Int J Adv Manuf Technol (2014) 70:385–395
DOI 10.1007/s00170-013-5282-5

ORIGINAL ARTICLE

Optimization of POLCA-controlled production systems with a simulation-driven genetic algorithm

M. Braglia · D. Castellano · M. Frosolini

Received: 3 April 2013 / Accepted: 29 August 2013 / Published online: 14 September 2013

Abstract The present work is meant to show the effective capability of optimizing an unbalanced Paired-Cell Overlapping Loops of Cards with Authorization (POLCA)-controlled production system by means of a heuristic algorithm. This objective is suggested by the fact that one of the most significant issues when using card-driven production control systems is represented by the optimized setting of the large number of cards within the control loops. This is particularly true in the case of unbalanced systems, where the number of cards may vary significantly among the different loops. Little law is usually adopted in literature to infer this number from historical data, but the obtained number is usually far from the optimum. Indeed, in real-world applications, the systems to be controlled are designed to process units with very different routings, each with different probability to occur. In all these situations, they result particularly difficult to set correctly. To this aim, in the present work a Genetic Algorithm is used. The objective is that of finding the correct number of cards and to reduce the overall Total Throughput Time and the average Work In Process. The proposed approach may provide a valid support tool to overcome these limitations, making the most of POLCA capabilities in many manufacturing configurations.

Keywords POLCA · m-CONWIP · Unbalanced production systems · Work in process · Heuristic · Genetic algorithms · Simulation

M. Braglia · D. Castellano · M. Frosolini (✉)
Dipartimento di Ingegneria Civile e Industriale, Università di Pisa,
Via Bonanno Pisano, 25/B, 56126 Pisa, Italy
e-mail: m.frosolini@ing.unipi.it

M. Braglia
e-mail: m.braglia@ing.unipi.it

D. Castellano
e-mail: davide.castellano@for.unipi.it

1 Introduction

Paired-Cell Overlapping Loops of Cards with Authorization (POLCA, [1]) has been proposed as an effective method to improve production systems throughput time performance. This result is obtained by means of an opportune control on the release and the dispatching of work orders to the shop floor, being this the key point where traditional Kanban system fails. Briefly, POLCA is a hybrid card-based pull production control mechanism characterized by multiple short control loops, each cycling over two successive workstations and overlapping two by two. This distinctive feature makes it able both to control the overall workload and to balance it among the various workstations. Researchers have shown [2] that in the case of balanced systems, POLCA is very effective in reducing the Total Throughput Time (TTT) that includes the time effectively spent by work packages within the floor shop and the time spent by orders within the orders queue. Here, the word "balanced" indicates that routings are uniformly distributed and that both working and inter-arrival times are, on average, similar. In particular, the model analyzed by the authors allows a significant reduction of the TTT with respect to the corresponding uncontrolled production system and to the standard CONWIP configuration. Actually, outcomes show that POLCA is an important and effective production control method. However, some considerations must be addressed carefully due to the fact that the investigated production system is very peculiar. Indeed, it is balanced by definition, due to the strong design hypotheses made by the authors. Often, in real-world applications, the systems to be controlled are designed to process units with very different routings, each with significantly different probability to occur. In all these situations, queues may form in front of certain workstations, whereas some other machines may be temporarily starving and waiting for units to be worked. In particular, when these systems have to be optimized one of the major drawbacks to be faced is represented by the setting of

the number of cards that are required within each different loop. Little law is usually adopted to infer this number from historical data [1]. However, due both to the variability of the quantities involved in the calculation and to the intrinsic complexity of the systems to be controlled, the obtained number is usually far from the optimum. Some authors [2] have proposed modifications to the law to include queuing times and uncertainty within the computation, but the result is rather complicated and difficult to be adopted in real situations. And in fact, in the case of balanced systems a simple "trial and error" procedure is adopted [2]. When routings and arrival times are different this method becomes useless at all. Indeed, in this case each loop needs a potentially different number of cards and, when the number of loops grows up, the trial and error procedure would be time-consuming and ineffective. The quest for a better method is definitely worth the effort. In literature, a significant number of works deal with the optimization of traditional Kanban systems by means of simulation and/or heuristic methods. Among them, it is worth mentioning the study of Koulouriotis et al. [3]), where the authors perform the optimization of different pull control policies (Kanban, CONWIP and hybrid pull systems) by means of simulation and a genetic algorithm (GA) with resampling. The obtained results confirm that the distribution of cards has a considerable effect on the performance of the controlled systems [4] and that genetic algorithms are very effective in such noisy environments. Selvaraj [5] proposes the adoption of a genetic algorithm to determine the correct number of Kanbans within a Generalized Kanban Control System with varying demand arrival rates. The objective is that of maximizing the throughput and the machine utilization level while reducing the average work in process. Results are validated by means of simulation tests. Alabas et al. [6] compare three heuristic procedures (tabu search, genetic algorithms, and simulated annealing) supported by a simulation tool to verify their speed of convergence and their effectiveness in finding the correct number of Kanbans. Köchel and Nieländer [7] combine a GA with simulation to find the optimal order policies in complex Kanban controlled manufacturing system. Similar applications of such methods can be found also in Hu and Yi [8]; Al-Tahat et al. [9] and Huang et al. [10]). With respect to m-CONWIP systems, many works have been published that deal with mathematical models to optimize the number of cards in complex systems [11–13]. On the contrary, the recent work of Ajorlou et al. [14] proposes the adoption of a Bee Colony approach to optimize the WIP and the total makespan of a multi-CONWIP-based manufacturing system. They show its effectiveness and affirm that the approach may be of great help in real applications. It clearly emerges that heuristic methods and simulation provide a useful tool to evaluate the impact of design alternatives and can certainly be adopted as a decision support instrument, able to assist managers in the control of complex manufacturing systems.

