A Hybrid Discrete Bat Algorithm with Krill Herd based advanced planning and scheduling tool for the capital goods industry

Sirikarn Chansombat¹, Ponnapa Musikapun², Pupong Pongcharoen¹*, and Christian Hicks³

¹ Centre of Operations Research and Industrial Applications (CORIA), Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand.
Email: sirikarn53@email.nu.ac.th ORCID ID: 0000-0003-4873-0478
Email: pupongp@nu.ac.th and Scopus author ID: 17435722900

² School of Logistics and Supply Chain, Naresuan University, Phitsanulok 65000, Thailand
Email: ponnapam@nu.ac.th and Scopus author ID: 56644715500

³ Newcastle University Business School, 5 Barrack Road, Newcastle upon Tyne, NE1 7RU, UK.
Email: chris.hicks@ncl.ac.uk and Scopus author ID: 7102667331

* Corresponding author: pupongp@nu.ac.th

Abstract

Capital goods companies produce high value products such as power plant or ships, which have deep and complex product structures, with components having long process routings. Contracts usually include substantial penalties for late delivery. The high value of items can lead to substantial holding costs. Efficient schedules minimise earliness and tardiness costs and need to satisfy assembly and operation precedence constraints as well as finite capacity. This paper presents the first advanced planning and scheduling (APS) tool
for the capital goods industry that uses a Discrete Bat Algorithm (DBA), modified DBA (MDBA) and hybrid DBA with Krill Herd algorithm (HDBK) to optimise schedules. The tool was validated using four datasets obtained from a collaborating capital goods company. A sequential experimental strategy was adopted. The first experiment identified appropriate parameter settings for the DBA. The second experiment evaluated and compared the performance of the proposed HDBK algorithm with an Artificial Bee Colony, Krill Herd (KH), Modified KH, DBA and MDBA metaheuristics. The experimental results revealed that the HDBK performed best in terms of the minimum penalty cost for all problem sizes and achieved up to a 47.837% reduction in mean total penalty costs of extra-large problem size.

**Keywords:** Advanced planning and scheduling; Capital Goods; Bat Algorithm; Krill Herd; Artificial Bee Colony.

1 **Introduction**

Suppliers of capital goods are an important sector of the world economy that enhances the productivity and supports the diffusion of superior technologies (Fauceglia 2014). The main business activities of capital goods companies are the design, manufacture and construction of plant. Typical products include cranes, large steam turbines, offshore production facilities, oil platforms and ships. These products are important because they underpin manufacturing, services, trade and distribution (Acha et al. 2004).

Scheduling is “a decision-making process that plays an important role in most manufacturing and service industries” (Pinedo and Chao 1999, p.2). It can enhance the productivity of a production process (Gen and Lin 2014). Scheduling is one of the most popular research topics in the area of production and operations management (Chaudhry and Luo 2005). Production scheduling problems may be categorised as: single machine, parallel machines, flow shop, job shop, open shop and others (Pinedo and Chao 1999). Most production scheduling research has focused on single machine, parallel machines or flow shops (Lei 2009). Most of the production scheduling literature is theoretical and does not
model the many of the complexities experienced in practice (Fuchigami and Rangel 2017). There is a limited literature that has taken into account multiple-level assembly relationships (Na and Park 2014).

In the capital goods industry, production scheduling is a complex combinatorial optimisation (CO) problem. This is because there are a large number of components and subassemblies and the product structures are usually deep and complex. Major subassemblies require a range of components which are produced using a mix of jobbing, batch, assembly and flow processes. Many components require numerous machining operations which take place on many types of machine (Hicks 1998). Production scheduling must take into account operation and assembly precedence relationships and finite capacity (Hicks and Braiden 2000; Hicks 1998). Effective production schedules minimise production lead-time and meet customer due dates whilst satisfying resource constraints (Chen, Ji, and Wang 2011; Dayou, Pu, and Ji 2009). Production scheduling problems are non-deterministic polynomial (NP) hard combinatorial optimisation problems which means that the amount of computation required increases exponentially with problem size (Blum and Roli 2003).

Metaheuristics are particularly suitable for solving very large combinatorial problems, however, it is impossible to search the whole solution space, therefore an optimal solution cannot be guaranteed (Nagar, Haddock, and Heragu 1995). Metaheuristic algorithms may be classified in alternative ways (Talbi 2009; Yang 2010a). Single-point algorithms are trajectory methods that use local search heuristics e.g. Tabu Search (TS) (Glover 1990), Simulated Annealing (SA) (Kirkpatrick, Gelatt, and Vecchi 1983), Multi-start local search (MS), Greedy Randomised Adaptive Search Procedure (GRASP) and Iterated Local Search (ILS) (Lourenco, Martin, and Stützle 2003). They intensify search in the local region, which is also called exploitation oriented search (Gen and Cheng 1997). Population-based algorithms produce multiple solutions that explore the whole search space to produce greater
diversity. Well-known algorithms include Genetic Algorithms (GA) (Goldberg 1989), Ant Colony Optimisation (ACO) (Dorigo 1992) and Particle Swarm optimisation (PSO) (Kennedy and Eberhart 1995).

The literature reports the application of many established nature-inspired optimisation algorithms (see Table 1), which are broadly classified into five categories (Nanda and Panda 2014; Gupta and Sharma 2016; Fister Jr et al. (2013); Zambonelli and Virolı 2011): evolutionary-based, physics and chemistry based, swarm intelligence based, bio-inspired based and other algorithms.

Table 1. Classification of nature-inspired metaheuristic algorithms.

