real life scenario. More specifically, the subject of the project is a part of the assembly line of a manufacturing company. The company is a leading group operating internationally in the oil and gas industry by designing, manufacturing and marketing components, systems and services for the regulation and measurement of natural gas. It aims to create technologically advanced solutions for the extraction, storing, distribution and use of natural gas, directing its research towards the total coverage of both industrial and domestic customers' needs and encouraging a clean and safe type of energy.

The layout of the assembly line is the one used to describe the model in section 3. Table 2 presents the values of all the relevant case parameters.

Table 2
Key parameters of the real case

Parameter	Value
ct + 2	17 sec
l_{aisle}	27 m
$l_{first\text{-}smk}$	9 m
$l_{last\text{-}smk}$	12 m
n_{aisles}	4
n_{cart}	200 pc
t_{linefx}	2 sec
$t_{linevar}$	6 sec
t_{rflfx}	4 sec
t_{rflvar}	4 sec
w_{smk}	16 m
c_s (same for every component)	0.08 Unit/(h*pc)
$c_{salaryws}$	20 Unit/(h*operator)
$c_{unitarybacklog}$	150 Unit/(h*pc)
c_{wip1}	0.38 Unit/(h*pc)
c_{wip2}	0.54 Unit/(h*pc)

Table 3 shows instead the value of decisional variables in the as-is scenario.

Table 3
Value of variables in the as-is solution

	Kanban size	Kanban number
Component 1	50	3
Component 2	50	4
Component 3	50	4
Component 4	50	4
Component 5	50	4
Component 6	50	4
Component 7	50	3
Component 8	50	3
Number of water spiders		3

The algorithm is then configured and run for 150 iterations in order to obtain the optimum value. The simulation is run for a period of 7200 seconds with warmup period of 253 seconds. Each scenario is replicated for 40 times in order to obtain more generalised results. Table 4 shows the new values of decisional variables, as found by the algorithm.

Table 4
Value of variables found by the algorithm

	Kanban size	Kanban number
Component 1	10	5
Component 2	5	12
Component 3	15	4
Component 4	20	4
Component 5	20	4
Component 6	10	7
Component 7	5	7
Component 8	10	4
Number of water spiders		2

Out of the four performance measures used by the algorithm, only the number of water spiders and the stocks level are selected for this comparison. Stock-out has not been used because the company prefers to avoid it, given the large implications on the rest of the system (this is why, in the model, a very high $c_{unitarybacklog}$ is adopted). The cost of work in process instead is not considered simply because the

company did not want to evaluate the possibility of changing the buffers' size. Table 5 presents the comparison of the two aforementioned solutions.

Table 5
Values of performance measures for the as-is solution and the one found with the algorithm

Solution	Inventory level	Number of water spiders
As is scenario	1450	3
Using the proposed method	475	2
Delta	-975	-1

The new solution is thus a significant improvement over the as-is scenario, since it allows the company to maintain the same performance level (most importantly in terms of production rate and stock-out avoidance), while reducing both the level of stocks in the system and the number of operators needed for the refilling procedure.

To further test the methodology presented in this research, two similar sizing methods previously presented in literature have also been applied to the above mentioned real case: Gamberini et al. (2013) and Faccio et al. (2013a). The main driver for the choice of these methods is the simplicity of application to the case.

What follows is a brief recap of the computations made in the two cases in order to find the values of the variables. Please note that, in the reported formulas, the notation of certain items is changed in order to reflect the one used in this work.

