J Intell Manuf (2013) 24:1145–1156
DOI 10.1007/s10845-012-0646-5

Artificial bee colony algorithm for CONWIP production control system in a multi-product multi-machine manufacturing environment

Saeede Ajorlou · Issac Shams

Received: 27 October 2011 / Accepted: 16 April 2012 / Published online: 29 April 2012

Abstract In this paper, a multi-product multi-machine serial production line operated under a constant-work-in-process protocol is considered. A mathematical model for the system is first presented, and then an artificial bee colony optimization algorithm is applied to *simultaneously* find the optimal work-in-process inventory level as well as job sequence order in order to minimize the overall makespan time. Unlike many existing approaches, which are based on deterministic search algorithms such as nonlinear programming and mixed integer programming, the proposed method does not use a linearized or simplified model of the system. A production line simulator implemented on MATLAB is, instead, employed to model the highly nonlinear dynamics of the production line and is used to evaluate the candidate solutions. The efficiency of the proposed approach, even for systems of large sizes, is validated via numerical simulations.

Keywords Artificial bee colony · CONWIP · Production control system

Introduction

Choosing the effective production control strategy is a vital decision for managers in today's competitive world class manufacturing (WCM) arena.Companies are now continuously facing serious challenges to improve their excellence by increasing their overall productivity, reacting more quickly and efficiently to the frequent changes of their customer expectations, and producing with higher profitability and quality but lower waste and cost. To this end, various manufacturing environments have been introduced during the past two decades, among which are included material requirements planning (MRP), manufacturing resource planning (MRP II), theory of constraints (TOC), just-in-time (JIT), agile manufacturing, CONstant work-in-process (CONWIP), and more recently enterprize resource planning (ERP).

The main focus of this study is on CONWIP production control systems. From a control policy point of view, we may classify the modern production systems into pure push-based, pure pull-based, and hybrid push/pull environment. A push control strategy (such as MRP) relies on existing inventory level, part lead times, bill of materials, and forecasted demands for scheduling its production jobs, and hence has large lots, high inventories and/or wastes, and too much work-in-process (WIP). Conversely, in a pull system, the start of a new job is trigged by the end product orders with backward information flow from the last work station to the initial one. Therefore, it has small lots, low inventories and/or wastes, and less WIP. In general, as mentioned in Li et al. (2007), pull-based systems control WIP and observe throughput (total number of finished jobs per period) by scheduling the releases whilst push-based systems observe WIP and control throughput (TH) by authorizing the releases. With respect to the rich literature about comparing these two manufacturing scenarios [see, e.g., Spearman et al. (1989, 1990), and Spearman and Zazanis (1992)] most researchers choose pull over push systems because of three factors, namely, efficiency, robustness, and observability. Having combined the advantages of push systems (broad applicability to industrial and manufacturing firms) with those of pull ones (superior performances in terms of JIT operational and managerial concepts), some researchers proposed a number of hybrid push/pull production control systems such as Drum-Buffer-Rope (DBR) model in Goldratt and Cox (1985), CONWIP model in Spearman et al.

S. Ajorlou (✉) · I. Shams
Wayne State University, Detroit, MI, USA
e-mail: er7212@wayne.edu

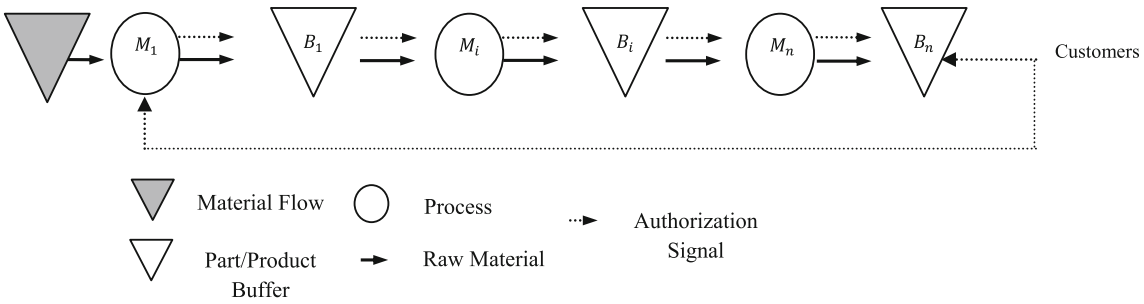

Fig. 1 A simple CONWIP scheme

(1990), and Generic Kanban model in Chang and Yih (1994). Taking into account the benefits and drawbacks of each policy, the CONWIP system has attracted much more attention due to its simplicity of implementation.

The primary mechanism of the CONWIP system is described in Fig. 1. Similar to Kanban systems, CONWIP uses cards to manage the number of WIPs. However, there is only one set of cards flowing backward from the end of the production line to its beginning in order to precisely monitor current inventory level of the system under study. It is assumed that as long as all required manufacturing modules are accessible, the requested demands are taken into account for early production. Inasmuch as no job can enter the system without its related card, once completed in the last station, the card is released and then sent back again to the first station, where it is attached to the subsequent job to be processed. Obviously, the system is identical to Kanban in that the production of the first workstation is also activated by the demand. In contrast, it differs from Kanban system in the sense that CONWIP is only pulled between the last and the first workstation, so it may be considered as a single-stage Kanban. Accordingly, for sake of this reason, as stated in Gaury et al. (2000), the implementation, modeling and optimization of CONWIP production strategy is much easier than Kanban. For more details about CONWIP systems and their differences with Kanban, we refer the readers to Gstettner and Kuhn (1996), Hopp and Roof (1998), and Marek et al. (2001).