<table>
<thead>
<tr>
<th>Types</th>
<th>Algorithms</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolutionary based</td>
<td>Genetic Algorithm (GA)</td>
<td>Holland (1975)</td>
</tr>
<tr>
<td></td>
<td>Genetic programming (GP)</td>
<td>Koza (1990)</td>
</tr>
<tr>
<td></td>
<td>Evolutionary Strategy (ES)</td>
<td>Rechenberg (1965)</td>
</tr>
<tr>
<td></td>
<td>Evolutionary Programming (EP)</td>
<td>Fogel, Owens, and Walsh (1966)</td>
</tr>
<tr>
<td>Physics and Chemistry</td>
<td>Simulated Annealing (SA)</td>
<td>Kirkpatrick et al. (1983)</td>
</tr>
<tr>
<td>based</td>
<td>Memetic Algorithm (MA)</td>
<td>Moscato and Norman (1992)</td>
</tr>
<tr>
<td></td>
<td>Harmony Search (HS)</td>
<td>Geem et al. (2001)</td>
</tr>
<tr>
<td></td>
<td>Shuffled Frog-Leaping Algorithm</td>
<td>Eusuff, Lansey, and Pasha (2006)</td>
</tr>
<tr>
<td>Swarm Intelligence</td>
<td>Ant Colony Optimisation (ACO)</td>
<td>Dorigo (1992)</td>
</tr>
<tr>
<td>based</td>
<td>Particle Swarm Optimisation (PSO)</td>
<td>Kennedy and Eberhart (1995)</td>
</tr>
<tr>
<td></td>
<td>Artificial Bee Colony (ABC)</td>
<td>Karaboga (2005)</td>
</tr>
<tr>
<td></td>
<td>Firefly Algorithm (FA)</td>
<td>Yang (2010a)</td>
</tr>
<tr>
<td></td>
<td>Bat Algorithm (BA)</td>
<td>Yang (2010a)</td>
</tr>
<tr>
<td></td>
<td>Krill Herd (KH)</td>
<td>Gandomi and Alavi (2012)</td>
</tr>
<tr>
<td></td>
<td>Earthworm Optimisation Algorithm (EOA)</td>
<td>Wang, Deb, and Coelho (2016)</td>
</tr>
<tr>
<td></td>
<td>Moth search algorithm (MSA)</td>
<td>Wang (2016)</td>
</tr>
<tr>
<td></td>
<td>Cuckoo Search (CS)</td>
<td>Yang and Deb (2009)</td>
</tr>
<tr>
<td></td>
<td>Chaotic Cuckoo Search (CCS)</td>
<td>Wang et al. (2016)</td>
</tr>
<tr>
<td></td>
<td>Elephant Herding Optimisation (EHO)</td>
<td>Wang et al. (2016)</td>
</tr>
<tr>
<td>Bio-inspired based</td>
<td>Flower Algorithm (FA)</td>
<td>Yang (2012)</td>
</tr>
<tr>
<td></td>
<td>Japanese Tree Frogs Calling</td>
<td>Hernandez and Blum (2012)</td>
</tr>
<tr>
<td></td>
<td>Atmosphere Clouds Model</td>
<td>Yan, Hao, and Xie (2013)</td>
</tr>
<tr>
<td>Other algorithms</td>
<td>Backtracking Optimisation Search (BOS)</td>
<td>Civicioglu (2013b)</td>
</tr>
<tr>
<td></td>
<td>League Championship Algorithm (LCA)</td>
<td>Kashan (2009)</td>
</tr>
<tr>
<td></td>
<td>Social Emotional Optimisation (SEO)</td>
<td>Xu, Cui, and Zeng (2010)</td>
</tr>
<tr>
<td></td>
<td>Artificial Cooperative Search (ACS)</td>
<td>Civicioglu (2013a)</td>
</tr>
</tbody>
</table>

Advanced planning and scheduling (APS) systems are based on optimisation and constraint-based planning algorithms that aim to meet customer requirements whilst...
satisfying specified constraints (Hvolby and Steger-Jensen 2010). APS systems aim to manage the supply chain to improve customer satisfaction, increase efficiency and reduce costs (Dayou, Pu, and Ji 2009). APS systems have been based upon GA (Chen, Ji, and Wang 2011) and GA with local search (Pu et al. 2007), but there are no reports of the BA being used to for APS.

The objectives of this paper were to: (i) review Swarm Intelligence based Algorithms including, the Artificial Bee Colony (ABC), Krill Herd (KH) and Bat Algorithms (BA); (ii) explain a novel APS scheduling tool that meets the requirements of the capital goods industry that manufacture complex products with multi-level assemblies. The tool incorporates a novel Discrete Bat Algorithm (DBA), a Modified Discrete Bat Algorithm (MDBA) and a Hybrid Discrete Bat Algorithm with Krill Herd algorithm (HDBK) for optimisation; (iii) conduct a series of computational experiments that identified appropriate parameter settings for the DBA; iv) outline the development of the MDBA and the HDBK; and v) compare the performance of the proposed methods (DBA, MDBA and HDBK) with other approaches (ABC, KH and modified KH).

The next section explains the characteristics of population-based metaheuristics. Section 3 describes the development of advanced planning and scheduling (APS) tool. Section 4 presents the experimental design and analyses the results. The last section provides conclusion and suggestions for future research.

2 Population-based metaheuristics

Population-based algorithms have been widely used to solve real world problems. They simultaneously consider multiple potential solutions and tend to perform better than single-point algorithms (Manda, Satapathy, and Poornasatyanarayana 2012). Prugel-Bennett (2010) identified five mechanisms that help give population-based algorithms an advantage:
(i) building blocks from different solutions are combined; (ii) the crossover operator focuses the search and can dramatically reduce the time taken to find a solution; iii) the population acts as a low pass filter, which ignores local distractions; iv) a population has the ability to search different parts of the search space simultaneously, which hedges against bad luck in the initial position; and v) it is possible to identify parameter values that make an appropriate balance between exploitation and exploration.

Pongcharoen (2001) developed a comprehensive Genetic Algorithm (GA) tool for scheduling the production of capital goods using the objective function shown equation (1) that aggregates earliness and tardiness costs (Pongcharoen, Hicks, and Braiden 2004). This objective function was also used to solve the same problems using the Artificial Bee Colony (Pongcharoen et al. 2012) and the Krill Herd (Puongyeam, Pongcharoen, and Vitayasak 2014) algorithms.

Total penalty cost = \[ \sum_{j=1}^{C} \sum_{k=1}^{P} Pe(E_{jk}) + \sum_{k=1}^{P} Pe(E_k) + \sum_{k=1}^{P} Pt(T_k) \]  \hspace{1cm} (1)

Notation:
- \( j \) assembly or component \( j (j = 1, 2, ..., C_{max}) \)
- \( k \) final product \( k (k = 1, 2, ..., P_{max}) \)
- \( Pe \) earliness penalty rate (currency units per day)
- \( Pt \) tardiness penalty rate (currency units per day)
- \( E_k \) earliness of product \( k \)
- \( E_{jk} \) earliness of component \( j \) in product \( k \)
- \( T_k \) tardiness of product \( k \)
2.1 Artificial Bee Colony (ABC) algorithm

The Artificial Bee Colony (ABC) algorithm is a popular swarm intelligence-based algorithm developed by Karaboga (Karaboga 2005; Karaboga and Akay 2009; Karaboga et al. 2014). This approach is based on the collective foraging behaviour of a bee colony, which includes three categories of bee: employed bees, which forage for nectar; onlookers waiting in the hive; and scouts, which undertake random search. There is only one bee that visits each source, so the number of employed bees is the same as the number of food sources. Once a food source is identified (a candidate solution), the nectar (fitness) is identified and computed. The scouts share information with the onlooker bees. Onlooker bees choose their food source depending on the probability of the food occurring. If bees are unable to improve the fitness of the food source, their solutions are rejected (see Karaboga and Basturk 2007).

