European Journal of Operational Research 250 (2016) 652–665

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

A GLNPSO for multi-level capacitated lot-sizing and scheduling problem in the poultry industry

Atiwat Boonmee, Kanchana Sethanan*

Research Unit on System Modeling for Industry, Industrial Engineering Department, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

ARTICLE INFO

Article history:
Received 30 October 2014
Accepted 10 September 2015
Available online 24 September 2015

Keywords:
Metaheuristics
Multi-level capacitated lot-sizing
Hen egg production
Particle swarm optimization
Local search

ABSTRACT

This paper presents a computation tool for the multi-level capacitated lot-sizing and scheduling problem in hen egg production planning with the aim of minimizing the total cost. A mixed-integer programming model was developed to solve small-size problems. For large-size problems, particle swarm optimization (PSO) was firstly applied. However, the component of traditional PSO for social learning behavior includes only personal and global best positions. Therefore, a variant of PSO such as the particle swarm optimization with combined gbest, lbest and nbest social structures (GLNPSO) which considers multiple social learning terms was proposed. The local search procedure was applied to decide the new sequence of chick and pullet allocation to rapidly converge to a better solution. Moreover, the re-initialization and the re-order strategy were used to improve the possibility of finding an optimal solution in the search space. To test the performance of the algorithm, the two criteria used to measure and evaluate the effectiveness of the proposed algorithm were the performance of the heuristic algorithm (P) obtained by comparing their solutions to optimal solutions, and the relative improvement of the solution (RI) obtained by the firm's current practice with respect to those of traditional PSO and the GLNPSO algorithms. The results demonstrate that the GLNPSO is not only useful for reducing cost compared to the traditional PSO, but also for efficient management of the poultry production system. Furthermore, the method used in this research should prove beneficial to other similar agro-food industries in Thailand and around the world.

1. Introduction

In the egg industry, egg production planning is considered a crucial activity because it may affect production costs. Since egg production planning concerns not only the number of chicks required over a period of time, but also how to allocate chicks to replace birds with diminishing laying capacity, egg production requires efficient planning for minimizing costs in order to maximize profit generation while meeting the customer demand. Moreover, egg production is considered a dynamic system leading to complex production, especially the chick ordering and the decision of how to allocate flocks to facilities. Allocation of flocks to facilities in the current period may affect the efficiency of allocation of the flocks in the next period. The main challenges for the firms when determining the lot size of chicks and allocating them to houses are due to the following production complexities, (1) The egg industry is a hierarchical multi-level production starting from purchasing chicks from a hatchery on a weekly basis and delivering them to the pullet-raising farms where the chicks stay until they are 17 weeks old, then transferring pullets to egg laying units (hen houses) where they are fed to have sufficient body weight to support egg production, and remain there until the age of 75 weeks. Once the hens are no longer valuable as layers, they are moved to be slaughtered at the firm's slaughter house. Hence, the number of chicks required depends on the customer demand (i.e. eggs), pullets on hand, layers on hand, egg production rate of hens and survival rate of both hens and pullets. Chick ordering is considered as medium term planning with hens to be determined before allocating pullets to hen houses (see Fig. 1). (2) There are numerous houses with limited and heterogeneous capacities. Due to an imbalance between the flow-in flocks and the capacity of the facilities, this challenges the firm attempting to allocate flocks to the facility as fully as possible, because a facility with partial fill of pullets or hens may lead to high production costs. Hence, the capacitated lot sizing problem is very important in managing chick ordering and considered as short term planning (see Fig. 2). (3) Even though the number of chicks required depends on the crucial factors mentioned earlier, the maximum ordering quantity of chicks must not exceed the maximum daily capacity of the firm's slaughter house. Thus, an egg industry should have an efficient method to make a balance between the customers' demand and the decision of how many chicks to purchase and

* Corresponding author. Tel. +66815536429, fax: +66 43 362299.
E-mail address: ksethanan@gmail.com, skanch@kku.ac.th (K. Sethanan).

http://dx.doi.org/10.1016/j.ejor.2015.09.020

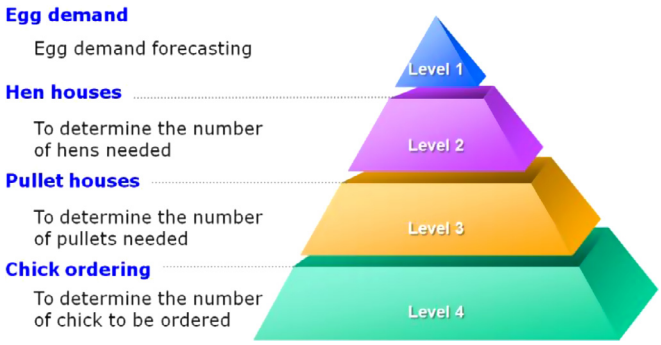


Fig. 1. Hierarchy production planning in the Poultry industry.

deliver to the industry in each time period, while the maximum number of chickens ordered in each period is restricted. However, it can be difficult and complex to combine lot sizing and scheduling problems as they are interdependent. As a result, they are often modeled and solved independently, which may lead to not obtaining globally optimal solutions (Mahdieh, Bijari, & Clark, 2011).

This paper therefore focuses on the multi-level capacitated lot sizing and scheduling problem for egg production with a restriction of the maximum lot size at each time. The objective of this research is to minimize the total costs consisting of chick ordering cost, feed cost of pullets and hens, farm utilization cost, egg storage cost, hen transportation cost from pullet houses to hen houses, and demand shortage costs. To solve the problem, interrelationships existing between the chick ordering level and chick allocation to house level are considered and taken in the model simultaneously in order to obtain globally optimal solutions. An efficient mixed integer programming model was developed for small-size problems. Since the problem considered is an NP-hard problem, the computational effort, in general, required to find an optimal solution grows exponentially with the size of the problem. In an effort to find a near optimal solution for problems with larger, more practical problems, a meta-heuristic was developed. A well-known meta-heuristic called Particle Swarm Optimization (PSO) is brought to develop an efficient algorithm.