It is worth to mention that in order to effectively set up a CONWIP control system in a specific manufacturing environment, some common issues have to carefully be addressed. The most crucial ones are forecasting the backlog list (which gives the sequence of orders to be introduced into the line), determining the number of cards, and sequencing the jobs in the system. Also due to some of its important merits, such as flexibility and robustness in dynamic and uncertain environments, CONWIP production control system has been applied not only to various manufacturing firms but also to different echelons of a supply chain in recent years (Knolmayer et al. 2002). And researches around these production systems are keep continued.

The remainder of this paper is organized as follows. The detailed literature review is presented in Section "Literature Review". Section "Problem Formulation" describes the problem formulation and model development. Artificial Bee Colony optimization algorithm and its application for solving the model are outlined in Section "Methodology". Section "Numerical Examples" verifies the efficiency of the proposed approach via numerical examples. Finally concluding remarks and future research directions are drawn in the last section.

Literature review

Different aspects of the CONWIP system such as operation, applicability, and also comparisons of CONWIP with other production systems can be found in the literature, and are well classified by Framinan et al. (2003). Lambrecht and Segaert (1990) use CONWIP in a merging/assembly line and also make some comparisons with the single Kanban system. It is shown there that, with the use of both analytical and simulation models, CONWIP surpasses single Kanban in case of variable processing times. Hopp and Spearman (1991) propose the CONWIP flowshop system in which the machines are subject to exponential failures and repairs, but processing times are deterministic. They compare two methods for estimating throughput as a function of WIP level with the help of closed queueing network. Bonvik et al. (1997) point out that, in a CONWIP system, high inventories may appear in front of some machines because of high processing time or machine breakdown. This occurs since there is no mechanism present to individually limit the WIP level in each part of a system. In order to overcome this shortcoming and combine the advantages of CONWIP (low overall WIP with high throughput) with those of Kanban, they propose a Hybrid Kanban/CONWIP control strategy by adding Kanban cells to CONWIP. Hopp and Roof (1998) propose a new method, namely statistical throughput control (STC), which uses real-time data to adjust WIP level under a make-to-order CONWIP protocol subject to environmental changes. This study can be categorized as card controlling, which deals with devising some

rules in order to maintain or change the current number of cards with respect to certain events such as abrupt changes in the demand. Although they employ simulation and consider single and multiple products with shared resources and assembly systems, their method does not rely on steady-state conditions.

Golany et al. (1999) propose a mathematical model to address the optimal number of cards and job sequencing simultaneously in a multi-cell, multi-family production environment with different routes, and solve it via a simulated annealing (SA) heuristic. Moreover, they compare two variations of CONWIP control policies, the multi-loop CONWIP system (in which containers are restricted to stay in given cells) and the single-loop CONWIP system (in which containers can circulate everywhere within the system), and show the superiority of the latter in all scenarios through simulations. Framinan et al. (2006) suggest a new procedure for card controlling which can be applied both to make-to-stock and make-to-order environments, and examine it on the CONWIP system. This procedure deals with adding or subtracting extra cards along with consistently monitoring throughput rate or service level. And in order to reach a target throughput rate, they use only two parameters, namely, the initial Kanban cards of the CONWIP system and the number of maximum (initial) extra cards. They also demonstrate that their method is robust pertaining to the values of the required parameters. In (Herer and Masin 1997) a deterministic mathematical programming model is developed for a multi-product CONWIP flowshop system in order to find the optimal job order and schedule, given demand and forecasted rate of throughput, via linear programming (LP). The goal of this model is to minimize total cost function consisting of finished goods holding cost, shortage cost, WIP holding cost, and overtime cost. However, they do not consider lot sizes and the effects of bottleneck machine on job orders, and do not propose an algorithm to efficiently solve their model. Luh et al. (2000) study Sikorsky Aircraft as a single CONWIP job shop production line, and develop a mathematical programming model in order to minimize weighted penalties on tardiness and earliness at a given WIP level. The model is partially solved by dynamic programming (DP) and heuristic methods with the help of Lagrangian Relaxation. Zhang and Chen (2001) propose an integer nonlinear mathematical programming method to find the optimal job sequencing and lot sizes in a single serial production line with a specified number of cards under a CONWIP protocol. Unbalanced workload term has been included in this model. Having linearized the objective function, they solve the model directly for a number of examples using LINGO software. Cao and Chen (2005) consider an assembly station feeding by two parallel fabrication lines. The model is partially linearized and a nonlinear mixed integer programming algorithm is proposed in order to obtain simultaneously the optimal job sequencing and lot sizes. However, WIP level and number of containers are not discussed in their study. Li et al. (2007) consider two distinct mathematical models on a single serial CONWIP line system, one is a make-to-stock environment in which it is desired to simultaneously settle the optimal lot size of each item on the part list and job sequencing, and another is a both make-to-stock and make-to-order environment with known part list for which it is aimed to determine the number of cards and job sequencing. The objective function of the first model is to minimize the setup costs along with unbalanced workload at the bottleneck machine, while the latter is to minimize the total makespan. They have applied SA algorithm to solve the models and have verified their method by computational results obtained from a series of examples.

More recently, Ip et al. (2007) consider a lamp assembly production line making more than 100 kinds of products with discrete distribution processing time and demands. They formulate the optimization model in a way that minimizes the total cost of production process while keeping the average shortage probability of demands below a threshold value. Having examined a single-loop and multi-loop CONWIP protocols for the assembly line, the authors clearly illustrates the predominance of the former policy over the latter one. Khojasteh-Ghamari (2010) develops a framework for performance analysis between KANBAN and CONWIP systems based on Activity Interaction Diagram and Critical Circuit analysis. The author displays that in those production control mechanism, the performance measures such as system throughput is strongly affected by critical circuit; hence parameters like initial inventories at workstations, Kanban card distribution in the KANBAN system, and total number of circulating cards in the CONWIP system may change the placement of the critical circuit and subsequently affects the system practice. Furthermore, Based on Little's law and theory of token transaction systems, Sato and Khojasteh-Ghamari (2010) propose an integrated framework for design of card-based production control systems. The authors study the interdependent dynamics of average WIP, average cycle time and average throughput rate in a sub-network of production process with FIFO control policy and periodic behavior. Having employed the critical circuit concept in feedback controlled manufacturing structure, they demonstrate that there is no universal superiority between CONWIP and KANBAN especially in a tree-shaped mechanism.