De Oliveira and Schirru (2011) developed an ABC for combinatorial optimisation that used random keys (Bean 1994) for mapping discrete variables to continuous variables. Cui and Gu (2012) developed a discrete ABC for hybrid flow shop scheduling that included a three-step differential evolution scheme (mutation, crossover and selection) for allocating employed bees to food sources. They used the algorithm developed by Nawaz, Enscore, and Ham (1983) in their procedure. Pansuwan, Rukwong, and Pongcharoen (2010) developed a scheduling tool for capital goods companies that used a discrete ABC together with the objective function shown in equation (1).

2.2 Krill Herd (KH) algorithm

The Krill Herd (KH) algorithm (Gandomi and Alavi 2012) is a swarm intelligence algorithm which is based on the herding of the krill swarms. The time-dependent position of an individual krill is determined by three main actions: (i) movement induced other krill; (ii) foraging action; and (iii) random diffusion.
Initially, a swarm of krill are randomly generated in the search space. Krill try to maintain a high density and move according to their mutual effects (Gandomi and Alavi 2012). Each krill moves through \( n \) dimensional search space to look for a potential solution by moving towards the highest density of food. To improve the performance and convergence speed, crossover and mutation genetic operations were incorporated into the algorithm. The iterative search is ended when the termination criteria are met.

Wang, Deb, and Thampi (2015) developed a discrete krill herd method for flexible job shop scheduling. Puongyeam, Pongcharoen, and Vitayasak (2014) developed a discrete krill herd for scheduling in the capital goods industry which used the objective function shown in equation (1).

### 2.3 Bat Algorithm (BA)

In 2010, Yang (2010b) presented a new metaheuristic algorithm, called the Bat Algorithm (BA) which is based on the echolocation capability of the micro-bats. In nature bats fly randomly in their search for prey with velocities \( v_i \) at positions \( x_i \) with varying wavelength/frequency (\( \lambda/f \)), pulse rate \( r_i \) and loudness \( A_0 \). The position of each bat represents a possible solution. Depending on the proximity of the prey, bats can automatically adjust their wavelength/frequency and pulse emission rate \( r_i \in [0,1] \). The loudness can vary from a minimum loudness \( (A_{min}) \) to a maximum loudness \( (A_0) \) with a typical range of \([1,2]\). Frequencies are normally in the range 25kHz to 150kHz (Yang 2010b).

Yang (2010b) outlined the BA as follows. The process starts by initialising a swarm (population) of \( n \) virtual bats, each of which has a random initial position (initial solution), where the ranges are problem specific, together with random values for pulse rate, loudness and frequency. Then, all of the bats move from their initial positions seeking a global best solution. Each individual bat randomly selects a frequency \( (f_i) \) where \( f \in [0,f_{max}] \), using
equation (2), where $\beta \in [0,1]$ is a random number drawn from a uniform distribution. The velocity of each bat $i$ is updated using equation (3), where $t$ is the iteration number, $t_{\text{max}}$ is the maximum number of iterations ($0 \leq t \leq t_{\text{max}}$), $v_{i}^{t-1}$ is the previous velocity, $x_{i}^{t}$ is the current position of bat $i$ in iteration $t$, and $x^*$ is the position of the best-so-far bat. Equation (4) calculates the new position based upon the previous position and current velocity. For local search, once a solution is selected among the current best solutions, a new solution $x_{\text{new}}$ for each bat is generated locally using random walk using equation (5), where $x_{\text{old}} = x_{i}^{t}$, $\varepsilon \in [-1,1]$ is a random number, $A'$ is the arithmetic mean loudness of all bats in the current iteration. Equation (6) updates the loudness $A_{i}^{t+1}$ for each individual bat, where $\alpha$ is the loudness coefficient, a parameter in the range $[0,1]$. The pulse emission rate $r_{i}^{t+1}$ is updated by equation (7), where $r_{i}^{0}$ is the initial pulse emission rate for the bat and $\gamma$ the pulse rate emission coefficient is a parameter that is greater than 0. This process is repeated until the maximum number of iterations $t_{\text{max}}$ has been completed.

$$f_{i} = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}})\beta$$  \hspace{1cm} (2)  

$$v_{i}^{t} = v_{i}^{t-1} + (x_{i}^{t} - x^*)f_{i}$$  \hspace{1cm} (3)  

$$x_{i}^{t} = x_{i}^{t-1} + v_{i}^{t}$$  \hspace{1cm} (4)  

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A_{i}^{t}$$  \hspace{1cm} (5)  

$$A_{i}^{t+1} = \alpha A_{i}^{t}$$  \hspace{1cm} (6)  

$$r_{i}^{t+1} = r_{i}^{0}[1 - \exp(-\gamma t)]$$  \hspace{1cm} (7)  

The Bat Algorithm (BA) is a continuous optimisation algorithm, whereas combinatorial optimisation problems require discrete optimisation. There are two ways to apply the BA to discrete problems: i) use continuous optimisation, if it is possible to map the problem to a continuous variable; or ii) develop a discrete BA (Luo et al. 2014). Marichelvam and Prabaharam (2012) used the mapping approach to solve flow shop scheduling problems.
Luo et al. (2014) used the BA for solving permutation flow shop scheduling problems. Random keys (Bean 1994) were used to map from discrete to continuous variables. This was an approach that had previously been adopted by Tasgetiren et al. (2007) for use in Particle Swarm Optimisation. Dao, Pan, and Pan (2018) developed a parallel BA that used random key mapping for job shop scheduling.