PSO is a population based search method which was motivated by group organism behavior such as bee swarms, fish schools, and bird flocks (Kennedy & Eberhart, 1995). It imitates the physical movements of the individuals in the swarm as a search method, altogether with its cognitive and social behavior local and global exploration abilities. Hence, in this paper, the PSO algorithm is firstly applied for the purpose. However, the component of traditional PSO for social learning behavior includes only personal and global best positions. Therefore, a variant of PSO such the particle swarm optimization with combined gbest, lbest and nbest social structures (GLNPSO) developed by Pongchairerks and Kachitvichyanukul (2005, 2006) was proposed in this paper which considers multiple social learning terms including personal, global, local, and near neighbor best positions. The local search procedure is applied to decide the new sequence of chicks and hens allocated in each time period to rapidly converge to the better solution. In order to avoid being trapped into a local optimum, the re-initialization and the re-order strategy are used to improve the possibility of finding an optimal solution in the searching space. To illustrate the proposed method effectiveness, numerical experimental results were compared with the mathematical model, the current practice of the egg industry selected as a case study and also with the traditional PSO. In the next section, related literature is reviewed. In Section 3, we present the hen egg production formulation. In Section 4, we provide a comprehensive detail of our solution procedures. In Section 5, the case study application and an outline of experimental results are presented. Finally, a summary of the main findings is given in Section 6.

2. Literature review

Multi-level production lot sizing and scheduling are the most important tasks to fulfill in a manufacturing system, because they have a strong impact on its performance and productivity in terms of operating costs, machine utilization, and customer service quality. A manufacturer must make decisions about these two problems hierarchically and integrally. Due to their dependency, this problem is also called the general lot-sizing and scheduling problem (GLSP), which considers lot-sizing and scheduling simultaneously. Generally, the lot-sizing model is first determined (Karimi, Fatemi Ghomi, & Wilson, 2003; Toledo & Armentano, 2006; Ramezanian, Saidi-Mehrabad, & Fattahi, 2013), then the output from this model will be an input to incorporate the sequences of lots, which involves the determination of when each lot is produced on which machine or production resource. The multi-level lot-sizing and scheduling problems can be very complex and difficult tasks depending on the precedence constraints which have to be satisfied and on the multi-level structure (Toledo, de Oliveira, & Franca, 2013). Generally, they are classified as NP-hard optimization problems (Stockmeyer & Meyer, 2002).

The reviews of Drexl and Kimms (1997), Jans and Degraeve (2008), and Clark, Almada-Lobo, and Almeder (2011) show that the field of lot sizing and scheduling has caught the attention of researchers in order to solve real world problems. Historically, the economic order quantity (EOQ) is well known and popularly applied to determine lot-sizing that balances the setup cost and the inventory holding cost (Harris, 1913; Wilson, 1934). A few decades later, Wagner and Whitin (1958) presented a dynamic programming algorithm for the single-item uncapacitated lot-sizing problem, which provides its optimal solution. More recent work by van Hoesel and Wagelmans (2001) provided a theoretical underpinning for fully polynomial

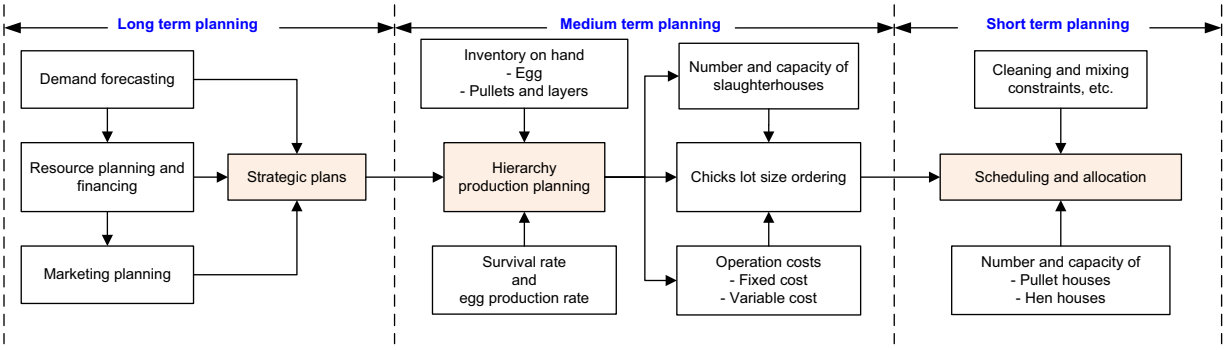


Fig. 2. Hen egg production planning.

approximation schemes for single-item capacitated economic lot sizing problems. Since industrial development and economic growth has produced more complex situations, traditional methods are cumbersome, and do not readily adapt to changes in demand, resource capacity or material availability, resulting in the need for simultaneous lot sizing and scheduling decisions. However, concurrent decision making in the lot sizing and scheduling (machine sequencing) solution is difficult and complex, so these decisions were separately made in a sequential manner: (1) estimate the available capacity for processing a set of items over required machines in the following periods, (2) determine lot sizes of different items for forthcoming planning periods, (3) determine the best schedule of machines to perform different processes on each items, regarding their lot sizes, and (4) check if the obtained production schedule is feasible in terms of required capacities; if not lead to a number of recursive corrective actions (Karimi-Nasab & Seyedhoseini, 2013). Most authors proposed algorithms and approaches for solving the lot sizing and scheduling problems separately. In previous research, various solution techniques have been applied to determine the quantity to be produced in different periods of the capacitated multistage lot-sizing problem to respond to the initially given demand forecast, while the scheduling involving the assignment of multiple tasks onto each machine was not considered (Berretta & Rodrigues, 2004; Xiao, Zhang, Zhao, Kaku, & Xu, 2014) leading to non-optimality of the above procedure due to non-simultaneity of decision making (Franck, Neumann, & Schwindt, 1997).