Considering the high complexity of production system models, typically a simplified model is instead used to find the optimal WIP level and job sequencing in a CONWIP system. Linearizing the model, and considering a single bottleneck machine to be able to model the unbalanced workload constraints are a few examples of such simplifying assumptions (Li et al. 2007; Cao and Chen 2005; Herer and Masin 1997). However, even the corresponding optimization problem are often NP-complete (Garey and Johnson 1979), and

hence are intractable for close to real world problems which dealt with large number of parts, machines, and production lines. Also less attention is paid in the literature to simultaneously finding the optimal WIP level and job sequencing due to this complexity. To overcome these shortcomings, we apply the ABC algorithm (Karaboga 2005), a novel heuristic optimization approach, to simultaneously finding the optimal WIP level and job sequencing with the aim of minimization the overall makespan time in a multi-product multi-machine serial production line under a CONWIP protocol. The highly nonlinear dynamics of the system is modeled via a production line simulator implemented on MATLAB, and is used to evaluate the candidate solutions. Numerical examples validate the efficacy of the proposed approach even for systems of large size.

Problem formulation

In this paper we consider a single serial production line with a number of machines processing a number of different part types. Each part is to be processed by all the machines sequentially. There is a process time associated with each pair of (machine, part type), which may differ from part to part for a certain machine. Also, there is a set up time required to change the line from processing one part type to another type. It is also assumed that the line uses a CONWIP production control strategy. In addition, there is a demand list determining the number of required parts of each type. And the following assumptions are further presumed to hold:

- The process is not interrupted due to the limited raw material
- There is no machine breakdown
- Set up times and process times are fixed and deterministic
- Parts are processed with all machines sequentially

The objective function is to find the optimal WIP level (card setting) and job sequencing discipline in order to minimize the total makespan time. To the best of the authors' knowledge and as pointed out in Section "Literature Review", card setting and job sequencing are treated separately in the literature. However, it is beneficial to consider these two problems at the same time since they both affect the performance of the CONWIP system. This problem can be categorized as NP-complete, and hence is intractable for close to real world problems dealing with large number of part types, several machines and production lines. This is due to the fact that flow shop sequencing problems are typically NP-hard (Garey and Johnson 1979). Therefore the common practice is to apply a non-traditional heuristic approach like Genetic Algorithm (GA), hill climbing, tabu search, SA, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Sheep Flock Heredity Algorithm (SFHA), and Biogeography-Based Optimization (BBO) to address the optimization problem.

In this research, we choose ABC algorithm due to its simplicity and ease of implementation (Singh 2009) and also its proven excellence to other non-traditional population-based algorithms in terms of employing fewer control parameters and efficiency for solving multimodal and multidimensional optimization problems. We refer readers to Karaboga and Akay (2009) for an elaborate comparative study of ABC to Evolution Strategy (ES), GA, PSO, and Differential Evolution (DE).

The following parameters are used in developing the mathematical model of the system:

n the number of different part types
P_i part type i, $i = 1, \ldots, n$
m the number of machines in the line
d_i the number of required parts of type P_i in the demand list, $i = 1, \ldots, n$
β weight associated with the unbalanced workload cost
T_{ij} the setup time required to switch the line from processing part type P_i to part type P_j, $i, j = 1, \ldots, n$, and $i \neq j$. $T_{ij} = 0$ for $i = j$
P_{ij} the processing time of machine i on part type P_j, $i = 1, \ldots, m$ and $j = 1, \ldots, n$

The decision variables are:

$$y_{ij} = \begin{cases} 1, \text{ if part type } P_i \text{ is followed by part type } P_j, \\ \quad i, j = 1, \ldots, n, \text{ and } i \neq j \\ 0, \text{ otherwise} \end{cases}$$

ϵ_k the difference between the line workload on kth batch and $(k-1)$th batch, $k = 2, \ldots, n$

The decision variables y_{ij} should satisfy following constraints:

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ij} = 1, \quad i = 1, \ldots, n \tag{1}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} y_{ij} = 1, \quad j = 1, \ldots, n \tag{2}$$

Following these notations and using Mixed Integer Non-Linear Programming (MINLP) approach, the optimization problem can be formulated as

$$\min \left\{ \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ij} T_{ij} + \beta \sum_{k=2}^{n} \epsilon_k \right\} \tag{3}$$

The first term in (3) corresponds to the setup time required to switch from processing the current part type to the next one for a specific job sequencing. The second term is a penalty term to achieve production smoothness, where β is the weight associated with the unbalanced workload cost. This means that at each setup, the line should process an amount of work similar to the previous setup. This is one of the key CONWIP requirements and can be achieved by requiring the following constraint for the solution of (3).