2.4 A comparison of the proposed population-based metaheuristics

The concept, terminology and parameters of metaheuristics vary. Table 2 provides a summary of the population-based metaheuristics (ABC, KH and BA) presented in this work.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Artificial Bee Colony (ABC)</th>
<th>Krill Herd (KH)</th>
<th>Bat Algorithm (BA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural inspiration</td>
<td>Foraging behaviour of a bee colony</td>
<td>Herding behaviour of krill</td>
<td>Echolocation behaviour of micro-bats</td>
</tr>
<tr>
<td>Solution initialisation</td>
<td>Random</td>
<td>Random</td>
<td>Random</td>
</tr>
<tr>
<td>Candidate solution</td>
<td>Food source individual’s position</td>
<td>Krill individual’s position</td>
<td>Bat individual’s position</td>
</tr>
<tr>
<td>Old solution</td>
<td>Old food source position</td>
<td>Old krill position</td>
<td>Old bat position</td>
</tr>
<tr>
<td>New solution</td>
<td>New food source position</td>
<td>New krill position</td>
<td>New bat position</td>
</tr>
<tr>
<td>Best solution</td>
<td>Any food source with the best fitness</td>
<td>Any krill with the best fitness</td>
<td>Any bat with the best fitness</td>
</tr>
<tr>
<td>Fitness/objective</td>
<td>Nectar amount of the food source</td>
<td>Distance between krill individual and food and the densest location in the herd</td>
<td>Distance between bat individual and target</td>
</tr>
<tr>
<td>Size of candidates</td>
<td>Colony</td>
<td>Herd</td>
<td>Population</td>
</tr>
<tr>
<td>Iterative search</td>
<td>Number of cycles</td>
<td>Number of generations</td>
<td>Number of iterations</td>
</tr>
<tr>
<td>Process for generating new solution</td>
<td>The employed bee becomes a scout.</td>
<td>Motion induced by krill herd, foraging activity and physical diffusion</td>
<td>Adjusting iterations, update velocity and position</td>
</tr>
<tr>
<td>Intensification</td>
<td>Neighbourhood search carried by employed and onlooker bees</td>
<td>Foraging motion</td>
<td>Random walk</td>
</tr>
<tr>
<td>Diversification Parameters</td>
<td>Random search of scout bees - Combination of the population size and the number of maximum cycles ($nMCN$)</td>
<td>Random diffusion - Combination of population size and the number of max generations ($nI_{max}$)</td>
<td>Flying randomly - Combination of population size and the number of maximum iterations ($nI_{max}$)</td>
</tr>
<tr>
<td></td>
<td>- Limit factor which is a predefined value that limits the number of times that a food source can be moved without producing an improvement before it is abandoned. (percentage of the maximum number of cycles)</td>
<td>- Inertia weight of motion induced ($\omega_k$) - Inertia weight of the foraging motion ($\omega_{k_{mf}}$) - The maximum diffusion speed ($D_{mf}$) - The crossover operation ($COP$) - The mutation rate ($M_{k}$)</td>
<td>- Pulse rate emission coefficient ($\gamma$) - Loudness coefficient ($\alpha$)</td>
</tr>
</tbody>
</table>

Table 2. Concept and terminology comparison of ABC, KH and BA.
Due to the different inspirations adopted within the metaheuristics, the unique mechanisms embedded in the metaheuristics have their own properties to avoid iterative search becoming trapped in local optima whilst performing the search in a more intelligently than random search. The advantages and disadvantages of the proposed of the classical algorithms, including ABC, KH, and BA are summarised in Table 3.

Table 3. Advantages and disadvantages of ABC, KH and BA.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| ABC     | - Not sensitive to initial parameter values (Bansal, Sharma, and Jadon 2013).  
         | - Not affected by the number of dimensions of the problem (Bansal, Sharma, and Jadon 2013).  
         | - Can avoid local minimum (Karaboga and Basturk 2007).  
         | - Efficient for multivariable, multimodal function optimisation (Karaboga and Basturk 2007).  
         | - Good exploration (Khorsandi, Hosseinian, and Ghazanfari 2013; Gao and Liu 2012).  
         | - Quick convergence (Cui and Gu 2015).  
         | - Few control parameters (Cui and Gu 2015; Luo, Wang, and Xiao 2013).  
         | - Premature convergence (Bansal, Sharma, and Jadon 2013).  
         | - Long execution times because of its stochastic nature (Kang, Li, and Li 2013).  
         | - Poor exploitation (Khorsandi, Hosseinian, and Ghazanfari 2013; Gao and Liu 2012).  
         | - Slow to converge (Luo, Wang, and Xiao 2013).  
         | - Can easily fall into the local optimum (Luo, Wang, and Xiao 2013).  
         | - Hard to find the best solution from all feasible solutions (Luo, Wang, and Xiao 2013).  
         | - Poor exploitation (Wang et al. 2013).  |
| KH      | - Powerful exploration (Wang et al. 2013).  
         | - Each agent can contribute to the search process according to its fitness (Gandomi and Alavi 2012).  
         | - Each neighbour has an attractive/repulsive effect on the movement of the krill individual (Gandomi and Alavi 2012).  
         | - Very few control variables (Mukherjee and Mukherjee 2016; Wang et al. 2013).  
         | - Good balance between global and local search (Agrawal, Pandit, and Dubey 2016).  
         | - Few parameters to regulate (Wang, Hossein Gandomi, and Hossein Alavi 2013).  
         | - Able to shrink the search region towards the promising area within a few generations (Wang et al. 2013).  
         | - Easy to fall into the local optimum (Gandomi and Alavi 2012; Wang et al. 2013).  
         | - No guarantee of fast convergence (Wang et al. 2013).  
         | - Poor exploitation (Wang et al. 2013).  |
| BA      | - Powerful exploitation (Yilmaz and Kucuksille 2013; Dos Santos Coelho and Askarzadeh 2016).  
         | - Parameter control (automatically switching from exploration to exploitation) (Kaur and Chhabra 2016; Yang 2013).  
         | - Frequency tuning (Kaur and Chhabra 2016; Yang 2013).  
         | - Automatic zooming (Yang 2013).  
         | - Quick convergence at the initial stage by switching from exploration to exploitation (Yang 2013).  
         | - Balance between exploration and exploitation (Chua et al. 2015).  
         | - Can easily to fall into the local optimum (Li and Zhou 2014; Pravesjit 2016).  
         | - Premature convergence (Ahmadi and Nikravesh 2016).  
         | - May be trapped in local optima (Dos Santos Coelho and Askarzadeh 2016).  
         | - May lead to stagnation after the initial stage (Yang 2013).  
         | - Obtains poor results when dealing with high-dimensional problems (Fister Jr, Fister, and Yang 2013).  |
Table 4 summarises a comprehensive literature review of previous research that has used the proposed population-based metaheuristics (including ABC, KH and BA) for solving production scheduling problems. The hybridisation of the KH algorithm with other metaheuristics for solving the production scheduling problem is a gap in the literature.

Table 4. Applications of metaheuristics to solve production scheduling problems.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Single metaheuristics</th>
<th>Hybridisation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2013: Chansombat et al. (2013); Marichelvam et al. (2013); Xie, Zhou, and Tang (2013)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2014: Luo et al. (2014)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2016: Kongkaew (2016)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2017: Xu, Bao, and Zhang (2017); Zaher, Ragaa, and Sayed (2017)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2018: Dao, Pan, and Pan (2018)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2011: Li, Pan, and Gao (2011); Pan et al. (2011); Tasgetiren et al. (2011)</td>
<td>2013: Liu and Liu (2013); Han et al. (2013); Lin, Ying, and Huang (2013); Thammano and Phu-Ang (2013); Zhang, Song, and Wu (2013)</td>
</tr>
<tr>
<td></td>
<td>2013: Han et al. (2013); Lei (2013); Pan et al. (2013); Tasgetiren et al. (2013); Wang, Xie, and Cheng (2013)</td>
<td>2015: Li and Pan (2015); Nasiri (2015)</td>
</tr>
<tr>
<td></td>
<td>2014: Vijaychakararvathy, Marimuthu, and Sait (2014); Kizilay et al. (2014); Li, Pan, and Tasgetiren (2014); Muthiah and Rajkumar (2014); Pan et al. (2014)</td>
<td>2016: Yue et al. (2016)</td>
</tr>
<tr>
<td></td>
<td>2015: Ribas, Companys, and Tort-Martorell (2015); Al-Salamah (2015); Caniyilmaz, Benli, and Ilkay (2015); Cui and Gu (2015); Gao et al. (2015)</td>
<td>2017: Li et al. (2017); Muthulakshmi and Somasundaram (2017); Sundar et al. (2017); Wang et al. (2017)</td>
</tr>
<tr>
<td></td>
<td>2016: Asadzadeh (2016); Gao et al. (2016)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2017: Zhang et al. (2017); Li (2017); Pan et al. (2017)</td>
<td></td>
</tr>
<tr>
<td>KH</td>
<td>2014: Puongyeam, Pongcharoen, and Vitayasak (2014)</td>
<td>No reported research</td>
</tr>
</tbody>
</table>