Branches of industry in which lot sizing and scheduling solutions should be taken concurrently are such as chemicals, drugs and pharmaceuticals, pulp and paper, textiles, foundries, glass containers, and food and beverage, along with many others (Clark, Almada-Lobo, & Almeder, 2011). In particular, agri-food production is one of the most dynamic and complex operations in most industries. The challenging task in today's food industry is that the decision-makers (agriculturist, planner of industry, or third party logistic service, etc.) often face a complex production planning problem for the allocation of scarce resources over time between competing activities to meet these requirements in an efficient manner (Kopanos, Puigjaner, & Georgiadis, 2011). The agri-foods production planning problems involve the determination of production lot sizes and the sequence of products on the capacitated machine (Guimarães, Klabjan, & Almada-Lobo, 2013) with the uncertain situations such as shelf-life and deterioration constraints of food and agricultural products, including increasing consumer attention about safety. Because of these dynamic and complex features, most lot sizing and scheduling problems in agri-food production planning are hard to solve, therefore the literature in the field of production scheduling and planning of food processing industries is a rather sparse (Kopanos, Puigjaner, Georgiadis, & Bongers, 2011). Most authors have applied mixed integer programming (MIP) for the scheduling and planning problems in agri-food processing such as dairy and yogurt (Kopanos, Puigjaner, & Georgiadis, 2010), an aggregate production planning in the sugar (da Silva, Marins, & Montevechi, 2013), shrimp farming (Yu, Leung, & Bienfang, 2006) or the meat production industry (Gribkovskaia, Gullberg, Hovden, & Wallace, 2006). The aforementioned research has not previously studied the optimal approaches for production planning and scheduling in the poultry and egg industry.

However, proofs from complexity theory as well as computational experiments indicate that the MIP formulations of most lot sizing problems, especially in the case of simultaneous consideration, are hard to solve in reasonable time for medium or large size problems (Jans & Degraeve, 2007; Pochet & Wolsey, 2006). Attempting to fill this gap, therefore, various solution techniques have been proposed to solve them. In the past decade, meta-heuristics have become popular and efficient tools for solving hard combinatorial optimization problems (Jans & Degraeve, 2007). Recently there has been significant research effort to apply evolutionary computation (EC) techniques for lot sizing and scheduling problems. For example, Toledo, da Silva Arantes, de Oliveira, and Almada-Lobo (2013) proposed a new hybrid evolutionary algorithm combining a multi-population hierarchically structured genetic algorithm (GA) with simulated annealing (SA), and a tailor-made heuristic named cavity heuristic (CH) to tackle the short-term production planning and scheduling problems. The hybrid evolutionary method presents a promising performance outperforming literature results. Recently, Seeanner, Almada-Lobo, and Meyr (2013) presented a new idea to solve the kind of problems that typically arise in the consumer goods industry due to sequence-dependent setups and shifting bottlenecks. The variable neighborhood decomposition search (VNDS) with the MIP-based Fix & Optimize heuristic is cross-fertilized. The heuristic provides very good solutions for a real-world problem instance and were significantly better for most of the problem instances when compared to the Xpress MIP Solver. For a real-world production problem such as soft drink production, a synchronized and integrated two-level lot sizing and scheduling problem must be considered. Therefore, as presented in Toledo, de Oliveira, de Freitas Pereira, França, and Morabito (2014), a genetic algorithm embedded with mathematical programming techniques was applied. The computational results show that this evolutionary/mathematical programming approach outperforms the literature methods in terms of production costs and computational times when applied to a set of real-world problem instances provided by a soft drink company.

Particle Swarm Optimization (PSO) is one of the most powerful and interesting methods (Dye & Hsieh, 2010; Dye & Ouyang, 2011; Nearchou, 2011; Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007). PSO was first proposed by Kennedy and Eberhart in 1995 (Kennedy & Eberhart, 1995) and involves applying logic inspired by social behavior of bird flocking or fish schooling. It is a population based stochastic optimization technique which shares many similarities with evolutionary computation techniques such as GA. The random solutions are initialized with a population called particles and fly through the problem space to search for optima by updating generations. Compared to other evolutionary computation techniques such as GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. However, the traditional PSO has problems of easily falling into local solutions, slow convergence speed and low convergence accuracy, especially for NP-hard problems (Song, Kong, Gan, & Su, 2008). Therefore, since its first publication, a large amount of research has been done to study performance and improve it (Clerc, 2006; Kennedy, Eberhart, & Shi, 2001). Additionally, how to improve the global convergence ability has been the main research scope. In order to enhance its performance and alleviate the deficiencies to the problem solving, many variants of the PSO algorithm have been proposed (Pratchayaborirak & Kachitvichyanukul, 2007; Shi & Eberhart, 1998; Veeramachaneni, Peram, Mohan, & Osadciw, 2003; Wang & Yeh, 2014). One of the powerful and interesting methods among the many variants of the PSO algorithm is particle swarm optimization combined with gbest, lbest and nbest social structures (GLNPSO). It is effective since it enables the swarm to explore different parts of the search spaces simultaneously (see, Pongchairerks & Kachitvichyanukul, 2005, 2006).

Moreover, experimental results reveal that the hybrids of PSO can effectively produce improved solutions over traditional methods with faster convergence, such as GA, ant colony system (ACS), and Tabu Search. In addition, local search techniques are often used to improve the current solution by moving repeatedly from the current solution to find good solutions in reasonable time. In recent years, it is observed from the literature that meta-heuristics hybridized with a local search procedure often give the best searching results (Lin, Hsu, & Hsu, 2013; Yu, Lin, Lee, & Ting, 2010). Therefore, a hybrid approach based on a variant of the particle swarm optimization, called GLNPSO with the local search procedure is proposed in this paper.