$$|W_k - W_{k-1}| < \epsilon_k, \quad k = 2, \ldots, n \tag{4}$$

where W_k and W_{k-1} are the line workloads on the kth and $(k-1)$th part batches. Calculating W_k is highly complicated and depends on many parameters such as the part processing times P_{ij}, setup times T_{ij}, WIP level, demand list, and job sequencing. Deterministic search algorithms, such as integer programming require simplified (and usually linear) estimates for W_k. This can be done (e.g. Cao and Chen) by assuming that the line work load on a part batch is determined by the bottleneck machine in the line, and there is only one bottleneck station. Under these assumptions W_k can be estimated as

$$W_k = \sum_{i=1}^{n} x_{ik} \left(d_i P_i^b + \sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ij} T_{ij} \right) \tag{5}$$

where P_i^b denotes the process time of part P_i by the bottleneck machine and

$$x_{ik} = \begin{cases} 1, \text{ if the } k\text{th part batch is of type } P_i \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

x_{ik}'s should satisfy the following constraints:

$$\sum_{i=1}^{n} x_{ik} = 1, \quad k = 1, \ldots, n \tag{7}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \quad i = 1, \ldots, n \tag{8}$$

$$\sum_{k=2}^{n} x_{i(k-1)} x_{jk} = y_{ij}, \quad i, j = 1, \ldots, n \ (i \neq j) \tag{9}$$

However, heuristic search algorithms such as the one employed in the present work, do not require such a simplified formula and can easily handle problems involving complicated nonlinear models. This is one of the advantages of the present work over the existing approaches to CONWIP production planning. In this work, we use a production line simulator implemented in MATLAB to calculate the line workload on part batches under each setting.

Methodology

Artificial bee colony

Since its introduction by Karaboga (2005), ABC optimization approach has been successfully utilized in many optimization problems arose from various disciplines including supply chain management (Kumar et al. 2010), production scheduling (Pan et al. 2011), clustering (Zhang et al. 2010), vehicle routing problem (Szeto et al. 2011), set covering (Sundar and Singh 2010), reliability redundancy allocation (Yeh and Hsieh 2011), and large scale engineering design optimization (Akay and Karaboga 2010). In this evolutionary algorithm, there are three foraging groups of bees: employed bees, onlooker bees, and scout bees. A bee going to the food source which is visited by itself in the last round is called an employed bee. An onlooker bee, on the other hand, waits on the dance floor to gather information about the positions and the nectar (fitness) of the food sources from the employed bees and then chooses one of them probabilistically. A bee doing a random search for food sources is called a scout bee. Employed and onlooker bees are responsible for exploitation part of the search while the scout bees carry out the exploration part. The number of food sources is equal to the number of employed bees; in other words, for any food source there exists a unique employed bee. The number of onlooker bees is assumed to be the same as that of employed bees. An employed bee corresponding to an exhausted food source becomes a scout bee. A food source (i.e. a solution) is assumed to be exhausted if its quality is not improved after a certain number of cycles called *limit*. The employed and onlooker bees also do a local search in the neighborhood of their food sources and switch to them if they have more nectar (higher fitness). The detailed steps taken in a typical ABC algorithm are listed below:

- Initialize the population of the solutions x_i
- Evaluate the population using the fitness function
- While the maximum cycle number is not reached
- Produce new random solutions v_i in the neighborhood of the existing solutions (x_i) for the employed bees
- Replace x_i with the newly generated solution v_i if its fitness is higher than x_i
- Assign probabilities to the solutions x_i according to their finesses
- Assign a solution to each onlooker bee based on the probabilities of x_i's, and produce random solutions v_i for the onlookers in the neighborhood of x_i's
- Replace x_i in the memory of an onlooker bee with v_i, if its fitness is higher than x_i
- Determine an abandoned solution and replace it with a new randomly generated solution for the scout bee

– Memorize the best solution found so far
– End While

The probabilities using which the onlookers choose food sources x_i are calculated as

$$p_i = \frac{fit(x_i)}{\sum_{j=1}^{n} fit(x_j)} \tag{10}$$

where n is the size of the population and $fit(x_i)$ is the fitness of solution x_i.

Finding the optimal WIP level and job sequencing

To solve this problem using the ABC algorithm, we fix the WIP level in each run and then we use an ABC search to find the optimal job sequencing. Each solution x_i is a vector in which $x_i(j)$ represents the jth part type to be processed in the production line. Assuming the number of part types being n, the space of possible solutions is of size $n!$, which makes the search intractable using deterministic search methods for large values of n. The function $f(x_i)$ to be minimized is the makespan time corresponding to the job sequencing x_i. We assign the following fitness function $fit(x_i)$ to any solution x_i

$$fit(x_i) = \frac{1}{1 + f(x_i)} \tag{11}$$

This fitness function will be used in evaluating the population and the associated probabilities for every solution according to (10). A very important step in any ABC algorithm is how to choose a random solution v_i in the neighborhood of x_i. In the case where the search space for x_i is continuous, this is simply done by randomly choosing a solution x_k and letting v_i to be a randomly selected point on the segment between x_i and x_k. However, this method is not applicable to the discrete search spaces, which makes the problem more challenging. In this paper we employ the following algorithm to construct v_i from x_i

– $j = 1$
– While $(j \leq n)$
– Generate a random number $p \in [0, 1]$
– If $p > p_{th}$ or $x_i(j) \in \{v_i(1), \ldots, v_i(j-1)\}$, then choose $v_i(j)$ randomly from $\{1, \ldots, n\} \backslash \{v_i(1), \ldots, v_i(j-1)\}$, otherwise, set $v_i(j) = x_i(j)$
– $j = j + 1$
– End While

In the above algorithm, p_{th} is a threshold used to keep v_i close to x_i. It is clear that to keep v_i closer to x_i, p_{th} should be chosen close enough to 1.

We run this algorithm for every possible WIP level varying from 1 to the size of the demand list to find the optimal job sequencing in each case. Finally we choose the job sequencing with the minimum WIP level associated with the minimum makespan time.