The above mentioned results and the fact that, at present, no such attempt has been carried out yet, authorize to believe that the problem of optimizing the number of cards in an unbalanced POLCA-controlled production system can be effectively approached by means of simulation-driven heuristics. The present work is therefore aimed at showing the effective capability of a simulation-driven Genetic Algorithm to individuate the correct number of cards and to reduce the overall TTT and the average WIP both with respect to the unconstrained model and to a multi-CONWIP-controlled model. Though the algorithm in itself does not represent the objective of the work, the GA has been selected due to the fact that it is well known, widely used, and that it has proved to be effective in solving a large number of engineering problems.

The paper is structured as follows: first of all, a brief introduction to the studied POLCA system is provided, followed by a short yet exhaustive description of the adopted genetic algorithm. Finally, the case study will be presented in detail to show the effective capabilities of the proposed approach.

2 The proposed unbalanced POLCA system

POLCA [1, 15] is a pull system with a peculiar authorization mechanism. The triggering method can be implemented either physically, by means of real cards cycling between two work centers, or electronically. An order can enter the production stage if and only if a card is available for the loop the item is trying to enter. Obviously, the number of available cards represents the constraint. The major characteristic of POLCA with respect to Kanban or CONWIP resides in the fact that loops are overlapping with one another (Fig. 1).

This assures that a workstation only processes items for which capacity is available in the immediately succeeding loop. In Fig. 1, when a card for loop A/B is available an item is allowed to enter the shop floor (being worked on machine A). Then, it waits before workstation B until a card for the subsequent loop B/C is available. The card A/B still remains attached to the item, and it is released only when the item is allowed to enter the next loop. Owing to this mechanism, the POLCA is able to effectively increase the performance of the production system. As already stated in the previous paragraph, recent researches have shown that in the case of balanced systems POLCA is very effective in reducing the TTT. It is noteworthy that "balanced" refers to the fact that routings are uniformly distributed and that both working and inter-arrival times are, on average, similar. Owing to this, the setting of the number of cards within bins is rather easy (see again [2]). Unfortunately, in real-world applications, the systems to be controlled are designed to process units with very different routings, each with significantly different probability to occur. In all these situations, queues may form in front of certain

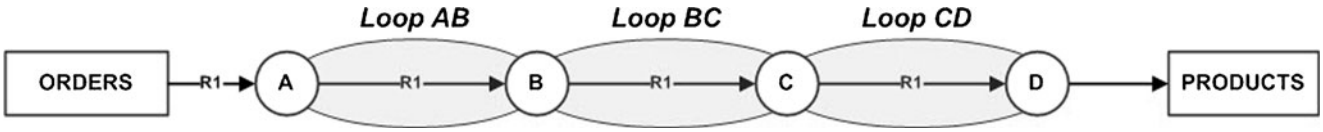

Fig. 1 Overlapping loops in a POLCA-controlled system

workstations, whereas some other machines may be temporarily starving and waiting for units to be worked. This makes the optimization of the number of cards particularly burdensome. This work tries to propose a solution to this issue.

The proposed model is made of four workstations (A, B, C, D) that serve three different products (P1, P2, P3). Each product is characterized by a specific routing (R1, R2, R3), and the corresponding arrival rate is kept very different with the aim of further unbalancing the whole system. A simple scheme of the model is reported in Fig. 2, where workstations, routings, and the corresponding POLCA loops are clearly evidenced. Inter-arrival times are such that all the workstations have enough capability to perform the required work yet maintaining the bottle-neck fully occupied.

In detail:

- Inter-arrival time is exponentially distributed with an average of 1.6 min;
- Product P1 follows the routing R1=[A, B, C, D] and represents the 60 % of the production mix;
- Product P2 follows the routing R2=[A, B, D] and represents the 30 % of the production mix;
- Product P3 follows the routing R3=[A, C, D] and represents the 10 % of the production mix;
- Workstation A has a processing time of 1 min;
- Workstation B has a processing time of 1.75 min;
- Workstation C has a processing time of 1 min;
- Workstation D has a processing time of 1.5 min;
- All the processing times follow the Erlang distribution with $k=2$;
- Orders are processed on "longest queue" basis at each workstation.

The overall performance of the system is measured considering two distinct quantities, namely the Shop floor Throughput Time (STT) and the Order Pool Time (OPT). The former identifies the time spent by units within the shop floor, whereas the latter is the time spent by orders within the pool prior to enter the production stage (waiting for available cards). Their sum is the TTT, which therefore represents the actual performance metric:

$$TTT = OPT + STT$$

Adopting the POLCA control system allows a significant reduction of the STT when constraints to the workload are introduced. Such restriction, in turn, increases the OPT, due to the fact that some orders have to wait for a free card to be allowed to enter the production system. However, the average reduction of the STT tends to balance the increase in the OPT and, in particular in the case of constant processing times, it allows a global reduction of the TTT (Fig. 3). When random processing times are used, the effect is generally small, but the corresponding reduction of the STT makes the control system worth the effort, due to the fact that orders could be effectively scheduled with the aim of reducing the OPT while maintaining very small values of the shop-floor throughput time. In general, while the TTT is kept unchanged or is diminished, the STT and the WIP may be effectively reduced. When constraints become even more restrictive (i.e., when the number of available cards is further reduced) the OPT begins to raise rapidly and soon prevails over the decrease in the STT, making the average TTT increasing again.