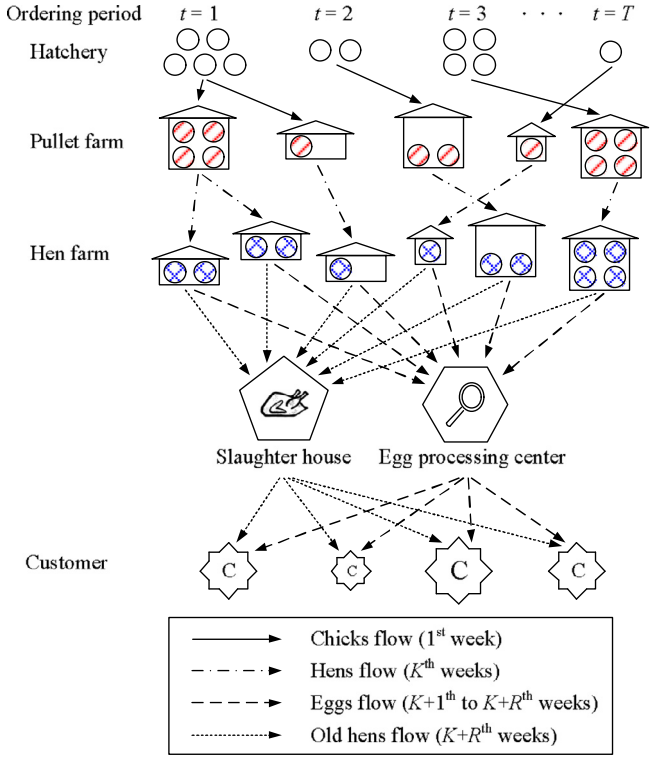


Fig. 3. Egg production supply chain.

3. Hen egg production problem

3.1. Problem overview

A brief description of the problem is reviewed in order to help in understanding the mathematical formulation. The problem involves the multi-level capacitated lot sizing and scheduling problem for egg production with a restriction on the maximum lot size at each time. Presently, the weekly chicks order volumes from hatcheries depend on egg demand volumes in the planning horizon. Then the chicks are transferred to the contracted pullet-raising farms. When the pullets are K weeks old, they are moved to hen houses. These hens will remain in the hen house until the age of $K + R$ weeks for egg production. After that, they are slaughtered at the firm's slaughter house (see Fig. 3). Hence, the number of chicks required depends on the egg demand, pullets on hand, layers on hand, egg production rate of hens, the sizes (capacities) of pullet and hen houses available, each of which normally ranges from small to large, and the capacity of the slaughter houses. To allocate chicken to pullet houses and hen houses, however, the maximum ordering quantity of chicks must not exceed the maximum daily capacity of the firm's slaughter house. This section presents a 0–1 mixed integer programming model with the objective of minimizing the total costs consisting of chick ordering cost, feed cost of pullets and hens, farm utilization cost, egg storage cost, hens transportation cost from pullet houses to hen houses, and demand shortage costs. The model is presented below with a brief explanation of each constraint.

3.2. A mixed integer programming formulation

A mixed integer programming formulation was developed as follows: in the problem there are I pullet houses, J hen houses and the planning horizon is finite and consists of T time periods. The egg demand for each time period is known in each time period and satisfied at the beginning of the period. The ordering of chicks will be made in terms of batches and must not exceed the total capacity of slaughterhouses. To allocate chicks and hens, the mixing of flocks at different ages in the same facility is not allowed. The chicks are transferred to pullet-raising farms where they stay until they reach a certain age (K weeks of age). These pullets are then moved to hen houses, and will remain in the laying unit (i.e. hen house) until a certain age ($K + R$ weeks of age). Once the pullets or hens are moved out from the houses, the houses are cleaned in order to be free from infections for C^P and C^H weeks, respectively. During this cleaning stage, no chicks or hens can be placed in the farm. Parameters and decision variables used in formulating the model are defined in the following.

Indices

i	index for pullet house; $i = 1,2,\ldots,I$
j	index for hen house; $j = 1,2,\ldots,J$
k	index for pullet age; $k = 1,2,\ldots,K$
r	index for hen age; $r = 1,2,\ldots,R$
t	index for time period; $t = 1,2,\ldots,T$

Parameters

DM_t	demand for eggs at each time period t
C_i^P	capacity of pullet house i
C_j^H	capacity of hen house j
P_r	production rate of eggs becoming laying hens at age r
$D_{i,j}$	distance from pullet house i to hen house j
$IQ_{i,k}^P$	number of pullets on hand in pullet house i aged k at the starting period
$IQ_{i,j,r}^H$	number of layers on hand transferred from pullet house i to hen house j aged r at the starting period
EG	initial number of eggs stored at the starting period
XC	total capacity of slaughterhouse
LS	number of chicks per lot size
C^P	cleaning time of pullet house after moving pullets to laying hen houses
C^H	cleaning time of hen house after moving hens to slaughterhouse
A	cost of chick ordering
B	transportation cost
ES	cost of egg storage
F^P	feed cost per unit time per pullet
F^H	feed cost per unit time per hen
O^P	cost of pullet farm utilization
O^H	cost of hen farm utilization
LC	cost of loss from inability to respond to demand
m	big number

Integer variables

BC_t	number of quantity (lot) of chick ordered at time period t
INV_t	number of eggs stored at time period t
BEG_t	number of eggs lost that cannot satisfy demands at time period t
$Q_{i,k,t}^P$	number of pullets in pullet house i aged k at time period t
$Q_{i,j,r,t}^H$	number of hens transferred from pullet house i to hen house j aged r at time period t

Binary variables

OD_t	= 1, representing chicks were ordered at time period t. Otherwise 0
$AC_{i,t}^P$	= 1, representing chicks were allocated to the pullet house i at time period t. Otherwise 0
$AC_{j,t}^H$	= 1, representing hens were allocated to the hen house j at time period t. Otherwise 0
$AQ_{i,k,t}^P$	= 1, representing pullet house i has pullets aged k at time period t. Otherwise 0
$AQ_{j,r,t}^H$	= 1, representing hen house j has hens aged r at time period t. Otherwise 0

By using the notation above, the mathematical formulation for multi-level production lot sizing and scheduling problem to solve the hen egg production planning is presented as follows:

Objective function

$$\begin{aligned}\text{Minimize Z} = &\sum_{t}^{T} OD_t \times A + \left(\sum_{i}^{I}\sum_{k}^{K}\sum_{t}^{T} Q^{P}_{i,k,t} \times F^{P} + \sum_{i}^{I}\sum_{j}^{J}\sum_{r}^{R}\sum_{t}^{T} Q^{H}_{i,j,r,t} \times F^{H}\right) \\ &+ \left(\sum_{i}^{I}\sum_{t}^{T} AC^{P}_{i,t} \times O^{P} + \sum_{j}^{J}\sum_{t}^{T} AC^{H}_{j,t} \times O^{H}\right) \\ &+ \left(\sum_{t}^{T} INV_t \times ES\right) \\ &+ \sum_{i}^{I}\sum_{j}^{J}\sum_{t}^{T} D_{i,j} \times Q^{H}_{i,j,r=1,t} \times B + \sum_{t}^{T} BEG_t \times LC \end{aligned} \quad (1)$$