3 The development of Advanced Production and Scheduling (APS) tool

The APS tool was developed for solving production scheduling problems in the capital goods company using a Discrete Bat Algorithm (DBA), a Modified DBA (MDBA), and a Hybrid Discrete Bat Algorithm with Krill Herd (HDBK) algorithm. The objective was
to find an optimal schedule which minimised the total cost of earliness and tardiness penalties (equation 1). The tool was coded in a modular style using the C sharp programming language. The APS tool starts by obtaining input data. The input data comprises: (a) order information - due dates, the number of products for the penalty cost coefficients for earliness and tardiness; (b) product information - product structure (including all assemblies, subassemblies and components); (c) operational information - process routings, set-up, machining and transfer times (for all assemblies, subassemblies and components); (d) resource information - list of machines and their availability; (e) the DBA’s parameters - the size of the population \( n \), the number of iterations \( I \), the pulse rate emission coefficient \( \gamma \), the loudness coefficient \( \alpha \) and the repositioning operation either using the swapping operator (Wang et al. 2003) or the adjustment operator (Wang et al. 2005).

A flowchart representing the proposed DBA, MDBA and HDBK used in the APS tool is shown in Figure 1, which includes:

i) The main menu of the APS tool is displayed when the mouse is double clicked. The problem dataset can be selected and uploaded into the APS tool. All operations are encoded into alphanumeric strings that represent sequences of operations. These are analogous to the discrete vector position of a bat with the number of dimensions equal to the total number of operations in the schedule;

ii) The graphical user interface (GUI) allows users to define parameters (\( pop, iteration, \gamma, \alpha, \) and repositioning operator), scheduling characteristics (\( Pe, Pt, \) and working hours per day), and the random seed number (if needed);
Start
Display: Main Menu
Upload problem dataset
Set the user-defined parameters: pop, iteration, etc. and work hours per day
Initialise the bat population \( x_i (i = 1, 2, ..., n) \)
Repair process
Feasible Solution?
Yes
Initialise the velocity \( v_i \), the pulse rate \( r_i \), the loudness \( A_i \), and the frequency \( f_i \) at position \( x_i \)
Evaluate the fitness \( f(x_i) \) using equation (1)
Store the best-so-far position \( (x^*) \)
\( t = 0 \) and \( i = 0 \)
No
Adjustment operator?
Yes
Swapping operator
No
Adjustment operator
 feas viable Solution?
Yes
Repair process
Feasible Solution?
Yes
Random number between 0 and 1
Select a solution among the best solutions randomly
MDBA?
Yes
Generate fifty local solutions
MDBA?
No
Generate a local solution
Randomly generate a new solution
New solution \( \Rightarrow D_{max} d \times \text{total number of swaps/adjustments} \)
HDBK?
Yes
Random number between 0 and 1
Swapping operator
Adjustment operator
No
Feasible Solution?
Yes
Repair process
Feasible Solution?
Yes
Random number between 0 and 1
Accept the new solution
Yes
Keep the old location as new location
Yes
i < n
No
Rank the bats and obtain the save best-so-far position
\( i < \text{iteration} \)
No
Output the fitness value of the best solution
Display: the best solution and Gantt chart
End

Figure 1. Flowchart of the DBA, MDBA and HDBK APS tool.
iii) A swarm of bats is randomly generated. The product structure representation is illustrated in Figure 2 (a) using a simple example. The root node represents the final product ($F_1$), which comprises assemblies ($A_1$ and $A_2$); subassemblies ($S_1$, $S_2$, and $S_3$); and components ($C_1$, $C_2$, $C_3$, and $C_4$) as the leaf nodes. All the nodes in the product structure will have a sequence of machining operations $O_1$, $O_2$...$O_n$, which need to be completed sequentially. If the component $C_1$ has three operations $O_1$, $O_2$, and $O_3$, $C_1$ can be represented as three intermediate items $C_1O_1$, $C_1O_2$, and $C_1O_3$ where $C_1O_3$ is the completed $C_1$, since it has three operations. Each bat represents a candidate solution (see Figure 2 (b));

iv) Candidate solutions may be infeasible because they contravene assembly or operation precedence constraints. A repair process (Pongcharoen, Hicks, and Braiden 2004) was adopted to change routings and/or assembly sequences (the position vectors) to ensure that all precedence constraints are satisfied. The repair process also takes into account timing, finite capacity and deadlock. Figure 3 illustrates the adjustment of an infeasible schedule (repair process). In bat 2, the intermediate item $C_1O_3$ is sequenced to take place before the intermediate item $S_1$. Therefore the algorithm swaps these operations so that they are in the correct sequence;

v) Initially, each bat is randomly assigned the velocity $v_i$, the pulse rate $r_i$ in the range $[0,1]$, the frequency $f_i$ in the range $[0,1]$, and the loudness $A_i$ in the range $[1,2]$, settings adopted (Chansombat et al. 2013);

vi) The total penalty costs for all of the individuals within the initial population are calculated using equation (1);

vii) The best-so-far position $x^*$ leading to the lowest penalty cost is identified;
viii) All bats move from their current location $x_i^t$ to a new location $x_i^{t+1}$. Instead of applying equation (3), which would apply for continuous optimisation, the discrete algorithm is based upon either the swapping operator (Wang et al. 2003) or the adjustment operator (Wang et al. 2005) which are described in steps ix) and x) below;
ix) Swapping operator - the first stage is to calculate how many swaps would be required to map $x_i^{t+1}$ to $x^*$ (the best-so-far solution) using the six steps illustrated in Figure 4(a). The first step compares the elements of $x^*$ from left to right with $x_i^{t+1}$. When a difference is detected (in this case $C_2O_2 / C_3O_1$), the second step swaps the current element in $x_i^{t+1}$ with the element containing the same value as $x^*$ ($C_2O_2$) and step 3 does the reverse swap to produce $x_i^{t+1'}$ (where the number of dashes “-” indicates the number of swaps). This process then continues until the number of swaps required to map $x_i^{t+1}$ to $x^*$ is determined. In this case, steps 4, 5 and 6 complete the process as $x_i^{t+1''}$ is the same as $x^*$ so the total number of swaps is 2; The second stage is to multiply this number of swaps by the random number $f_i$ determined by step (v) above. The value is then rounded up to determine the actual number of swaps to be performed on $x_i^{t+1}$ to determine its new position. This process is illustrated in Figure 4(b) that shows a situation where two swaps were required to transform, $x_i^{t+1}$ to $x^*$, the $f$ value was 0.1, giving 0.2 swaps, which would round up to one swap, so $x_i^{t+1}$ is changed to $x_i^{t+1'}$.