The "trial and error" procedure (Gems and Riezebos [2]) that could be used to determine the optimal number of cards can be resumed as follows:

1. The initial number of cards for each loop is kept relatively high to simulate the unconstrained systems, for which OPT=0 and STT=TTT;
2. The critical points are found by progressively reducing the number of cards in all control loops;

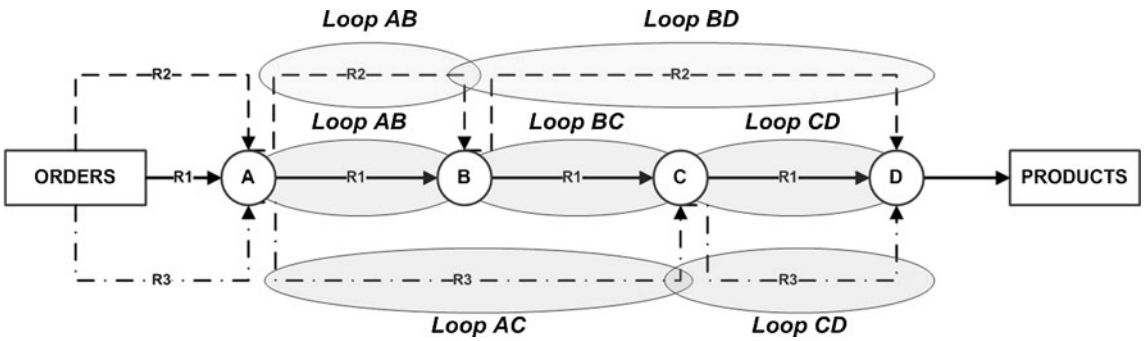

Fig. 2 The unbalanced model with routings and loops

3. When the TTT begins to increase again, due to excessive values of the OPT (meaning that some units were waiting for cards to enter the shop floor), the iteration is stopped.

Unfortunately, the relatively high number of loops in the model, the fact that the distribution of cards has a considerable effect on the performance of the controlled systems and, finally, the consideration that the number of cards in a loop modifies the optimal number in the remaining loops make the whole procedure time-consuming and often almost ineffective. Tests have shown, for instance, that many configurations exist that minimize the TTT while are not able to optimize the average WIP value. Such considerations justify the adoption of a completely different approach. The same problem affects as well the multi-CONWIP systems but, due to the lower number of routings with respect to the actual number of loops in a POLCA-controlled system, the latter is definitely more cumbersome to optimize.

3 The genetic algorithm

To optimize the number of cards within the proposed model, a standard Genetic Algorithm (GA) [16] has been opportunely configured. Though the algorithm in itself does not represent the objective of the work—this is, actually, the capability of optimizing the number of cards within the loops by means of a heuristic search—the GA has been selected due to the fact that it is well known, widely used, and effective. Indeed, this heuristic approach has been extensively and successfully applied in the last decades to solve production and operations management problems (see, for instance, [17–19]). In a GA, a population of aspirant solutions to a generic optimization problem evolves towards better configurations. The evolution starts from a randomly generated population and moves by generations, during which the fitness of every individual is evaluated. The more suitable individuals are stochastically selected, two by two, and are later recombined and mutated to form a new population. The new population is then used in the next iteration of the algorithm. The algorithm terminates when either a maximum number of generations has been reached or a suitable fitness level has been obtained for the population (convergence). The typical genetic algorithm requires therefore:

- A genetic representation of the solution space;
- A fitness function to evaluate the solutions;
- Crossover and mutation operators;
- A valid termination condition;
- A final neighborhood search procedure to further improve the obtained solution.

The latter point is very important as a GA generally moves toward the optimal solution but is often unable to reach the actual *minimum*. The neighborhood procedure is therefore necessary to explore with a finer mesh the solution space and to individuate eventual better individuals.

As already stated, the representation of each individual is generally constituted by a fixed-length array of bits. Variable length representations may also be used, but the crossover implementation becomes consequently burdensome. Once the genetic representation and the fitness function are defined, the GA proceeds to initialize a population of solutions and then to improve it through the iterative application of the available operators.

To begin with, in the present circumstance, chromosomes embody the numbers of cards in each loop by means of a simple binary representation. To grant that the GA is able to explore the whole solution space, including the unconstrained configurations, after some initial testing the number of bits for each loop within the model has been fixed to 6, thus giving a maximum of 63 cards. Being 7 the loops, each chromosome has an overall length of 42 bits. Figure 4 clearly shows the sequence of loops and the corresponding routings within a chromosome.

In brief, the first six bits are used to calculate the cards in the first loop within the first routing, the following six refer to the second loop in the same routing and so on till all the loops of the routing have been correctly allocated. Then, loops from the other routings follow up in a similar fashion till the completion of the chromosome.

The GA uses two different but equally probable crossover operators:

- Single-point crossover. A position is randomly selected within each two chromosome (ancestors) and the offspring is generated by combining the leftmost part of the first with the rightmost part of the second and vice versa;
- Two-point crossover. In this case, two points are selected and the genes comprised between these positions are moved from the first chromosome to the second and vice versa.

Also, it makes use of two equally probable mutation operators:

- Position exchanging. Two positions are randomly selected within a single chromosome and the corresponding bits are exchanged;
- Bit inversion. In this case, a position is randomly selected and the corresponding bit is modified.