Constraints

$$OD_t \times m - \sum_{i=1}^{I} Q^{P}_{i,k,t} \geq 0; \quad \forall t, \quad k = 1 \quad (2)$$

$$BC_t \times LS = \sum_{i=1}^{I} Q^{P}_{i,k,t}; \quad \forall t, \quad k = 1 \quad (3)$$

$$\sum_{i=1}^{I} Q^{P}_{i,k,t} \leq XC \times OD_t; \quad \forall t, \quad k = 1 \quad (4)$$

$$EG + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{r=1}^{R} Q^{H}_{i,j,r,t} \times P_r - DM_t = INV_t - BEG_t; \quad t = 1 \quad (5)$$

$$INV_{t-1} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{r=1}^{R} Q^{H}_{i,j,r,t} \times P_r - DM_t = INV_t - BEG_t; \quad t > 1 \quad (6)$$

$$Q^{P}_{i,k+1,t} = IQ^{P}_{i,k}; \quad \forall i, \quad 1 \leq k \leq K-1, \quad t = 1 \quad (7)$$

$$Q^{P}_{i,k+1,t+1} = Q^{P}_{i,k,t}; \quad \forall i, \quad 1 \leq k \leq K-1, \quad 1 \leq t \leq T-1 \quad (8)$$

$$Q^{H}_{i,j,r+1,t} = IQ^{H}_{i,j,r}; \quad \forall i, j \quad 1 \leq r \leq R-1, \quad t = 1 \quad (9)$$

$$Q^{H}_{i,j,r+1,t+1} = Q^{H}_{i,j,r,t}; \quad \forall i, j \quad 1 \leq r \leq R-1, \quad 1 \leq t \leq T-1 \quad (10)$$

$$\sum_{j=1}^{J} Q^{H}_{i,j,r,t} = IQ^{P}_{i,k}; \quad \forall i, \quad k = K, \quad r = 1, \quad t = 1 \quad (11)$$

$$\sum_{j=1}^{J} Q^{H}_{i,j,r,t} = Q^{P}_{i,k,t}; \quad \forall i, \quad k = K, \quad r = 1, \quad 1 \leq t \leq T-1 \quad (12)$$

$$\sum_{k=1}^{K} Q^{P}_{i,k,t} \leq C^{P}_{i} \times AC^{P}_{i,t}; \quad \forall i, t \quad (13)$$

$$C^{P}_{i} \times AQ^{P}_{i,k,t} \geq Q^{P}_{i,k,t}; \quad \forall i, k, t \quad (14)$$

$$\sum_{k=1}^{K} AQ^{P}_{i,k,t} = AC^{P}_{i,t}; \quad \forall i, t \quad (15)$$

$$\sum_{r=1}^{R} Q^{H}_{i,j,r,t} \leq C^{H}_{j} \times AC^{H}_{j,t}; \quad \forall i, j, t \quad (16)$$

$$C^{H}_{j} \times AQ^{H}_{j,r,t} \geq \sum_{i=1}^{I} Q^{H}_{i,j,r,t}; \quad \forall j, r, t \quad (17)$$

$$\sum_{r=1}^{R} AQ^{H}_{j,r,t} = AC^{H}_{j,t}; \quad \forall j, t \quad (18)$$

$$\sum_{t}^{t+K+C^{P}-1} AC^{P}_{i,t} \leq K; \quad \forall i, \quad 1 \leq t \leq T-K-C^{P}+1$$

$$\text{where } T \geq K + C^{P} \quad (19)$$

$$\sum_{t}^{t+R+C^{H}-1} AC^{H}_{j,t} \leq R; \quad \forall j, \quad 1 \leq t \leq T-R-C^{H}+1$$

$$\text{where } T \geq R + C^{H} \quad (20)$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} Q^{H}_{i,j,r,t} \leq XC; \quad \forall t, \quad r = R \quad (21)$$

$$BC_t,\ INV_t,\ BEG_t,\ Q^{P}_{i,k,t},\ Q^{H}_{i,j,r,t} \geq 0; \quad \forall i, j, k, r, t \quad (22)$$

$$OD_t,\ AC^{P}_{i,t},\ AC^{H}_{j,t},\ AQ^{P}_{i,k,t},\ AQ^{H}_{j,r,t} \in \{0, 1\}; \quad \forall i, j, k, r, t \quad (23)$$

This model is developed with the objective to minimize the total cost in planning horizon (see Eq. (1)). The first term is chick ordering cost. The second term is feed costs of pullets and hens. The third term is farm utilization cost. This cost can be farm renting from farmers in a contract farming system or operation and depreciation costs for raising hens if the firm owns the farms. The fourth term is eggs storage cost. The fifth term is the transportation cost from pullet houses to hen houses and the sixth term is egg demand shortage cost.

Constraints (2) and (3) ensure that chicks aged 1 week will be purchased and delivered to a pullet house. Constraint (4) ensures that the chick ordering in each time period must not exceed the capacity of the slaughterhouse. Constraints (5) and (6) determine the relationship among demands, inventory levels of egg, and planned quantities. Demand of eggs supplied in each time period, and total egg in-flow from each hen house and eggs stored in distribution center should be equal to total the egg out-flow to customer and eggs stored at time period $t = 1$ and time period $t > 1$ respectively. Since there could be pullets on hands at the starting period (i.e., $t = 1$) in each pullet house i, Constraint (7) is required to determine the age of the pullets from k weeks old since the last period (period $t-1$ or period $t = 0$) to $k+1$ weeks old in the starting period. For example, if the pullets in pullet house i were 1 week old at period $t-1$, then at the starting period, these pullets are 2 weeks of age. Constraint (8) is required to determine the age of the pullets in each pullet house for any period t where $1 \leq t \leq T-1$. In a similar way, Constraint (9) is required to determine the age of hens on hand in each hen house j at the starting period (i.e., $t = 1$), while Constraint (10) is needed to determine the age of hens in each hen house for any period t where $1 \leq t \leq T-1$. Constraints (11) and (12) ensure that the pullets will remain in the pullet house until the age of K weeks. Then they are transferred to egg laying unit (hen house) to support egg production. Constraints set (13)–(18) ensure that the pullets and the laying hens in each house must not exceed its capacity. Constraints (19) and (20) ensure that chicks cannot be allocated to pullet house, and hens cannot be allocated to hen house until the cleaning stage is finished, respectively. Constraint (21) ensures that hens will remain in the hen house until the age of R weeks. Then they are slaughtered at the slaughterhouse with the total number of hens aged R weeks from all hen houses. The total number of hens transferred to the slaughterhouse must not exceed the slaughterhouse capacity in each time period. Relations (22) and (23) represent the types of variables.