x) Adjustment operator - the first step is to calculate how many adjustments would be required to map $x_i^{t+1}$ to $x^*$. This is illustrated in Figure 5. The procedure compares the elements of $x^*$ with $x_i^{t+1}$. When a difference is detected the current element in $x^*$ is inserted into $x_i^{t+1}$, the duplicate value in $x_i^{t+1}$ is deleted as shown. The remaining values to the right of the insertion point are moved one position to the right. This process is then continued until the number of adjustments required to map $x_i^{t+1}$ to $x^*$ is determined. Again, this value is multiplied by $f$ and rounded up to determine the number of adjustments to be made on $x_i^{t+1}$ to determine its new position;
xi) The new positions of the bats are checked and repaired as necessary;

xii) The total penalty costs for all of the individuals within the initial population is calculated and the best-so-far value is identified;

xiii) A random number (rand) in the range 0-1 is generated;

xiv) rand is compared with the pulse rate (r);

xv) If rand > r, the best-so-far solution $x^*$ is taken as the start point for a local search.
If the MDBA is selected, this step is repeated 50 times to improve the exploitation capability, otherwise just once;

to improve the exploration capability of the BA, the random diffusion of the KH was incorporated into the conventional BA. This is illustrated in Figure 6.

A new solution is repaired if necessary;

The fitness of new solution is evaluated;

A new rand in the range 0-1 is generated. If rand < A_i and if the penalty cost of x_i < x* then x_i becomes x*, A_i will decrease using equation (6) and r_i will increase following equation (7). Otherwise, x* remains unchanged;

If the specified number of bats has not been completed, the procedure returns to step xiii). Otherwise, all bats are ranked and the best-so-far position is saved;

If the required number of iterations has not been completed the procedure returns to step (viii), otherwise the program terminates and reports the best-so-far solution and displays it graphically as a Gantt chart.

Figure 5. Adjustment operator.
4 Computational experiments

The computational experiments used data representing an 18 months schedule from a collaborating capital goods company. The first experiment identified the best DBA parameter settings. The second experiment evaluated the performance of the proposed HDBK and compared the performance with the MDBA, DBA, ABC, KH and MKH algorithms. Both experiments used the same datasets. The APS tool was experimented on a personal computer with a Core I7, 3.50 GHz CPU and 6 GB RAM.

4.1 Datasets

Pongcharoen et al. (2002) developed Genetic Algorithms for scheduling the production of capital goods and considered three problems (small, medium and large). Chainual, Lutuksin, and Pongcharoen (2007) developed an Ant Colony scheduling tool using the same problems. Xie, Hicks, and Pongcharoen (2010) additionally considered an extra-

If \( rand = 0.67 \) → Swapping operator

<table>
<thead>
<tr>
<th>Position</th>
<th>Representation of operation sequences (Position of bat ( x_i^t ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^* )</td>
<td>C, O, O, C, O, S, 1, 2, A, A, F, 1</td>
</tr>
<tr>
<td>( x_i^t )</td>
<td>C, O, O, C, O, C, O, S, 1, 2, A, A, F, 1</td>
</tr>
<tr>
<td>Swap#1 ( x_i^{t'} )</td>
<td>C, O, O, C, O, C, O, C, O, C, O, S, 1, 2, A, A, F, 1</td>
</tr>
</tbody>
</table>

Total number of swaps is 2

From New solution \( \Rightarrow D^{max} \delta \times \) Total number of swaps/adjustments

If \( D^{max} = 0.1, \delta = 0.75 \), \( x_i^t = 0.1 \times 0.75 \times 2 = 0.15 \) → 1 = Number of swaps

\[
\begin{align*}
\text{Initial Solution:} & \quad C, O, O, C, O, C, O, S, 1, 2, A, A, F, 1 \\
\end{align*}
\]
large dataset that represented a complete schedule for a major product. These four datasets were used to test: the Artificial Bee Colony (Pansuwan, Rukwong, and Pongcharoen 2010; Pongcharoen et al. 2012) and Krill Herd (Puongyeam, Pongcharoen, and Vitayasak 2014). These algorithms can be directly compared because they were applied to common datasets with the same objective function outlined in equation (1). The characteristics of the problems considered are shown in Table 5.

Table 5. The characteristics of the four problems.

<table>
<thead>
<tr>
<th>Problem sizes</th>
<th>No. of products (part number)</th>
<th>No. of items</th>
<th>Machining/assembly operations</th>
<th>No. of machines</th>
<th>Levels of product structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>2 (245, 451)</td>
<td>15</td>
<td>25/9</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Medium</td>
<td>2 (229, 451)</td>
<td>18</td>
<td>57/10</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>Large</td>
<td>2 (4, 228)</td>
<td>29</td>
<td>118/17</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>Extra-large</td>
<td>1 (227)</td>
<td>85</td>
<td>229/39</td>
<td>25</td>
<td>20</td>
</tr>
</tbody>
</table>

4.2 Identifying appropriate parameter settings

It is important to select metaheuristic parameters that obtain optimal results. A design of experiments strategy is much more effective and efficient than a trial-and-error approach. Factorial designs may be necessary to avoid misleading conclusions when interactions are present. They allow the effects of a factor to be estimated at several levels of the other factors producing results that are valid over a wide range (Montgomery 2012). Previous research on production scheduling in the capital goods has used this approach to identify appropriate parameter settings (Pongcharoen 2001; Pansuwan, Rukwong, and Pongcharoen 2010; Puongyeam, Pongcharoen, and Vitayasak 2014).