- In Gamberini et al. (2013), the emptying time of each SKU is computed, considering the each component's consumption rate and the size of the kanban. About the latter, since this method does not include the computation of the best Kanban size, several values were tested to see which one guaranteed the best performance. The emptying time allows computing the hourly consumption of kanban, which is used to obtain the number of refilling requests for each component. The number of kanban for each component is chosen in order to minimize what Gamberini et al. (2013) call limit in the waiting time (LWT), which has the following formula:

$$LWT = \frac{T_{cov}}{nrkbpn} \times (nrkbpn - 1) - t_{rfl}\ , \quad (2)$$

where T_{cov} is simply the emptying time of an SKU multiplied by nrkbpn. Afterwards, by using the Erlang-C function (which was computed via the open access software found at http://www-ens.iro.umontreal.ca/~chanwyea/erlang/erlangC.html), it is possible to find which is the smallest number of water spiders required in order to respect the minimum LWT.

- In Faccio et al. (2013a), the number of kanban for each component is obtained through the following formula:

$$nrkbpn = \frac{D * t_{rfl} + safetystocks}{szbipn} \quad (3)$$

The size of the safety stocks depends directly on the service level, which is a variable of this model. As in the previous case, the dimension of each kanban is selected by evaluating different possible values, since the article considers this element as a parameter.

A cost function is then built, incorporating the cost of stocks, stock-out and handling. Then, using Microsoft Excel's solver it is possible to find the service level and the number of water spiders that minimize the total cost function. Table 6 summarizes the different solutions obtained.

Table 6
Solutions found using methodologies in existing literature.

	Gamberini et al. (2013)		Faccio et al. (2013a)	
	Kanban size	Kanban number	Kanban size	Kanban number
Component 1	15	4	5	16
Component 2	15	6	5	20
Component 3	15	4	5	16
Component 4	15	6	5	20
Component 5	15	6	5	20
Component 6	15	6	5	20
Component 7	15	3	5	10
Component 8	15	3	5	10
Number of water spiders	4		2	

It is then possible to compare each of these solutions to the one found by the algorithm proposed in this paper.

Table 7
Difference in performance measures between Gamberini et al. (2013) methodology and the proposed algorithm

Solution	Inventory level	Number of water spiders
Gamberini et al. (2013)	570	4
This work	475	2
Delta	-95	-2

Gamberini et al. (2013) method gives a slightly higher level of inventory, but a much higher number of water spider. The main reason for this is probably the usage of the Erlang-C function to determine the value of this variable, which leads to an excessive approximation and thus a sub-optimal solution.

Table 8
Difference in performance measures between Faccio et al. (2013a) methodology and the proposed algorithm

Solution	Inventory level	Number of water spiders
Faccio et al. (2013a)	660	2
This work	475	2
Delta	-185	0

Faccio et al. (2013a) gives a solution with the same number of water spiders, but about 40% higher level of Inventory. This can be due, among other things, to the fact that uses the same kanban size for all the components.

It is possible to notice how all three methods tested are improving over the as-is situation. Looking at tables 7 and 8 it is possible to conclude that the solution found through the algorithm presented in this work represents the best one. Considering the level of stocks, a key advantage of the method proposed in this work is the ability to act on both of the factors that influence it, number of kanban and kanban size. This means that the algorithm has more room to work with and thus is able to find, for each scenario, a more tailor-made solution.

4.4 Other possible applications

Throughout this work, there have been several chances to underline the flexibility of the proposed model's design. This feature is thought in order to allow the application to a variety of scenarios. For instance, one of the possible extensions of its field of use is the so-called industry 4.0. The term industry 4.0 groups all the innovations that have or will take place because of the advanced digitalization of factories (Lasi et al., 2014).

Regarding the field treated in this work, advances in technology have enabled the substitution of the operator-cart combination with Automated Guided Vehicles (AGVs) (Cao & Lin, 2014; Kesen & Baykoc, 2007), in order to increase the efficiency of handling operations (Chen et al., 2009). They are used in conjunction with sensors that constantly monitor the level of stocks, which helps maintaining a low inventory level (Sanders et al., 2016). Industrial wireless networks are often in place to guarantee information exchange between these vehicles and the rest of the factory (Yue et al., 2015).