4. Metaheuristic development

4.1. Particle swarm optimization

When the size of problems becomes too large and too complicated to be solved by a mathematical model, particle swarm optimization (PSO) is typically employed. Particle swarm optimization was first proposed by Kennedy and Eberhart in 1995 (Kennedy & Eberhart, 1995). PSO is a population based random search method that imitates the physical movements of the individuals in the swarm as a searching mechanism. Within the swarm, the learning mechanisms are based on the cognitive and social behavior of particles. In the traditional PSO algorithm, it consists of a population of particles initialized with random positions and velocities. This population of particles is usually called a swarm.

In one iteration step, each particle is firstly evaluated to find each individual fitness or objective function value. For each particle, if a position is reached which has a better objective function than the previous best position, the personal best position (*pbest*) is updated. Also, if a current objective function is better than the previous best objective function of the whole swarm, the global best position (*gbest*) is updated. The velocity of each particle is then updated based on three terms: (1) the velocity term, (2) the gap between the current position and the particle's personal best position, and (3) the gap between the current position and the global best position found so far by the swarm. Every particle is then moved from the current position to the new position based on its updated velocity. The process repeats until the stopping criterion is met. The velocity and position of each particle are updated by the Eqs. (24) and (25), respectively.

$$\omega_{lh}(\tau+1) = w(\tau)\omega_{lh}(\tau) + c_p u(\psi_{lh} - \theta_{lh}(\tau)) + c_g u(\psi_{gh} - \theta_{lh}(\tau)) \quad (24)$$

$$\theta_{lh}(\tau+1) = \theta_{lh}(\tau) + \omega_{lh}(\tau+1) \quad (25)$$

where, $\omega_{lh}(\tau)$ is the velocity of the lth particle at the hth dimension in the τth iteration, c_p and c_g are called cognitive and social coefficients, u are uniform random numbers in the interval [0,1]. ψ_{lh} is the best previous position of the lth particle, $\theta_{lh}(\tau)$ is the position of the lth particle at the hth dimension in the τth iteration, and ψ_{gh} is the best position among the whole swarm. $w(\tau)$ is the inertia weight in the τth iteration that exerts the effect of the old velocity on the new one.

4.2. Solution representation

For the traditional PSO, the component for social learning behavior includes only personal and global best positions. To solve a lot-sizing and scheduling problem for the poultry industry, the modified GLNPSO was thus proposed to improve the research solution. This technique is a hybrid approach based on the variant of PSO algorithm called GLNPSO developed by Pongchairerks and Kachitvichyanukul (2005, 2006) with the local search procedure. The GLNPSO algorithm is an extension of the traditional PSO where each particle communicates with multiple specific subsets of the swarm. Each particle consists of the personal best position (*pbest*), global best position (*gbest*), local best position (*lbest*), and near neighbor best position (*nbest*), for updating the particle velocities in order to improve the search performance. For each iteration step, a local search procedure with neighborhood search solution was used. Then the inertia weight is the parameter affecting particle movement. High inertia weight means that the particles tend to maintain the current direction and low inertia weight means the particles may be influenced more by the cognitive and social terms. The GLNPSO algorithm for the hen egg production planning is described and the notation used is as follows.

Notation

τ	index for iteration; $\tau = 1, 2, \ldots, \Gamma$
ri	index for re-initialization's iteration of particles; $ri = 1, 2, \ldots, RI_{max}$
ro	index for re-order's iteration of chick ordering; $ro = 1, 2, \ldots, RO_{max}$
l	index for particle; $l = 1, 2, \ldots, L$
h	index for dimension; $h = 1, 2, \ldots, H$
V^R	uniform random number in the interval [0,1]
V^C, V^P	random number generation of 0 or 1
ψ^L_{lh}	local best position (*lbest*) of the lth particle at the hth dimension
ψ^N_{lh}	near neighbor best position (*nbest*) of the lth particle at the hth dimension
G	number of adjacent neighbors
c_l	local best position acceleration constant
c_n	near neighbor best position acceleration constant
$\theta^{\max}$	maximum position value
$\theta^{\min}$	minimum position value
Θ_l	vector position of the lth particle, $[\theta_{l1}, \theta_{l2}, \ldots, \theta_{lH}]$
Ω_l	vector velocity of the lth particle, $[\omega_{l1}, \omega_{l2}, \ldots, \omega_{lH}]$
Ψ_l	vector personal best position of the lth particle, $[\psi_{l1}, \psi_{l2}, \ldots, \psi_{lH}]$
Ψ_g	vector global best position, $[\psi_{g1}, \psi_{g2}, \ldots, \psi_{gH}]$
Ψ^L_l	vector local best position of the lth particle, $[\psi^L_{l1}, \psi^L_{l2}, \ldots, \psi^L_{lK}]$
R_l	the lth set of solutions
$Z(\Theta_l)$	fitness value of Θ_l
FDR	fitness-distance ratio

The GLNPSO is a population-based search method. A swarm of L particles is generated from a random position (Θ_l) in the range $[\theta^{\min}, \theta^{\max}]$ with a velocity (Ω_l). In each GLNPSO iteration, the solution of particle (R_l) consists of chicks required in each time period and the allocation sequence of chicks to pullet house and hens to hen house based on a ranked-order value (ROV) rule with the fitness value of position, represented by $Z(\Theta_l)$. Since, the problem considered is very complex in terms of number of chicks purchased and delivered to the industry with the capacitated lot sizing and the allocation of chicken to farms, some parts of the GLNPSO algorithm are modified in order to determine the number of chicks ordered in each time period and suitably applied for allocation of chickens to farms. Therefore, after the decoding process, the local search algorithm is added. Finally, if the objective value does not improve, Re-initialization and Re-order processes are conducted.