Numerical examples

Example 1

In this example, we consider a single serial CONWIP production line with 3 machines, producing 12 part types. All 12 part types are processed by 3 machines sequentially, and it is assumed that all the assumptions mentioned in Section "Problem Formulation". are hold. Processing times on 3 machines for 12 products are generated randomly with uniform distribution over $\{1, \ldots, 30\}$ and are shown in Table 1. Sequence dependent set up times are also generated randomly with uniform distribution over $\{1, \ldots, 9\}$. These numbers are shown in Table 2, with T_{ij} denoting the set up time required to change from processing part type P_i to processing part type P_j (for instance, T_{12} in this production line is 4 time units). Finally, Table 3 shows the number of parts to be produced through the production line. We also choose the weight associated to the unbalanced workload $\beta = 1$.

Artificial bee colony

For any WIP level, the search space for finding the optimal job sequencing has 12! elements. It is well known that for any heuristic algorithm, the initial parameter setting highly affects the performance of the method. Therefore the control parameters of the ABC algorithm are chosen as follows. The colony size is set to be 50, which corresponds to 25 employed bees and also 25 onlookers, and is based on Karaboga and Basturk (2008)'s approach in order to reduce the number of control parameters. The maximum number of cycles is

Table 1 Part processing times

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Machine1	9	7	22	9	12	20	17	28	9	12	17	19
Machine2	11	5	4	16	4	8	6	14	4	8	6	7
Machine3	3	10	8	30	19	26	9	2	15	27	8	11

Table 2 Sequence dependent setup times

T_{ij}	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$	$j=11$	$j=12$
$i=1$	0	4	5	7	1	4	7	3	6	4	9	4
$i=2$	2	0	4	2	8	7	6	5	2	8	3	9
$i=3$	7	2	0	6	4	1	1	4	8	3	7	3
$i=4$	6	7	9	0	9	6	8	1	3	9	2	5
$i=5$	3	9	5	8	0	8	5	7	1	6	8	3
$i=6$	4	4	6	7	5	0	4	9	7	2	3	8
$i=7$	2	6	2	9	3	9	0	4	3	1	7	2
$i=8$	9	8	7	5	7	4	6	0	8	7	3	1
$i=9$	7	2	1	5	3	9	8	9	0	4	5	8
$i=10$	2	5	4	3	8	4	6	3	9	0	1	3
$i=11$	3	7	5	8	4	2	4	2	4	3	0	5
$i=12$	6	1	8	6	2	5	7	7	5	5	7	0

Table 3 Demand list

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Demand	5	7	4	6	9	3	7	5	9	3	5	7

Table 4 Makespan time and optimal job sequencing for various WIP levels

WIP level	Makespan	Optimal job sequencing
1	2509	$P_9P_{11}P_6P_{10}P_7P_1P_5P_{12}P_8P_4P_2P_3$
2	1386	$P_1P_5P_{12}P_3P_9P_2P_4P_8P_{11}P_{10}P_6P_7$
3	1265	$P_2P_1P_{11}P_4P_8P_{10}P_6P_3P_9P_{12}P_5P_7$
4	1189	$P_{10}P_{11}P_9P_7P_1P_5P_{12}P_2P_4P_8P_6P_3$
5	1107	$P_4P_8P_{11}P_6P_1P_5P_{12}P_9P_7P_2P_{10}P_3$
6	1063	$P_4P_8P_2P_1P_5P_{11}P_6P_{10}P_7P_3P_9P_{12}$
7	1047	$P_2P_6P_3P_9P_{11}P_4P_8P_5P_{12}P_1P_{10}P_7$
8	1045	$P_9P_5P_{12}P_{11}P_6P_4P_8P_1P_2P_{10}P_7P_3$
9	1043	$P_4P_8P_2P_{12}P_5P_1P_{11}P_{10}P_9P_7P_6P_3$
10	1040	$P_4P_8P_{10}P_1P_2P_9P_5P_{12}P_7P_{11}P_6P_3$
11	1040	$P_4P_8P_{10}P_1P_2P_9P_5P_{12}P_7P_{11}P_6P_3$
12	1040	$P_4P_8P_{10}P_1P_2P_9P_5P_{12}P_7P_{11}P_6P_3$
13	1040	$P_4P_8P_{10}P_1P_2P_9P_5P_{12}P_7P_{11}P_6P_3$

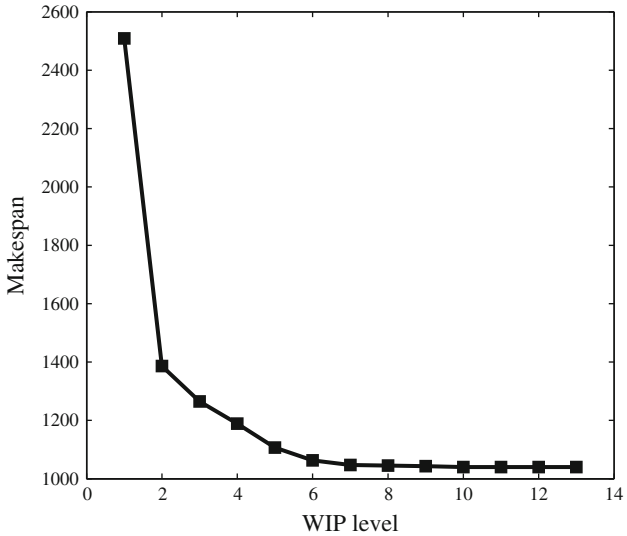

Fig. 2 CONWIP system performance for optimal job sequencing

1500 which can provide an acceptable convergence speed for search. Also the limit successive iteration value, as shown by Karaboga (2009), plays an important role on the performance of ABC heuristic for solving function optimization problems. We decide to adopt Szeto et al. (2011)'s approach and sets the limit value to be proportional to the number of part types produced. Additionally, the threshold probability is chosen to be $p_{th} = 0.8$.