This experiment used the full factorial design to identify the appropriate parameter settings for the DBA. The factors included: (i) the combination of population size and the number of iterations ($nI$), which determines the amount of search. In the experiments the value was fixed at 2,500 to ensure comparability with previous research; (ii) the pulse rate
emission coefficient \((\gamma)\); (iii) the loudness coefficient \((\alpha)\); and (iv) the repositioning operator (the swapping operation \((SO)\) (Wang et al. 2003) or the adjustment operator \((AO)\) (Wang et al. 2005). The experiment was replicated ten times with different random number seeds. The number of runs for each replicate was \(3^3 \times 2 = 54\), giving a total of 540 runs. For each run the best-so-far penalty cost was the dependent variable. The results were analysed using a general linear model form of analysis of variance (ANOVA). The main effects and first level interactions were considered in accordance with the sparsity of effects principle that states that a system is usually dominated by main effects and low level interactions (Montgomery 2012). Table 6 shows the ANOVA table, which shows the source of variation \((Source)\), degrees of freedom \((DF)\), sum of squares \((SS)\), mean square \((MS)\), \(F\) value, and \(P\) value. The factors with a \(P\) value of \(<=0.05\) were statistically significant with a 95% confidence interval. All the DBA parameters were considered as the main sources of variation as well as the interaction effects.

From Table 6, it can be seen that all of the DBA’s parameters except \(\gamma\) were statistically significant. The only two-way interaction that was statistically significant was \(\alpha*Repositioning\) operation. The best parameter settings for the DBA were determined by considering the lowest mean best-so-far total cost obtained from main effect and interaction plots. Figure 7 shows the best combination for the interactions which were: (a) \(\alpha = 0.9\) with \(AO\); (b) \(nI = 100*25\) and \(\alpha = 0.9\); (c) \(nI = 100*25\) and \(AO\); and (d) \(\gamma = 0.1\) and \(AO\).

4.3 Performance comparison of the proposed algorithms with other approaches

The performance of the proposed algorithms (DBA, MDBA and HDBK) were compared against the ABC algorithm (Pansuwan, Rukwong, and Pongcharoen 2010), the KH and MKH algorithms (Puongyeam, Pongcharoen, and Vitayasak 2014). In each case the appropriate parameter settings had been identified through a design of experiments approach.
Each experiment was replicated 30 times to be consistent with Pansuwan, Rukwong, and Pongcharoen (2010) and Puongyeam, Pongcharoen, and Vitayasak (2014).

Table 6. ANOVA analysis of DBA parameters.

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nl</td>
<td>2</td>
<td>5,637,225,926</td>
<td>2,818,612,963</td>
<td>42.160</td>
<td>0.000</td>
</tr>
<tr>
<td>γ</td>
<td>2</td>
<td>160,270,370</td>
<td>80,135,185</td>
<td>1.200</td>
<td>0.302</td>
</tr>
<tr>
<td>α</td>
<td>2</td>
<td>29,966,267,593</td>
<td>14,983,133,796</td>
<td>224.110</td>
<td>0.000</td>
</tr>
<tr>
<td>Repositioning operation</td>
<td>1</td>
<td>4,547,201,852</td>
<td>4,547,201,852</td>
<td>68.010</td>
<td>0.000</td>
</tr>
<tr>
<td>nl*γ</td>
<td>4</td>
<td>142,679,630</td>
<td>35,669,907</td>
<td>0.530</td>
<td>0.711</td>
</tr>
<tr>
<td>nl*α</td>
<td>4</td>
<td>301,440,741</td>
<td>75,360,185</td>
<td>1.130</td>
<td>0.343</td>
</tr>
<tr>
<td>nl*Repositioning operation</td>
<td>2</td>
<td>68,137,037</td>
<td>34,068,519</td>
<td>0.510</td>
<td>0.601</td>
</tr>
<tr>
<td>γ*α</td>
<td>4</td>
<td>53,362,963</td>
<td>13,340,741</td>
<td>0.200</td>
<td>0.939</td>
</tr>
<tr>
<td>γ*Repositioning operation</td>
<td>2</td>
<td>17,403,704</td>
<td>8,701,852</td>
<td>0.130</td>
<td>0.878</td>
</tr>
<tr>
<td>α*Repositioning operation</td>
<td>2</td>
<td>1,290,334,259</td>
<td>645,167,130</td>
<td>9.650</td>
<td>0.000</td>
</tr>
<tr>
<td>Error</td>
<td>514</td>
<td>34,364,118,519</td>
<td>66,856,262</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>539</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7. Interaction plots of (a) $\alpha$ * Repositioning operation, (b) $nl$*α, (c) $nl$*Repositioning operation and (d) $\gamma$*Repositioning operation.
Table 7 shows that the performance of the HDBK, MDBA, DBA, ABC, KH and MKH in term of minimum (Min), maximum (Max), and arithmetic mean best-so-far penalty value (Mean) and standard deviation (SD). The dependent variable in this analysis was the best-so-far result achieved by each replicate. In terms of minimum total penalty cost, the HDBK outperformed the MDBA, DBA, ABC, KH and MKH for all problem sizes except the small problem.

Table 7. Performance comparison (penalty cost in currency units).

<table>
<thead>
<tr>
<th>Problems</th>
<th>Total Penalty Cost</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>HDBK</td>
</tr>
<tr>
<td>Small</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>15,000</td>
<td>15,000</td>
</tr>
<tr>
<td>Max</td>
<td>15,000</td>
<td>15,000</td>
</tr>
<tr>
<td>Mean</td>
<td>15,000</td>
<td>15,000</td>
</tr>
<tr>
<td>SD</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>31,000</td>
<td>31,500</td>
</tr>
<tr>
<td>Max</td>
<td>57,000</td>
<td>54,500</td>
</tr>
<tr>
<td>Mean</td>
<td>36,933</td>
<td>36,283</td>
</tr>
<tr>
<td>SD</td>
<td>6,276</td>
<td>5,474</td>
</tr>
<tr>
<td>Large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>163,000</td>
<td>165,000</td>
</tr>
<tr>
<td>Max</td>
<td>204,000</td>
<td>189,500</td>
</tr>
<tr>
<td>Mean</td>
<td>181,767</td>
<td>178,033</td>
</tr>
<tr>
<td>SD</td>
<td>9,081</td>
<td>6,608</td>
</tr>
<tr>
<td>Extra-large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>5,251,500</td>
<td>5,664,000</td>
</tr>
<tr>
<td>Max</td>
<td>7,434,500</td>
<td>7,361,500</td>
</tr>
<tr>
<td>Mean</td>
<td>6,350,617</td>
<td>6,572,750</td>
</tr>
<tr>
<td>SD</td>
<td>621,625</td>
<td>426,103</td>
</tr>
</tbody>
</table>

A student t test established whether the mean differences were statistically significant. Table 8 shows the T value obtained by the t-test method, the P value, and the percentage improvement (%Imp) achieved by the algorithms. Almost all of the comparisons between the results obtained from the HDBK and the other approaches for extra-large problem were statistically significant with a 95% confidence interval (P value ≤ 0.05) except the modified DBA. For medium and large problems, the statistical comparisons indicated that the results obtained from the HDBK were significantly better than the results obtained from the ABC, KH and MKH. For small problems, the results obtained from the HDBK were significantly better than the results obtained from the KH and MKH. The performance of HDBK achieved
the highest percentage improvement (\%Imp) of 47.837\% when compared with KH followed by 44.356\% when compared with MKH and 35.595\% when compared with ABC.