The study of an AGV system can be quite complex, considering also the fact that they are governed by dispatching rules: simulation can be very useful in these cases (Kesen & Baykoc, 2007). Moreover, the material logistics apparatus can be seen as a sophisticated discrete event system (Chen et al., 2009).

Another aspect worth considering is the fact that automated refilling systems have fewer possible sources of variation; this means that a simulation model will require a smaller amount of assumptions to be able to faithfully reproduce its real counterpart.

These statements back the usage of the model presented in this work for the purposes of representing an AGV feeding system. Simpler cases may be even modelled right off the bat, just considering the water spiders as AGVs and, of course, changing accordingly the cost values. On the other hand, complex scenarios, such as the ones featuring a centralized dispatching system, may require a couple of intuitive modifications to the model.

A particularly fitting aspect of this work's model is the penalty for small bins (assumption 10). In this case, small bins would imply an even bigger cycle time given the fact that, unlike humans, many AGVs are not able to retrieve or release more than one bin at a time. A partial exception to this statement is the mechanism found in Emde et al. (2012): thanks to the peculiar way of releasing bins through elastic springs, multiple bins can be unloaded simultaneously. At the same time though, the loading time is increased, since the operator must also pay attention to the order bins are put on the cart. In any case, the proposed model is able to adapt to every type of vehicle, given the usage of two separate parameters for the time needed to handle one bin in the line and in the supermarket.

6. Conclusions and further research

This work proposes a new method to size a kanban-based feeding system for a mixed-model assembly line. The most significant contribution of this work to the research field is certainly the joint analysis of the number and dimension of kanban for each component and the study of the corresponding trade-off. Another addition is the evaluation of the impact of the size of inter-station buffers on the design of the refilling system.

The methodology selected for this work is a so-called hybrid approach, which combines a model, used to simulate the real life scenario, and a meta-heuristic algorithm to explore the solutions space and find the best values for each decisional variable. A discrete-event simulation is used: the software used to create and run the model is Mathworks Simulink. The algorithm has been written in MATLAB code, in order to simplify the integration with the model. Specifically, a Particle Swarm Optimization algorithm is chosen, mainly because of better reported performances in comparison with other meta-heuristic techniques. After some testing, the size of the swarm is set at 40 particles and the number of iterations at 150. During each iteration, particles explore the space, taking into account also the position of each of their neighbours. Their objective is to minimize the total cost function, which includes the costs of inventory, stock-out, handling and work in process. The simulation is run for a period of 7200 seconds with warmup period of 253 seconds. Each scenario is replicated for 40 times in order to obtain more generalised results. In order to evaluate the significance of each of the decisional variables chosen, different experiments are performed. Each parameter is set to vary within a specific range, obtained by researching the values found in literature. The results show how the trade-off between number of kanban

and kanban size behaves in different scenarios. Moreover, a similar analysis is conducted for the buffer size parameter, given the fact that this work is the only research of its kind to mention it. The behaviour of the system is investigated while varying the parameter in question and, in turn, the decisional variables. Results show that buffer size has only very little influence on the design of the system.

ANOVA is used to understand the parameters with the highest impact on the objective function. Several factors are considered: results show that the two with the biggest influence are the capacity of the cart and the length of the supermarket's aisles. A correspondence has also been shown with the conclusions obtained by Battini et al. (2009). The width of the supermarket and its distance from the line are also found to have a moderate impact on the cost function. Moreover, the result for buffer size is found to be in line with the one of the sensitivity analysis.

Furthermore, the methodology explained in this research has been tested by applying it to a real life scenario. In order to widen the scope of experiment, the same problem is also solved through the methods proposed in two similar works. Two performance measures are used to compare solutions, total number of stocks in the system and number of water spiders, since they are two of the most used metrics in literature. First of all, all solutions proved to be better than the as-is situation. The methodology proposed in this work is then shown to be the best one, outmatching both of its competitors in all the performance measures.