Accordingly, the optimal job sequencing and makespan time for various WIP levels are shown in Table 4 and Fig.2. As can be seen, up to some point, any increase in the WIP level leads to a decrease in the makespan time and consequently alter the optimum job sequence order. However, a single specified point, increasing WIP level will not improve the performance of the system (WIP level 10 in this example). This is the best WIP level in the sense that its corresponding optimal job sequencing results in the minimum possible makespan time, and using a higher WIP level cannot improve the performance of the system.

Genetic algorithm

We also apply GA approach to solve this example. Briefly, each gene is defined as a job performed on each part and

Table 5 CPU time (seconds)

Number of Cards	ABC	GA
1	4	29
2	6	45
3	10	105
4	18	121
5	37	367
6	48	415
7	56	482
8	69	594
9	92	676
10	132	705

the corresponding chromosome is job sequence vector on the whole set of machines. The initial population is created using Random Permutation Job Sequence (RPJS) procedure and besides the *roulette wheel selection* is employed for electing among chromosomes at each iteration. Not surprisingly, the final solution of GA is identical to that from ABC (i.e. WIP level of 10 and minimum makespan time of 1040 units). However, as indicated in Table 5, the CPU time needed to perform GA procedure across different WIP levels is significantly higher than that resulted from ABC algorithm. So we can conclude that the algorithm proposed in this study extremely outperforms GA approach in term of time domain.

Table 6 Sequence dependent setup times

T_{ij}	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$	$j=11$
$i=1$	6	8	8	2	4	5	3	9	9	6	3
$i=2$	2	1	9	3	4	10	6	2	1	7	7
$i=3$	10	5	8	2	8	3	5	3	6	1	9
$i=4$	7	7	8	5	2	3	6	9	2	8	10
$i=5$	5	7	10	1	3	3	10	10	8	4	3
$i=6$	5	7	10	1	3	3	10	10	8	4	3
$i=7$	6	2	3	9	4	5	8	2	2	3	8
$i=8$	8	9	1	6	5	4	3	2	10	8	2
$i=9$	3	8	3	4	8	8	1	7	10	10	7
$i=10$	1	3	1	4	1	9	6	2	6	7	4
$i=11$	6	3	6	4	1	10	5	8	7	10	3
$i=12$	10	5	5	9	6	1	2	5	7	5	8
$i=13$	7	6	8	2	1	10	1	7	8	5	1
$i=14$	1	9	9	1	2	7	1	3	1	7	8
$i=15$	8	9	10	6	6	5	4	7	2	3	3
$i=16$	5	2	9	4	4	1	3	2	1	3	8
$i=17$	3	7	1	4	5	5	9	6	1	10	9
$i=18$	5	9	1	4	10	7	3	3	9	6	2
$i=19$	1	3	4	8	8	7	3	6	2	9	4
$i=20$	9	4	5	9	5	7	2	1	3	2	8
$i=21$	10	2	10	2	8	6	2	5	1	3	5

T_{ij}	$j=12$	$j=13$	$j=14$	$j=15$	$j=16$	$j=17$	$j=18$	$j=19$	$j=20$	$j=21$
$i=1$	9	1	4	6	10	3	8	6	9	4
$i=2$	2	4	7	7	9	1	10	9	9	8
$i=3$	8	10	3	3	7	8	1	4	10	3
$i=4$	10	9	8	1	10	3	2	4	6	1
$i=5$	3	8	6	7	6	5	7	1	8	2
$i=6$	3	8	6	7	6	5	7	1	8	2
$i=7$	3	4	10	10	10	8	4	6	8	10
$i=8$	2	4	4	5	1	1	2	6	4	5
$i=9$	7	9	3	7	3	8	8	7	4	1
$i=10$	3	10	9	6	4	9	10	2	4	1

Table 6 continued

T_{ij}	$j=12$	$j=13$	$j=14$	$j=15$	$j=16$	$j=17$	$j=18$	$j=19$	$j=20$	$j=21$
$i=11$	4	5	4	7	8	6	7	9	7	8
$i=12$	9	8	3	5	8	2	2	5	7	7
$i=13$	1	5	4	8	4	4	6	1	5	7
$i=14$	4	8	4	5	3	6	6	5	9	10
$i=15$	3	7	3	2	6	9	7	9	5	3
$i=16$	9	10	10	1	5	2	4	1	8	2
$i=17$	5	3	8	7	8	2	4	9	6	1
$i=18$	4	4	6	7	9	9	4	5	2	5
$i=19$	1	5	7	6	3	9	9	2	5	9
$i=20$	8	6	9	4	3	9	1	5	2	4
$i=21$	2	4	1	9	2	1	5	10	3	3

Table 7 Part processing times

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
Machine1	14	20	28	28	37	45	20	4	2	3	12
Machine2	4	13	37	8	32	25	44	4	37	32	37
Machine3	37	30	19	30	9	39	12	30	33	45	32
Machine4	38	29	8	20	30	16	44	27	5	34	20
Machine5	10	13	16	39	27	19	9	34	23	5	9
Machine6	1	10	19	33	13	23	14	39	22	33	2
Machine7	11	3	45	18	16	23	23	10	33	17	20
Machine8	10	20	14	31	20	37	29	19	39	1	4
Machine9	38	40	36	38	44	40	14	3	19	28	45
Machine10	39	4	12	21	29	34	45	28	33	14	29
Machine11	12	11	9	7	2	9	10	4	21	12	1
Machine12	12	1	5	12	10	28	31	12	16	21	29