The crossover procedure is controlled by the crossover ratio (C_R) and the mutation by the mutation ratio (M_R). At the end of each iteration, the best individual within the population undergoes an opportune neighborhood search. In brief, the number of cards is obtained from the binary representation for each loop and is later modified (increased or reduced of one unit). The

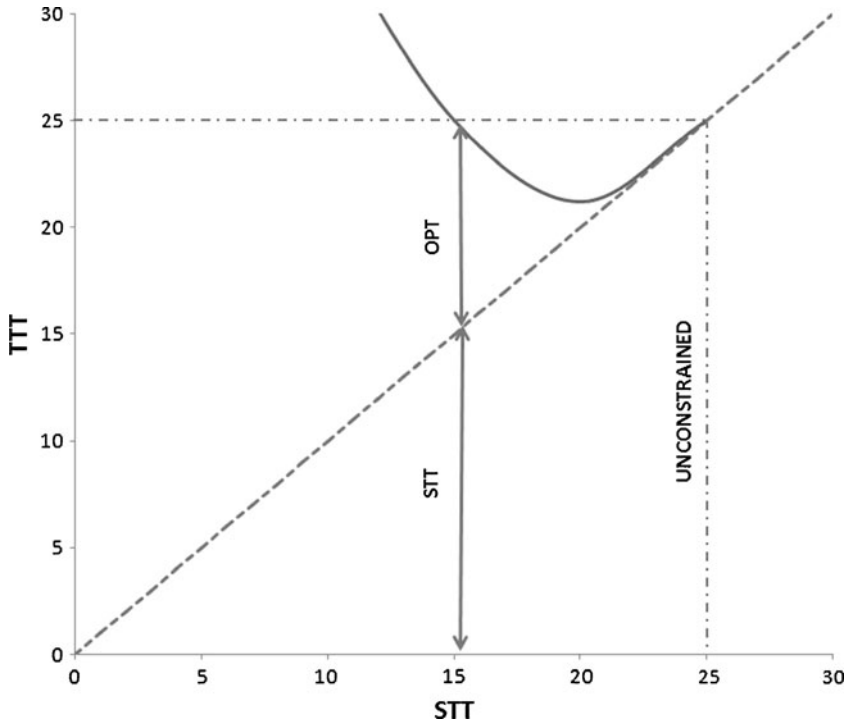

Fig. 3 TTT reduction trend when constraints are applied

corresponding fitness is evaluated and compared to the original one. If the new solution is better than the former, it is kept in its place and the GA begins a new iteration. The best individual is also saved in a separate structure, the *elite*, and is re-inserted within the population at each iteration to make the most of its performing genotype. When all the epochs have been run or if no improvement to the best individual has been found during the last ten consequential iterations the algorithm is stopped.

The effective capability of the GA to perform well and to optimize the number of cards mainly resides in the evaluation of the fitness function for each individual. In this study, the fitness is obtained by means of a Monte Carlo simulation process. In brief, using the commercial simulation software "Simul8," the whole POLCA model has been accurately represented. The GA has been built using the free Pascal Object-Oriented Programming language. At each iteration, every individual is used to pass the proper parameters (i.e., the number of cards for all the loops) to the simulation process. The simulation is then executed, and the results are collected to evaluate the fitness function.

As already stated, there are many different configurations that minimize the TTT with different values of WIP. Therefore, the fitness function has been evaluated with three well-defined objectives in mind:

1. Minimizing the average TTT;
2. Reducing the WIP, by reducing the number of cards for each loop;
3. Maximizing the overall throughput.

The simulation model has been programmed to return, at the end of each run, the average TTT value, the average WIP

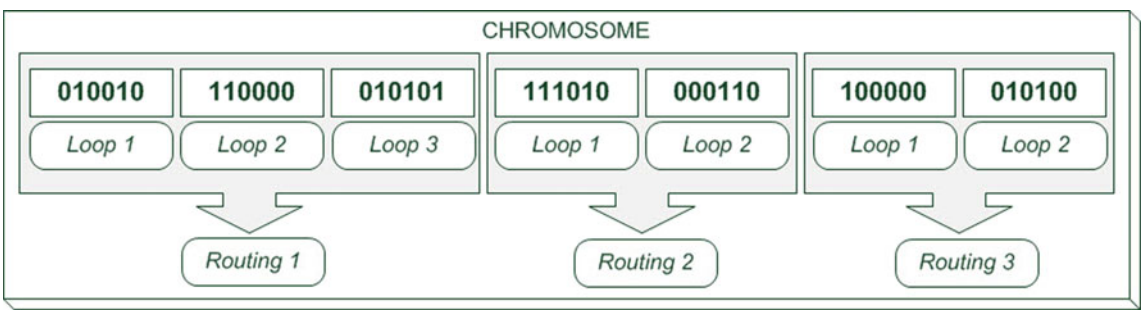

Fig. 4 Structure of a chromosome

and the throughput value for each run. The fitness function has been evaluated as follows:

$$f(i) = \overline{TTT}_i + \sum_j \left(\overline{WIP}_{ij} \times \bar{t}_j \right) + \frac{N_{Ci}}{\sum_j \overline{WIP}_{ij}} \quad (1)$$

where:

- *f(i)* is the fitness of the *i*th individual;
- $\overline{TTT}_i$ is the average Total Throughput Time for the *i*th individual, as computed by the simulation. It assures that better individuals, having low values of both OPT and STT, have better probabilities of being selected. Also, the contribution of the OPT to the TTT assures that the overall throughput is effectively maximized adding penalties to those configurations that have greater values of the OPT;
- the second addend represents the penalty due to the number of items that are on average within the shop floor. The calculation is performed for all the *j* routings separately, considering the average cycle time (t_j) for the routing itself;
- the last term represents the penalty due to the number of cards allocated within all the loops of all the *j* routings for the individual. This has been introduced to move the GA toward those configurations that use the lower number of cards. To assure that greater penalties are given to those configurations that have a large number of cards and do not use the most of them, the actual number of cards N_{Ci} is compared to the value of the total average WIP_i. One may observe that this behavior is secured by the second addend, but test has shown that the introduction of the last one significantly improves and speeds up the convergence of the GA.