In the GLNPSO procedure, there are four operations: solution initialization, local search generation, update the velocity and the position, and re-initialization and re-order operations. The overall steps of the GLNPSO are presented in Fig. 4 and details of each operation are presented below.

***Step 1:* Solution initialization**

A particle is a set of parameters which defines a proposed solution to the problem that the GLNPSO algorithm is trying to solve. Generating the initial population is a crucial part of the particle swarm optimization algorithm and a good initial population directly affects the quality of the solution. The design of the population and its parameters is represented in the encoding and decoding procedures as follows.

1) Encoding step

In the encoding step, each particle in the swarm is separated into two parts. The first part is the number of chicks purchased and delivered to the poultry industry in each time period and the second part is the allocation of chicks and pullets to farms.

The first part of a particle: chick ordering part

We adapted the hierarchy with 4-levels of production planning (egg demand, hen farming, pullet farming and chick ordering) and a lot for lot technique to determine the initial chick ordering. To understand the initial chick ordering process clearly, an example for the hen egg production planning problem is presented. To simplify the

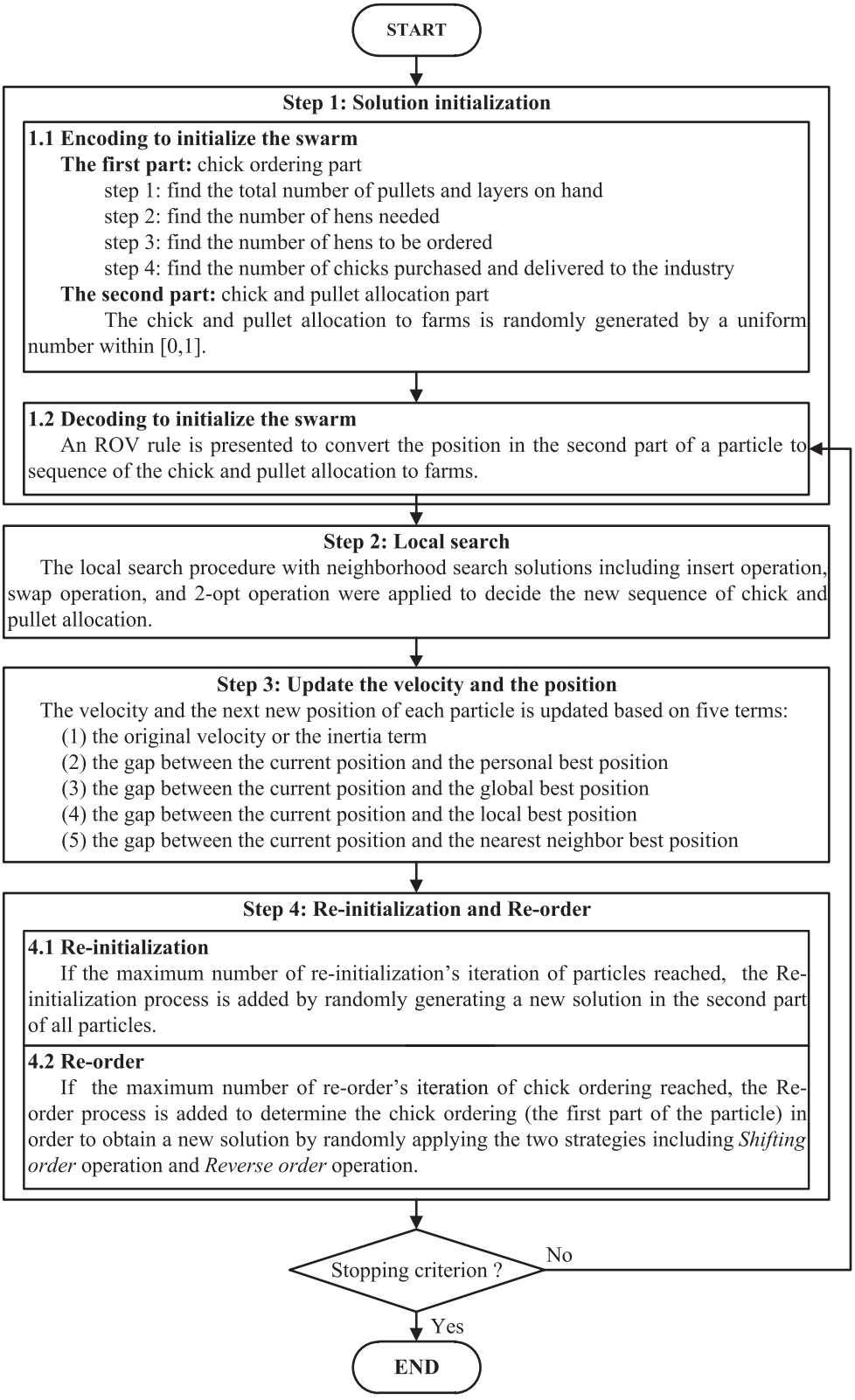
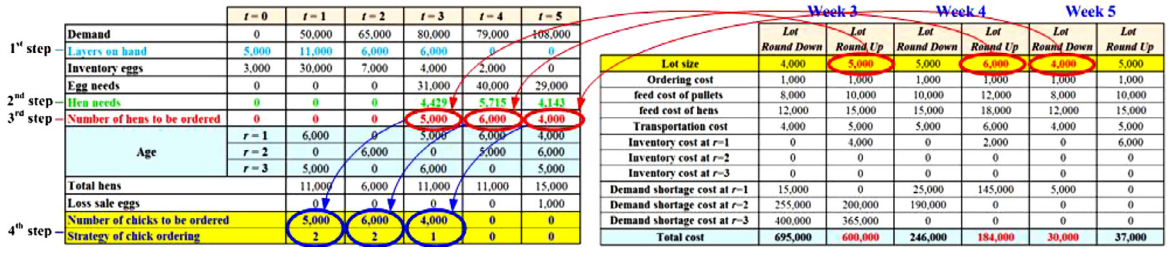


Fig. 4. The GLNPSO algorithm process.

problem, Table 1 gives the egg demands for a small hen egg production planning problem with five pullet houses and eight hen houses with five weeks of total time periods in the planning horizon. In this example, we consider the maximum ages of pullets and hens are two and three weeks, respectively. When pullets or hens are completely

Table 1
Egg demands in planning horizon.