Finally, a brief study is conducted in order to understand if the model could fit a scenario based on the industry 4.0 paradigm. Looking at a selection of articles on the refilling subject, the conclusion is that there isn't much difference between the solutions utilized in those researches and the one proposed in this work.

As a possible future expansion of this research, the introduction of an adaptive type of kanban is suggested, in order to first evaluate all of its eventual benefits, but also if and how relations between variables change. The model, and in particular the demand subsystem, has already been designed with this extension in mind: the demand proportion, meaning the ratio between the demand of different product variants, can be set to change at any point during the simulation, even more than once. Another potentially interesting issue is the water spider policy; in detail, investigating if performances of the system change in relation to this choice.

References

Aghajani, M., Keramati, A., Moghadam, R.T., & Mirjavadi, S.S. (2016). A mathematical programming model for cellular manufacturing system controlled by kanban with rework consideration.*The International Journal of Advanced Manufacturing Technology*, *83*(5-8), 1377-1394.

Azadeh, A., Ebrahimipour, V., & Bavar, P. (2010). A hybrid ga-simulation approach to improve jit systems. *International Journal of Production Research*, *48*(8), 2323-2344.

Ballou, R.H. (2004). *Business logistics/supply chain management: planning, organizing, and controlling the supply chain.* Pearson Education International.

Battini, D. Faccio, M., Persona, A., & Sgarbossa, F. (2009). Design of the optimal feeding policy in an assembly system. *International Journal of Production Economics*, *121*(1), 233-254.

Battini, D., Faccio, M., Persona, A., & Sgarbossa, F.(2010). “supermarket warehouses”: stocking policies optimization in an assembly-to-order environment. *The International Journal of Advanced Manufacturing Technology*, *50*(5-8), 775-788.

Battini, D., Boysen, N., & Emde, S.(2013). Just-in-time supermarkets forpart supply in the automobile industry. *Journal of Management Control*, *24*(2), 209-217.

Battini, D., Gamberi, M., Persona, A., & Sgarbossa, F. (2015) Part-feeding with supermarket in assembly systems: transportation mode selectionmodel and multi-scenario analysis. *Assembly Automation*, *35*(1), 149-159.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, *35*(3), 268-308.

Bowden, R.O., Hall, J.D., & Usher, J.M. (1996). Integration of evolutionary programming and simulation to optimize a pull production system. *Computers & Industrial Engineering*, *31*(1), 217-220.

Cao, Z., & Li, H. (2013). *Multi-objective optimization of material collaborative delivery for mixed model automotive assembly process*. In Business Intelligence and Financial Engineering (BIFE), 2013 Sixth International Conference on, pages 405-409. IEEE.

Cao, Z., & Lin, Z. (2014). *Multi-objective optimization of material delivery for mixed model automotive assembly line based on particle swarm algorithm.* In Control Conference (CCC), 2014 33rd Chinese, pages 2979-2984. IEEE.

Caputo, A.C., & Pelagagge, P.M. (2011). A methodology for selecting assembly systems feeding policy. *Industrial Management & Data Systems*, *111*(1), 84-112.

Chen, H., Wang, A., & Ning, R. (2009). *Material logistic process control in hierarchical workshop model*. In Technology and Innovation Conference 2009 (ITIC 2009), International, pages 1-6. IET.

Choi, W., & Lee, Y. (2002). A dynamic part-feeding system for an automotive assembly line. *Computers & Industrial Engineering*, *43*(1), 123-134.

Clerc, M., & Kennedy, J. (2002). *The particle swarm-explosion, stability, and convergence in a multidimensional complex space*. *IEEE transactions on Evolutionary Computation, 6*(1), 58-73.

Eberhart, R., & Kennedy, J. (1995). *A new optimizer using particle swarm theory*. *In Micro Machine and Human Science, 1995*. MHS'95., Proceedings of the Sixth International Symposium on, pages 39-43. IEEE.

Emde, S., & Boysen, N. (2012a). Optimally locating in-house logistics areas to facilitate jit-supply of mixed-model assembly lines. *International Journal of Production Economics*, *135*(1), 393-402.