Finally, the already mentioned neighborhood search deserves a brief explanation. Indeed, it is well known from literature [20] that GAs perform exceptionally when moving to good solutions with a coarse resolution. They become ineffective when local minima have to be located exactly. Consequently, several different neighboring search procedures have been used widely to enhance the heuristic. In this context, such procedure has been used both at the end of each epoch, trying to improve the best individual of the moment, and at the end of the GA execution, both if all epochs have been run or if the termination condition has been met, to verify that the found solution actually represents a minimum. Technically speaking, the procedure used at the end of each iteration randomly selects a loop within the best available chromosome and increases or decreases (with equal probability) the number of cards in the loop by one unit. Then, a simulation is run and the fitness is evaluated again. If this is better than the original one, the new modified chromosome is saved within the population. On the contrary, the neighborhood procedure used at the end of the algorithm consists of an iteration during which the number of cards in all loops is increased or decreased, by one unit at time, up to the 10 % of the number itself. The procedure has the major drawback of being quite slow, but its use is fully justified by the fact that it finds extremely improved solutions in most cases and that the number of simulations required is tolerable. In fact, the number of cards within the loops of the best chromosome at the end of the trial is quite small, and the neighborhood search entails a maximum of one or two simulations for each loop. Globally, this is comparable to performing a further iteration on the whole population set. The complete flow diagram of the proposed algorithm is reported in Fig. 5.

In brief, the algorithm can be resumed as follows:

- The initial population is randomly created;
- In the first loop (index *i*, in Fig. 5), the algorithm cycles through the individuals and uses them to set the initial number of cards for the simulation. A simulation is run for each chromosome and the corresponding fitness is saved;
- A second loop (index *j*) is started to cycle through the epochs;
  - A nested loop (index *z*) cycles through the individuals to generate the new population at the given epoch;
    - Two individuals are selected by using the classic roulette wheel procedure, based on the individual fitness;
    - Crossover and mutation operators are applied (as previously explained, two equally probable crossover and mutation operators are used);
    - The new fitness is evaluated and the *z*th chromosome is saved within the new population;
  - Before moving to the next epoch a "short" neighborhood search is performed starting from the best individual within the new population;
    - A single simulation is run and the original individual is overwritten if the neighbor has a better fitness;
  - Elitism procedure is performed and the best individual is preserved for the future generations;
  - If the maximum number of epochs is reached or a termination condition applies the outer loop (index *j*) is terminated;
- An extensive neighborhood procedure is run (as explained above).

4 The case study

The proposed approach has been tested verifying the GA outcomes against the unconstrained POLCA model, to show that it actually allows a significant reduction of the STT when constraints to the workload are introduced. Such restriction, in turn, increases the OPT, due to the fact that some orders have to wait for a free card to be allowed to enter the production system. However, the average reduction of the STT results to be greater than the corresponding increase in the OPT. To validate the experiment, the POLCA-controlled system has been as well compared with an identical model controlled by a