	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}
Machine1	11	18	34	34	9	38	9	38	14	45
Machine2	28	38	28	31	31	11	39	44	26	8
Machine3	33	45	32	33	24	45	1	30	34	5
Machine4	34	42	29	15	12	10	16	41	8	34
Machine5	28	3	21	38	14	9	31	20	9	21
Machine6	2	30	18	26	8	19	26	15	33	16
Machine7	23	44	31	23	38	21	7	28	28	12
Machine8	12	7	34	3	36	21	4	26	10	45
Machine9	33	1	1	42	27	27	15	23	41	39
Machine10	10	20	31	32	2	14	2	36	14	41
Machine11	32	4	16	11	17	10	18	9	2	11
Machine12	34	39	11	17	12	19	18	10	36	19

Example 2

In order to demonstrate the applicability of our approach close to real world situations, we here examine a more elaborate case. Thus in the following example, there are 21 part types to be manufactured by a 12-machine serial production line under a single loop conwip protocol. Similarly the set up time, processing time, and demand

Table 8 Demand list

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Demand	10	5	2	6	7	12	7	9	2	12	17	8
	P_{13}	P_{13}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}	
Demand	19	2	6	6	15	1	3	8	18	1	3	

Table 9 Makespan time and optimal job sequencing for various WIP levels

WIP level	Makespan	Optimal job sequencing
1	9963	$P_{19}P_{14}P_8P_{21}P_5P_1P_{13}P_{11}P_7P_2P_9P_{20}P_{15}P_6P_3P_{10}P_{17}P_{12}P_{16}P_4P_{18}$
2	5816	$P_{19}P_{14}P_8P_{10}P_5P_1P_{20}P_{11}P_7P_2P_9P_{13}P_{15}P_6P_3P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
3	4954	$P_{19}P_{14}P_8P_{10}P_5P_1P_{20}P_{11}P_7P_2P_9P_{13}P_{15}P_6P_3P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
4	4379	$P_{19}P_{14}P_8P_{10}P_7P_1P_{13}P_{15}P_5P_2P_9P_{20}P_{11}P_6P_3P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
5	3954	$P_{19}P_{14}P_8P_{10}P_5P_1P_{13}P_{11}P_7P_2P_9P_{20}P_{15}P_6P_3P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
6	3723	$P_{19}P_{14}P_8P_{10}P_5P_1P_{13}P_{11}P_2P_7P_9P_{20}P_{15}P_3P_6P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
7	3511	$P_{19}P_{14}P_8P_{10}P_1P_5P_{13}P_{11}P_2P_7P_9P_{20}P_{15}P_3P_6P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
8	3451	$P_{19}P_{14}P_8P_{10}P_1P_5P_{13}P_{11}P_2P_7P_9P_{20}P_{15}P_3P_6P_{21}P_{17}P_{12}P_{16}P_4P_{18}$
9	3451	$P_{19}P_{14}P_8P_{10}P_1P_5P_{13}P_{11}P_2P_7P_9P_{20}P_{15}P_3P_6P_{21}P_{17}P_{12}P_{16}P_4P_{18}$

information are summarized in Tables 6, 7, and 8, respectively.

Artificial bee colony

Following same steps as previous part, at each WIP level, the optimal sequence path along with related makespan time are abstracted in Table 9. Thus one should utilize only 8 cards in this example to reach the best operating condition of the system with regards to makespan time. Also as expected, increasing WIP level will not change the optimum condition of the problem and hence we introduce this as the final reachable solution.

Genetic algorithm

At this point, we try to solve the example by Genetic Algorithm approach. Due to the fact that the dimension of the problem is grown here, we investigate several different combinations of crossover and mutation procedures but no feasible solution can be found after hours of computation. Thus, comparing to ABC algorithm, we infer that GA cannot be able to address this example.

Conclusion

This paper considers a CONWIP-based serial production line with a multi-product multi-machine environment in which all product types follow the same path through all machines. We assume deterministic processing and setup times without machine breakdown and preemption presumption. A Mixed Integer Non-Linear Programming (MINLP) approach is applied to model the optimization problem with the aim of minimizing the overall makespan time. Apart from simplifying and linearization techniques, an ABC heuristic is then proposed to find the optimal WIP level and job sequence order simultaneously. Two sets of numerical case studies are presented to show the viability of our proposed method and their results are also compared with the non-traditional Genetic Algorithm approach. It is shown that for both cases the ABC method greatly outperforms Genetic Algorithm in terms of run time. Namely in a small-sized problem, GA is able to converge to final solution but at a much longer time than ABC and for a large-sized dimension, Genetic Algorithm cannot reach to the solution after hours of calculation.

Briefly, the most important contribution made in this research is that our proposed modeling approach treats both WIP inventory level and job sequencing at the same time. In addition, our solution way does not use any simplifying assumptions or linearization routine, and instead explores a production line simulator developed on MATLAB for modeling the nonlinear complex dynamics of the system and calculating the correspondent fitness for candidate solutions. lastly, as illustrated by numerical simulations, we indicate that the proposed method can be applicable to industrial problems of a similar type but involving much more number of parts and machines.

At last, there are some directions which can be sought for future research. For example, one can consider different selections of machines for various types of products and develop card-based production systems in a job shop, flow shop, and assembly line settings. Alternatively, Multi-CONWIP control policy can be studied for such systems and compared with single-loop CONWIP protocol across various assumptions like machine failure, preemption and rework. Also effects of randomness in processing and setup times need to be examined for different problem dimensions.

Acknowledgments The authors are highly grateful to the anonymous reviewers for their insightful comments that have significantly improved the earlier version of this paper.

References

Akay, B., & Karaboga, D. (2010). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, Published online March 2010. doi:10.1007/s10845-010-0393-4.