Emde, S., & Boysen, N. (2012b). Optimally routing and scheduling tow trains for jit-supply of mixed-model assembly lines. *European Journal of Operational Research*, *217*(2), 287-299.

Emde, S., Fliedner, M., & Boysen, N. (2012). Optimally loading towtrains for just-in-time supply of mixed-model assembly lines. *IIE Transactions*, *44*(2), 121-135.

Emde, S., & Gendreau, M. (2017). Scheduling in-house transport vehicles to feed parts to automotive assembly lines. *European Journal of Operational Research*, *260*(1), 255-267.

Emde, S., & Schneider, M. (2018). Just-In-Time Vehicle Routing for In-House Part Feeding to Assembly Lines. *Transportation Science*, *52*(3), 657-672.

Faccio, M., Gamberi, M., &Persona, A. (2013a). Kanban number optimisation in a supermarket warehouse feeding a mixed-model assembly system. *International Journal of Production Research*, *51*(10), 2997-3017.

Faccio, M., Gamberi, M., Persona, A., Regattieri, A., & Sgarbossa, F. (2013b). Design and simulation of assembly line feeding systems in the automotive sector using supermarket, kanbans and tow trains: a general framework. *Journal of Management Control*, *24*(2), 187-208.

Faccio, M. (2014). The impact of production mix variations and models varieties on the parts-feeding policy selection in a jit assembly system. *The International Journal of Advanced Manufacturing Technology*, *72*(1-4), 543-560.

Faccio, M., Cohen, Y., Bortolini, M., Ferrari, E., Gamberi, M.,Manzini, R., & Regattieri, A. (2015). New kanban model for towtrain feeding system design. *Assembly Automation*, *35*(1), 128-136.

Gamberini, R., Meli, M., Galloni, L., Rimini, B., & Lolli, F. (2013). Alternative refilling policies for an assembly line managed by kanbans. *IFAC Proceedings Volumes*, *46*(9), 1914-1919.

Golz, J., Gujjula, R., Gunther, H.O., Rinderer, S., & Ziegler, M. (2012). Part feeding at high-variant mixed-model assembly lines. *Flexible Services and Manufacturing Journal*, *24*(2),119-141.

Hanson, R., & Finnsgard, C. (2014). Impact of unit load size on in-plant materials supply efficiency. *International Journal of Production Economics*, *147*, 46-52.

Hao, Q., & Shen, W. (2008). Implementing a hybrid simulation model for a kanban-based material handling system. *Robotics and Computer-Integrated Manufacturing*, *24*(5), 635-646.

Heng, Z., Aipinga, L. I., Xuemeia, L. I. U., Liyuna, X. U., & Moronia, G. (2017). Modeling and Performance Evaluation of Multistage Serial Manufacturing Systems with Rework Loops and Product Polymorphism. *Procedia CIRP*, *63*, 471 – 476.

Hobbs, D.P. (2003). Lean manufacturing implementation: a complete execution manual for any size manufacturer. *J. Ross Publishing, Inc.*, ISBN 1-932159-14-2.

Hou, T.H.T., & Hu, W.C.(2011). An integrated moga approach to determine the pareto-optimal kanban number and size for a jit system. *Expert Systems with Applications*, *38*(5), 5912-5918.

Kennedy, J. (2011). *Particle swarm optimization*. In Encyclopedia of machine learning, pages 760-766. Springer.

Kesen, S.E., & Baykoc, O.F. (2007). Simulation of automated guided vehicle (agv) systems based on just-in-time (jit) philosophy in a job-shop environment. *Simulation Modelling Practice and Theory*, *15*(3), 272-284.

Koster, R.D., Le-Duc, T., & Roodbergen, K.J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, *182*(2), 481-501.

Kumar, C.S., & Panneerselvam, R.(2007). Literature review of jit-kanban system. *The International Journal of Advanced Manufacturing Technology*, *32*(3-4), 393-408.