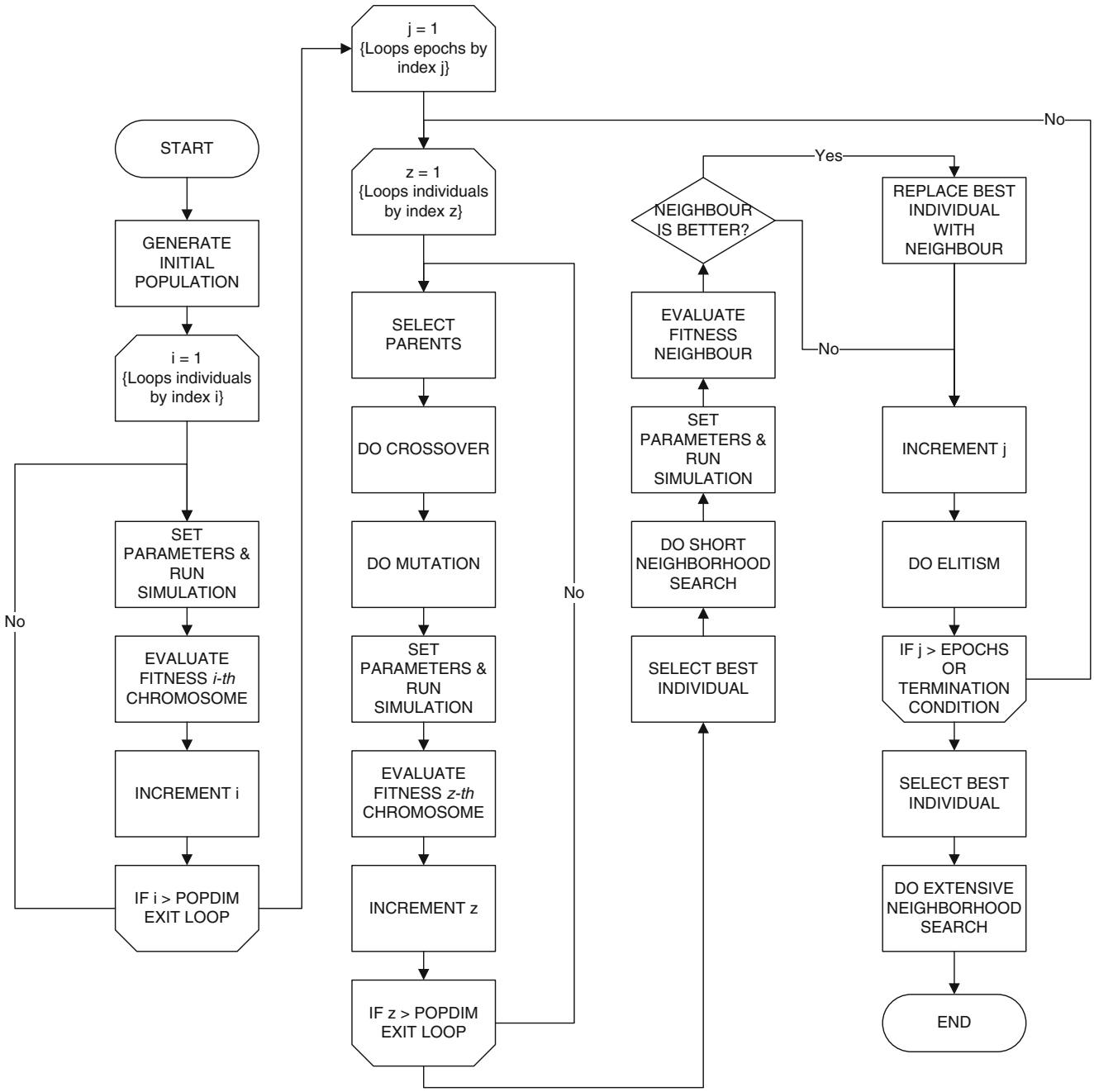

Fig. 5 Algorithm flow diagram

multi-CONWIP (*m*-CONWIP) mechanism with an appropriate "forward-looking loading rule." In particular, the *Work In Next Queue* rule has been adopted to grant that the items are favored within routings for which capacity is available in the succeeding work centers. This helps the *m*-CONWIP to reduce the OPT and to improve its performance. The *m*-CONWIP has already proved (see for instance Gems and Riezebos [2]) to be effective in balancing the workload among the available workstations under particular circumstances. In particular, the authors verified that it behaved generally better than the POLCA model. Therefore, it may be used as a reliable benchmark to verify the capabilities of the proposed approach. The optimization of the number of cards, in this case, has been obtained by means of the same genetic algorithm, provided that the chromosome length is appropriately adjusted due to the presence of only three cards loops. Owing to this, the present work aims to put in evidence both the performance of the POLCA control and the ease with which the optimal number of cards can be set.

To begin with, a first set of tests has been used to determine the optimal configuration of the GA for both cases. In particular, the major issue to be faced at this stage dealt with the population

size and the number of epochs (iterations) to be used. Indeed, the simulation-driven GA tends to become really slow as soon as the number of individuals to be evaluated grows up. This is due to the fact that each chromosome requires an independent run of the simulator and this is a really time-consuming activity. The simulation itself had been previously validated with a significant number of tests and, following the outcomes of such experiments, each simulation has a warm-up period of 25,000 min, to eliminate the initial transient, and a result collection period of 100,000 min. When the GA requires the simulation to be run, this takes a few seconds to open the software package, load the initial values, and get the results back. Necessarily, this grows up linearly with the population size and the epochs number. The optimal values for the GA parameters have been set, by means of trials, as follows:

- The initial crossover and mutation ratios have been selected equal to C_R=0.70 and M_R=0.15;
- The number of epochs and the population sizes, instead, have been selected after testing several configurations (20, 30, 40, and 50 individuals with the epochs varying between 50 and 100 in steps of 10). Epochs have been kept low enough to avoid excessive simulation durations;
- After determining the "optimal" number of epochs and the population size, the crossover and mutation ratios have been further investigated to improve the obtained solution. In particular, their values have been studied in the ranges $0.60 \leq C_R \leq 0.80$ (in steps of 0.05) and $0.10 \leq C_R \leq 0.30$ (in steps of 0.05).

Though the adopted procedure is coarse and far from optimality, it can be considered sufficient enough to the aim of the present work. Indeed, the focus is not that of optimizing the GA, but that of showing it can be effectively used to optimize the unbalanced POLCA-controlled production system. In brief, the performed tests have shown clearly that a standard GA, configured with almost standard values of its parameters, is able to optimize the production system under exam. Moreover, the results remain almost steady while modifying the parameters, indicating that the algorithm can be considered robust and effective. In particular, when the parameters have been tested within the above mentioned ranges, the corresponding variation of the fitness of the best individuals remained within the 6 % of the "absolute" best configuration. This, in turn, is characterized by the following values:

- Population size=30;
- Epochs=50;
- C_R=0.65;
- M_R=0.20;

As already stated, if the best individual remains unchanged for ten consecutive epochs the GA is stopped. At each iteration, all the population is renewed and the elite is used to reinsert the best individual of the moment. It is important to pinpoint that though a number of interesting studies are available that deal with the convergence analysis and the stopping criteria of GAs (see for instance [21, 22]) in the present work a "brute force" approach has been preferred, mainly due to the fact that the epochs number had to be kept low to avoid too long computing times. Therefore, an in-depth analysis on convergence and on the stopping criteria would have been costly and almost ineffective. The theoretical values, indeed, would result far larger than those actually used. This would result, in the end, in extremely large computing times. However, the great number of test showed that the outcomes can be considered almost reliable.

5 Results and discussion

The test has been performed running 100 trials for the single CONWIP, the *m*-CONWIP, and the POLCA models with 100 different sets of pseudo-random numbers.

As an example, let us consider the results derived from the first set of pseudo-random numbers. In this case, the unconstrained situation is characterized by the following average values (obtained by simulation):

- OPT=0;
- STT=TTT=39.19 (min);
- WIP≈36.

Test have shown that the standard CONWIP (single loop) is not able to improve this situation and that the TTT value increases (due to the increase of the OPT) as soon as constraints are imposed, in perfect accordance to the outcomes of Germs and Riezebos [2]. On the contrary, the *m*-CONWIP model outperforms the single loop approach and is able to improve the situation, finding an optimal solution in 14 epochs. These improvements are resumed in Table 1.