Table 8. Statistical analysis using a \(t\)-test.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Statistical analysis</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(T) value</td>
<td>Small</td>
</tr>
<tr>
<td>HDBK versus MDBA</td>
<td>(T) value</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>0</td>
</tr>
<tr>
<td>HDBK versus DBA</td>
<td>(T) value</td>
<td>-1.00</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.326</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>0.879</td>
</tr>
<tr>
<td>HDBK versus ABC</td>
<td>(T) value</td>
<td>-1.44</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.161</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>0.220</td>
</tr>
<tr>
<td>HDBK versus KH</td>
<td>(T) value</td>
<td>-28.21</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>20.00</td>
</tr>
<tr>
<td>HDBK versus MKH</td>
<td>(T) value</td>
<td>-4.71</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>1.426</td>
</tr>
<tr>
<td>MDBA versus DBA</td>
<td>(T) value</td>
<td>-1.00</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.326</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>0.879</td>
</tr>
<tr>
<td>MDBA versus ABC</td>
<td>(T) value</td>
<td>-1.44</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.161</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>0.220</td>
</tr>
<tr>
<td>MDBA versus KH</td>
<td>(T) value</td>
<td>-28.21</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>20.00</td>
</tr>
<tr>
<td>MDBA versus MKH</td>
<td>(T) value</td>
<td>-4.71</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>1.426</td>
</tr>
<tr>
<td>DBA versus ABC</td>
<td>(T) value</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.470</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>-0.665</td>
</tr>
<tr>
<td>DBA versus KH</td>
<td>(T) value</td>
<td>-19.86</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>19.291</td>
</tr>
<tr>
<td>DBA versus MKH</td>
<td>(T) value</td>
<td>-0.63</td>
</tr>
<tr>
<td></td>
<td>(P) value</td>
<td>0.531</td>
</tr>
<tr>
<td></td>
<td>% Imp</td>
<td>0.552</td>
</tr>
</tbody>
</table>

Remark * mean that all minimum values (total penalty cost) are identical.

5 Conclusions and future work

This research has developed a novel APS tool that effectively solves production scheduling problems for capital goods with many levels of product structure and multiple
resource constraints. It was the first research to adopt a Discrete Bat Algorithm (DBA), a Modified Discrete Bat Algorithm (MDBA) with additional local search and a Hybrid Discrete Bat Algorithm with Krill Herd (HDBK) for solving this problem. This required a novel representation to be developed that included product structure relationships and operations to be included. A repair process was included to ensure that operation and assembly precedence relationships were satisfied as well as taking into account finite capacity and avoiding deadlock. The tool was tested using four datasets obtained from a collaborating capital goods company. These had been used by previous researchers investigating ABC, KH and MKH optimisation. The first experiment identified appropriate parameter settings for the DBA. All of the DBA’s parameters except $\gamma$ were statistically significant. The only two-way interaction that was statistically significant was $\alpha \times$repositioning operation. The appropriate parameter settings for the DBA were determined by considering the lowest mean best-so-far total cost obtained from main effect and interaction plots. The best settings were $nI = 100*25$, $\gamma = 0.1$, $\alpha = 0.9$ with the adjustment operator ($AO$).

The second experiment was aimed to evaluate and compare the performance of the proposed HDBK with MDBA, DBA, ABC, KH, and MKH by using the Student $t$-test. The minimum total penalty costs indicated that the HDBK outperformed the other approaches (MDBA, DBA, ABC, KH, and MKH) for all problem sizes except small problem. Almost all of the comparisons between the results obtained from the HDBK and the other approaches for extra-large problem were statistically significant with a 95% confidence interval ($P$ value $\leq 0.05$) except the MDBA. The HDBK achieved the highest $%Imp$ 47.837 when compared with the KH. These results demonstrate that the HDBK is a promising approach for advanced planning and scheduling systems for complex scheduling situations such as those encountered in the capital goods industry.
Future research may focus on the application of mathematical analysis and/or metaheuristics to solve production scheduling problem in capital goods industry. The integration of production and preventive maintenance scheduling problem in the capital goods industry or other integrations (e.g. lot sizing, or uncertainty issues in manufacturing environment) can also be another research direction in the future.

Acknowledgement

The first author would like to acknowledge funding from the Thailand Research Fund Royal Golden Jubilee Ph.D. Scholarships Program - Grant Number PHD/0135/2552.

References


Hicks, C. 1998. "Computer Aided Production Management (CAPM) Systems in Make-to-
order / Engineer-to-order Heavy Engineering Companies." PhD, University of
Newcastle.
Hicks, C., and P.M. Braiden. 2000. "Computer Aided Production Management issues in the
engineer-to-order production of complex capital goods explored using a simulation
Holland, J. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with
Applications to Biology, Control, and Artificial Intelligence.* 1st ed. U.S.A: University of
Michigan Press.
Kang, F., J. Li, and H. Li. 2013. "Artificial Bee Colony Algorithm and pattern search
Erciyes University, Erciyes, Turkey, Report No. report-tr06.
http://dx.doi.org/10.1016/j.amc.2009.03.090.
function optimization: Artificial Bee Colony (ABC) Algorithm." *Journal of Global
Artificial Bee Colony (ABC) Algorithm and applications." *Artificial Intelligence Review*
function optimization. Paper presented at the International Conference of Soft
Computing and Pattern Recognition, Malacca, Malaysia, 4th - 7th December 2009.
Kaur, N., and A. Chhabra. 2016. Analytical review of three latest nature inspired algorithms
for scheduling in clouds. Paper presented at the Electrical, Electronics, and Optimisation
Techniques (ICEEOT), Chennai, India, 3rd-5th March 2016.
IEEE International Conference on Neural Networks, Perth, Australia, 27th November -
1st December 1995.
Algorithm based on fuzzy multi-objective technique for optimal power flow problem." 
Kizilay, D., M.F. Tasgetiren, O. Bulut, and B. Bostan. 2014. A discrete Artificial Bee Colony
Algorithm for the assignment and parallel machine scheduling problem in DYO paint
company. Paper presented at the Evolutionary Computation (CEC), Beijing, China, 6th -
11th July 2014.
Kongkaew, W. 2016. Solving the single machine total weighted tardiness problem using bat-
inspired algorithm. Paper presented at the IEEE International Conference on Industrial
Engineering and Engineering Management, IEEM 2015, Singapore, 6th - 9th December
2015.
Koza, J.R. 1990. *Genetic programming: A paradigm for genetically breeding populations of
computer programs to solve problems.* Vol. 34: Stanford University, Department of
Computer Science Stanford, CA.


Pongcharoen, P. 2001. "Genetic algorithms for production scheduling in capital goods industries." PhD, University of Newcastle upon Tyne.