Lasi, H., Fettke, P., Kemper, H.G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, *6*(4), 239-242.

Limere, V., Landeghem, H.V., Goetschalckx, M., Aghezzaf, E.H., & McGinnis, L.F. (2012). Optimising part feeding in the automotive assembly industry: deciding between kitting and line stocking. *International Journal of Production Research*, *50*(15), 4046-4060.

Lolli, F., Gamberini, R., Giberti, C., Rimini, B., & Bondi, F. (2016). A simulative approach for evaluating alternative feeding scenarios in a kanban system. *International Journal of Production Research*, *54*(14), 4228-4239.

Monden, Y. (2011). *Toyota production system: an integrated approach to just-in-time*. CRC Press.

Ohno, T.(1988). *Toyota production system: beyond large-scale production*. CRC Press.

Portioli-Staudacher, A., & Tantardini, M. (2012). Lean implementation in non-repetitive companies: a survey and analysis. *International Journal of Services and Operations Management*, *11*(4), 385-406.

Pradeepmon, T.G., Sridharan, R., & Panicker, V.V. (2018). Development of modified discrete particle swarm optimization algorithm for quadratic assignment problems. *International Journal of Industrial Engineering Computations*, *9*(4), 491–508.

Roukya, N., Abourrajaa, M.N., Boukachoura, J., Boudebousa, D., Alaouib, A.E.H., & Khoukhic, F.E. (2019). Simulation optimization based ant colony algorithm for the uncertain quay crane scheduling problem. *International Journal of Industrial Engineering Computations*, *10*(1), 111–132.

Sali, M., Sahin, E., & Patchong, A. (2015). An empirical assessment of the performances of three line feeding modes used in the automotive sector: line stocking vs. kitting vs. sequencing. *International Journal of Production Research*, *53*(5), 1439-1459.

Sali, M., & Sahin, E. (2016). Line feeding optimization for Just in Time assembly lines: An application to the automotive industry. *International Journal of Production Economics*, *174*, 54-67.

Sanders, A., Elangeswaran, C., & Wulfsberg, J. (2016). Industry 4.0implies lean manufacturing: Research activities in industry 4.0 function as enablers for lean manufacturing. *Journal of Industrial Engineering and Management*, *9*(3), 811-833.

Satoglu, S.I., & Sahin, I.E. (2013). Design of a just-in-time periodic material supply system for the assembly lines and an application in electronics industry. *The International Journal of Advanced Manufacturing Technology*, *65*(1-4), 319-332.

Savino, M.M., & Mazza, A. (2015). Kanban-driven parts feeding within a semi-automated o-shaped assembly line: a case study in the automotive industry. *Assembly Automation*, *35*(1), 3-15.

Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and kanban system materialization of just-in-time and respect-for-human system. *International Journal of Production Research*, *15*(6), 553-564.

Toffoli, T., & Margolus, N. (1987). *Cellular automata machines: a new environment for modeling*. MIT press.

Toncovicha, A.A., Rossita, D.A., Frutosa, M., & Rossita, D.G. (2019). Solving a multi-objective manufacturing cell scheduling problem with the consideration of warehouses using a simulated annealing based procedure. *International Journal of Industrial Engineering Computations*, *10*(1), 1–16.

Vatalaro, J., &Taylor, R. (2005). *Implementing a mixed model Kanban System: The lean Replenishment Technique for pull production*. Volume 1, CRC Press.

Widyadana, G.A., Wee, H.M., &Chang, J.Y. (2010). Determining the optimal number of kanban in multi-products supply chain system. *International Journal of Systems Science*, *41*(2), 189-201.

Yue, X., Cai, H., Yan, H., Zou, C., & Zhou, K. (2015). Cloud assisted industrial cyber-physical systems: an insight. *Microprocessors and Microsystems*, *39*(8), 1262-1270.