The TTT is greater with respect to the one in the unconstrained case, but the number of cards is very little, granting a very high throughput with low values of WIP, whose values are, on average, 12. The best chromosome, showing the number of cards for the three loops, is the following:

[[11] [5] [1]]

The same pseudo-random numbers were used to verify the POLCA model, and results are shown in Table 2.

The TTT is about 1.5 min lower than in *m*-CONWIP case, and the number of cards has decreased enough, granting also in this case a very good throughput with low values of WIP (15, on average), as confirmed by the trend of the fitness function. The best chromosome, having the structure reported in Fig. 4, is the following:

[[8 7 4] [4 10] [25]]

It is noteworthy that while the TTT shows a little but not negligible improvement with respect to the *m*-CONWIP

Table 1 Improvements during a trial of the *m*-CONWIP model

Epoch	TTT	Fitness	N_C
1	41.54	76.43	94
2	40.16	66.59	64
5	40.16	65.01	45
8	40.33	64.54	39
9	41.05	64.54	38
12	42.85	63.02	22
14	41.04	62.89	17

model, the fitness function shows a greater difference. This may be explained, in accordance to Eq. 1, with lower values of the average WIP. Thus, the test confirms a very good behavior of the POLCA model with the "longest queue" selection rule in the case of unbalanced systems.

An excerpt of the whole experiment outcomes, where the best solutions for the first 20 trials are summarized for both models, is reported in Table 3.

Interestingly, the GA found the same optimal solution in 95 cases out of 100 in the case of the POLCA system and all the times in the case of the *m*-CONWIP. In the five cases where the best chromosome differed (as, for instance, in trial 2), the following individual was found:

[[8 10 4] [4 10] [25]]

This is strictly similar to the previous one, differing for only three cards in the second loop of the first routing. However, the values of the TTT and of the fitness function always showed to be better than in the *m*-CONWIP case.

This confirms that the algorithm was effectively able to converge to very good individuals. Besides, the outcomes show that the POLCA results are generally better than those found the *m*-CONWIP model in all cases. This is a major result, as it shows that the POLCA model with the "longest queue" rule, in the case of unbalanced routings, performs better than the *m*-CONWIP, whereas Germs and Riezebos [2] verified that in the case of balanced systems, the latter behaved decidedly better. This is due to the fact that POLCA shorter loops have a better capability of recovering and balancing the orders within the shop floor with respect to the longer loops of the *m*-CONWIP. While this aspect was irrelevant in a perfectly balanced system, in the present case it became the determining factor. Therefore, being the setting of POLCA cards a major operating difficulty, above all if compared to standard CONWIP or even to *m*-CONWIP-controlled systems, the proposed approach may provide a valid support tool to overcome this limitation making the most of POLCA capabilities in many manufacturing configurations.

In conclusion, it clearly emerges from these experiments that heuristic methods (other than GAs could be used as well) and simulation may provide a useful tool to evaluate the impact of design alternatives for the POLCA systems and can certainly be adopted as a decision support instrument, able to assist managers in the control of complex manufacturing systems. Indeed, they dramatically reduce the difficulty that one may encounter in properly setting the number of cards within the loops, keeping in mind that the distribution of cards has a considerable effect on the performance of the controlled systems. The performed tests, though they could be further improved with a finer setting of the parameters and a higher number of epochs for the GA, confirmed a very good behavior of the POLCA model with the "longest queue" selection rule in the case of unbalanced systems. In particular, while the TTT

Table 2 Improvements during a trial of the POLCA model

Epoch	TTT	Fitness	N_C
1	39.68	82.78	265
2	39.78	78.37	207
4	40.67	69.82	95
7	41.03	67.31	63
9	40.78	65.55	44
11	39.44	58.53	49
14	39.44	58.48	46
16	39.39	58.58	40

Table 3 Results of the first 20 trials

	m-CONWIP			POLCA		
Trial	TTT	Fitness	N_C	TTT	Fitness	N_C
1	41.04	62.89	17	39.39	58.48	40
2	98.76	120.57	17	96.74	116.76	43
3	44.81	65.05	17	43.28	61.92	40
4	63.25	83.91	17	60.99	79.96	40
5	39.46	60.36	17	37.93	57.29	40
6	43.67	65.12	17	42.12	59.16	40
7	48.23	71.17	17	47.91	69.67	40
8	51.56	82.31	17	44.11	76.45	40
9	59.45	94.01	17	59.13	91.43	40
10	43.42	61.12	17	41.88	59.80	40
11	44.82	61.77	17	42.07	56.18	40
12	57.82	86.45	17	53.12	83.11	40
13	46.32	68.13	17	46.01	63.67	40
14	38.98	60.01	17	36.57	59.63	40
15	39.12	63.77	17	37.63	59.89	40
16	47.54	65.62	17	46.13	64.48	40
17	44.16	71.22	17	41.05	67.02	40
18	42.02	57.11	17	40.78	57.02	40
19	49.15	63.52	17	48.57	62.37	40
20	47.87	62.21	17	47.53	58.79	40

was kept near the optimal values, both the STT and the WIP were effectively reduced, and this result was obtained by means of a fast simulation-driven GA that eliminated completely the burden of setting the correct number of cards within the many control loops required by the approach.