Bonvik, A. M., Couch, C. E., & Gershwin, S. B. (1997). A comparison of production-line control mechanisms. *International Journal of Production Research, 25*(3), 789–804.

Cao, D., & Chen, M. (2005). A mixed integer programming model for a two line CONWIP-based production and assembly system. *International Journal of Production Economics, 95*(3), 317–326.

Chang, T. M., & Yih, Y. (1994). Generic Kanban systems for dynamic environments. *International Journal of Production Research, 32*, 889–902.

Framinan, J. M., Gonzleza, P. L., & Ruiz-Usano, R. (2003). The CONWIP production control system: Review and research issues. *Production Planning and Control, 14*, 255–265.

Framinan, J. M., Gonzleza, P. L., & Ruiz-Usano, R. (2006). Dynamic card controlling in a Conwip system. *International Journal of Production Economics, 99*(1/2), 102–116.

Garey, M. R., & Johnson, D. S. (1979). *Computers and interactability: A guide to the theory of NP-completeness*. San Francisco: Freeman.

Gaury, E. G. A., Pierreval, H., & Kleijnen, J. P. C. (2000). An evolutionary approach to select a pull system among Kanban, Conwip and Hybrid. *Journal of Intelligent Manufacturing, 11*, 157–167.

Golany, B., Dar-EL, E. M., & Zeev, N. (1999). Controlling shop floor operations in a multi-family, multi-cell manufacturing environment through constant work-in-process. *IIE Transactions, 31*, 771–781.

Goldratt, E. M., & Cox, J. (1985). *The goal*. Croton-on-Hudson, NY: North River Press.

Gstettner, S., & Kuhn, H. (1996). Analysis of production control systems Kanban and Conwip. *International Journal of Production Research, 34*(11), 3253–3274.

Herer, Y. T., & Masin, M. (1997). Mathematical programming formulation of CONWIP-based production lines; and relationships to MRP. *International Journal of Production Research, 35*(4), 1067–1076.

Hopp, W. J., & Spearman, M. L. (1991). Throughput of a constant work in process manufacturing line subject to failures. *International Journal of Production Research, 29*(3), 635–655.

Hopp, W. J., & Roof, M. L. (1998). Setting WIP levels with statistical throughput control (STC) in CONWIP production lines. *International Journal of Production Research, 36*(4), 867–882.

Ip, W. H., Huang, M., Yung, K. L., Wang, D., & Wang, X. (2007). CONWIP based control of a lamp assembly production line. *Journal of Intelligent Manufacturing, 18*(2), 261–271.

Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of the Franklin Institute, 346*, 328–348.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06 , Kayseri, Turkey: Erciyes University.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing, 8*, 687–697.

Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation, 214*(1), 108–132.

Khojasteh-Ghamari, Y. (2010). Developing a framework for performance analysis of a production process controlled by Kanban and CONWIP. *Journal of Intelligent Manufacturing*, Published online October 2009. doi:10.1007/s10845-009-0338-y.

Knolmayer, G., Mertens, P., & Zeier, A. (2002). *Supply chain management based on SAP systems*. Berlin: Springer.

Kumar, S. K., Tiwari, M. K., & Babiceanu, R. F. (2010). Minimisation of supply chain cost with embedded risk using computational intelligence approaches. *International Journal of Production Research, 48*(13), 3717–3739.

Lambrecht, M., & Segaert, A. (1990). Buffer stock allocation and assembly type production lines. *International Journal of Operations and Production Management, 10*(2), 47–61.

Li, N., Zhang, M. T., Deng, S., Lee, Z. H., Zhang, L., & Zheng, L. (2007). Single-station performance evaluation and improvement in semiconductor manufacturing: A graphical approach. *International Journal of Production Economics, 107*(2), 397–403.

Luh, P. B., Zhou, X., & Tomastik, R. N. (2000). An effective method to reduce inventory in job shops. *IEEE Transactions on Robotics and Automation, 16*, 420–424.

Marek, R. P., Elkins, D. A., & Smith, D. R. (2001). Understanding the fundamentals of Kanban and CONWIP pull systems using simulation. In *Proceedings of the 2001 winter simulation conference*, Vol. 2, pp. 921–929.

Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences, 181*(12), 2455–2468.

Sato, R., & Khojasteh-Ghamari, Y. (2010). An integrated framework for card-based production control systems. *Journal of Intelligent Manufacturing*, Published online June 2010. doi:10.1007/s10845-010-0421-4.

Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing, 9*(2), 625–631.

Spearman, M. L., Woodruff, D. L., & Hopp, W. J. (1989). A hierarchical control architecture for constant work-in-process (CONWIP) production systems. *Journal of Manufacturing and Operations Management, 2*, 147–171.

Spearman, M. L., Woodruff, D. L., & Hopp, W. J. (1990). Conwip: a pull alternative to Kanban. *International Journal of Production Research, 28*, 879–894.

Spearman, M. L., & Zazanis, M. A. (1992). Push and Pull Production Systems: Issues and Comparison. *Operations Research, 40*(3), 521–532.

Sundar, S., & Singh, A. (2010). A hybrid heuristic for the set covering problem. *Operational Research: An International Journal*, Published online September 2010. doi:10.1007/s12351-010-0086-y.

Szeto, W. Y., Wu, Y., & Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research, 215*(1), 126–135.

Yeh, W. C., & Hsieh, T. J. (2011). Solving reliability redundancy allocation problems using an artificial bee colony algorithm. *Computers and Operations Research, 38*(11), 1465–1473.

Zhang, W., & Chen, M. (2001). A mathematical programming model for production planning using CONWIP. *International Journal of Production Research, 39*(12), 2723–2734.

Zhang, C., Ouyang, D., & Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems with Applications, 37*, 4761–4767.