6 Conclusions

POLCA is a hybrid card-based pull production control mechanism characterized by multiple short control loops, each cycling over two successive workstations and overlapping two by two. This distinctive feature makes it able both to control the overall workload and to balance it among the various workstations. In the case of balanced systems, POLCA is very effective in reducing the Total Throughput Time (TTT) but often, in real-world applications, the systems to be controlled are designed to process units with very different routings, each with significantly different probability to occur. In all these situations, queues may form in front of certain workstations, whereas some other machines may be temporarily starving and waiting for units to be worked. When these systems have to be optimized, one of the major drawbacks to be faced is represented by the setting of the number of cards that are required within each different loop. Little law is usually adopted to infer this number from historical data. However, due both to the variability of the quantities involved in the calculation and to the intrinsic complexity of the systems to be controlled, the obtained number is usually far from the optimum. In the case of balanced systems, a simple "trial and error" procedure can be adopted, but it is as much true that when routings and arrival times are different this method becomes ineffective. This is mainly due to the fact that in this case each loop needs a potentially different number of cards and that, when the number of loops grows up, the trial and error procedure would be time-consuming and unwieldy. The present paper has therefore addressed the optimization of the number of cards by means of a simulation-driven genetic algorithm. Though the algorithm in itself does not represent the objective of the work, the GA has been selected due to the fact that it is well known, widely used, and effective. The objective was that of showing the effective capability of the proposed approach to individuate the correct number of cards and to reduce the overall TTT and the average WIP, both with respect to the unconstrained model and to a multi-CONWIP-controlled model. The results show that the POLCA outcomes are generally better than those found the m-CONWIP model in all cases. This is a major result, as it shows that the POLCA model with the "longest queue" rule, in the case of unbalanced routings, performs better than the m-CONWIP, whereas in the case of balanced systems, the latter had behaved absolutely better. This may be ascribed to the fact that POLCA shorter loops have a better capability of recovering and balancing the orders within the shop floor with respect to the longer loops of the m-CONWIP. While this aspect was irrelevant in a perfectly balanced system, in the present case it became the determining factor. The setting of POLCA cards is undoubtedly a major operating difficulty, above all if compared to standard CONWIP or even to m-CONWIP-controlled systems. Therefore, the proposed approach may provide a valid support tool to overcome this limitation making the most of POLCA capabilities in many manufacturing configurations.

As actual production systems are characterized by rapid modifications to the production mix and to the order arrival times, a possible future work could be addressed to the identification of an auto-adaptive tool able to rapidly re-configure the number of cards within loops as soon as those modifications occur.

References

1. Suri R (1998) Quick Response manufacturing: a companywide approach to reducing leadtimes. Productivity Press, Portland
2. Germs R, Riezebos J (2010) Workload balancing capability of pull systems in MTO production. Int J Prod Res 48(8):2345–2360
3. Koulouriotis DE, Xanthopoulos AS, Tourassis VD (2010) Simulation optimization of pull control policies for serial manufacturing lines and assembly manufacturing systems using genetic algorithms. Int J Prod Res 48(10):2887–2912
4. Gstettner S, Kuhn H (1996) Analysis of production control systems Kanban and CONWIP. Int J Prod Res 34(11):3253–3273
5. Selvaraj, N. (2009) Determining the number of Kanbans in EKCS: a simulation modeling approach. *Proceedings of the International Multi-Conference of Engineers and Computer Scientists IMECS*, Vol. II, Hong Kong
6. Alabas C, Altiparmak F, Dengiz B (2000) The optimization of number of kanbans with genetic algorithms, simulated annealing and tabu search. Congress on Evolutionary Computation (CEC2000), July 16–19, 2000, San Diego, USA, 580–585
7. Köchel P, Nieländer U (2002) Kanban optimization by simulation and evolution. Prod Plan Control 13(8):725–734
8. Hu S, Yi W (2009) Simulation and optimization of kanban system for steel rolling. *International Workshop on Intelligent Systems and Applications, ISA 2009*, 1–4
9. Al-Tahat M, Liverani A, Persiani F, Caligiana G (2002) Using genetic algorithms to optimize kanban-based production system. The European Applied Business Research Conference, Rothenburg
10. Huang M, Wang D, Ip WH (1998) A simulation and comparative study of the CONWIP, Kanban and MRP production control systems in a cold rolling plant. Prod Plan Control 9(8):803–812
11. Hopp WJ, Roof ML (1998) Setting WIP levels with statistical throughput control in CONWIP production lines. Int J Prod Res 36(4):867–882
12. Herer YT, Masin M (1997) Mathematical programming formulation of CONWIP-based production lines and relationships to MRP. Int J Prod Res 35(4):1067–1076
13. Golany B, Dar-El EM (1999) Controlling shop floor operations in a multi-family, multi-cell manufacturing environment through constant WIP. IIE Trans 31(8):771–781
14. Ajorlou, S., Shams, I., Aryanezhad, M.G. (2011) Optimization of a multiproduct CONWIP-based manufacturing system using artificial

bee colony approach. *Proceedings of the International Multi-Conference of Engineers and Computer Scientists IMECS*, Vol. II, Hong Kong
15. Riezebos, J. (2009) Design of POLCA material control systems. *Proceedings of the International Conference on Intelligent Manufacturing & Logistics Systems IML2009*, Waseda University, Kitakyushu, Japan
16. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional, Upper Saddle River, Reading
17. Aytug H, Khouja M, Vergara FE (2003) Use of genetic algorithms to solve production and operations management problems: a review. Int J Prod Res 41(17):3955–4009
18. Dostál P (2013) The use of soft computing for optimization in business, economics, and finance. Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance, 41–86
19. Senvar O, Turanoglu E, Kahraman C (2013) Usage of metaheuristics in engineering: a literature review. *Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance*, 484–528
20. Reeves CR (1994) Genetic algorithms and neighborhood search. Evol Comput Lect Notes Comput Sci 865:115–130
21. Rudolph G (1994) Convergence analysis of canonical genetic algorithms. IEEE Trans Neural Netw 5:96–101
22. Aytug H, Koehler GJ (2000) New stopping criterion for genetic algorithms. Eur J Oper Res 126:662–674