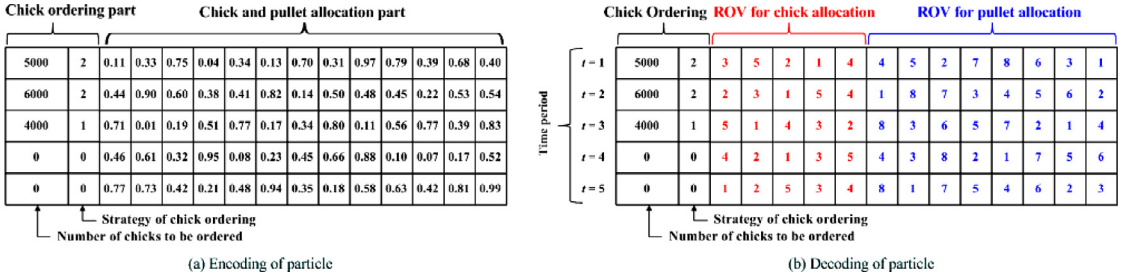
Week	1	2	3	4	5
Demand	50,000	65,000	80,000	79,000	108,000



	t = 0	t = 1	t = 2	t = 3	t = 4	t = 5
Demand	0	50,000	65,000	80,000	79,000	108,000
Layers on hand	5,000	11,000	6,000	6,000	0	0
Inventory eggs	3,000	30,000	7,000	4,000	2,000	0
Egg needs	0	0	0	31,000	40,000	29,000
Hen needs	0	0	0	4,429	5,715	4,143
Number of hens to be ordered	0	0	0	5,000	6,000	4,000
Age r = 1		6,000	0	5,000	6,000	4,000
Age r = 2		0	6,000	0	5,000	6,000
Age r = 3		5,000	0	6,000	0	5,000
Total hens		11,000	6,000	11,000	11,000	15,000
Loss sale eggs		0	0	0	0	1,000
Number of chicks to be ordered		5,000	6,000	4,000	0	0
Strategy of chick ordering		2	2	1	0	0

	Week 3 Lot Round Down	Week 3 Lot Round Up	Week 4 Lot Round Down	Week 4 Lot Round Up	Week 5 Lot Round Down	Week 5 Lot Round Up
Lot size	4,000	5,000	5,000	6,000	4,000	5,000
Ordering cost	1,000	1,000	1,000	1,000	1,000	1,000
feed cost of pullets	8,000	10,000	10,000	12,000	8,000	10,000
feed cost of hens	12,000	15,000	15,000	18,000	12,000	15,000
Transportation cost	4,000	5,000	5,000	6,000	4,000	5,000
Inventory cost at r=1	0	4,000	0	2,000	0	6,000
Inventory cost at r=2	0	0	0	0	0	0
Inventory cost at r=3	0	0	0	0	0	0
Demand shortage cost at r=1	15,000	0	25,000	145,000	5,000	0
Demand shortage cost at r=2	255,000	200,000	190,000	0	0	0
Demand shortage cost at r=3	400,000	365,000	0	0	0	0
Total cost	695,000	600,000	246,000	184,000	30,000	37,000

Fig. 5. Example of initial chick ordering.



5000	2	0.11	0.33	0.75	0.04	0.34	0.13	0.70	0.31	0.97	0.79	0.39	0.68	0.40
6000	2	0.44	0.90	0.60	0.38	0.41	0.82	0.14	0.50	0.48	0.45	0.22	0.53	0.54
4000	1	0.71	0.01	0.19	0.51	0.77	0.17	0.34	0.80	0.11	0.56	0.77	0.39	0.83
0	0	0.46	0.61	0.32	0.95	0.08	0.23	0.45	0.66	0.88	0.10	0.07	0.17	0.52
0	0	0.77	0.73	0.42	0.21	0.48	0.94	0.35	0.18	0.58	0.63	0.42	0.81	0.99

Time period														
t = 1	5000	2	3	5	2	1	4	4	5	2	7	8	6	3	1
t = 2	6000	2	2	3	1	5	4	1	8	7	3	4	5	6	2
t = 3	4000	1	5	1	4	3	2	8	3	6	5	7	2	1	4
t = 4	0	0	4	2	1	3	5	4	3	8	2	1	7	5	6
t = 5	0	0	1	2	5	3	4	8	1	7	5	4	6	2	3

Fig. 6. Example of a particle construction.

removed from houses, pullet houses and hen houses are cleaned within one and two weeks, respectively. At the starting period, the number of eggs stored in the distribution center is 3,000 eggs, the number of pullets on hand in pullet house No.3 with the age of two weeks ($k = 2$) is 6,000 and the number of layers on hand in hen house No.5 with the age of one week ($r = 1$) is 5,000. Total capacity of slaughterhouse is 10,000 chickens per week and the number of chicks per lot size is 1,000 chicks with the fixed cost of chick ordering being 1,000 units per order. For each week, feed cost of pullet and hen is one unit per chicken and holding cost for eggs is one unit per egg per time period. Hen transportation cost from pullet house to hen house is one unit per kilometer per hen and the demand shortage cost is five units per egg. To find the initial chick order in each period, the four steps of initial chick ordering process can be explained as follows.

In the first step, the total number of pullets and layers on hand at the starting period is converted into total layers on hand in each week. Then the number of hens needed in each week is obtained in the second step by considering egg demand in the planning horizon, total number of hens, and number of eggs stored at the starting period. In the third step, the number of hens to be ordered in terms of lot size is obtained by considering the minimum total cost consisting of ordering cost, feed cost of pullets and hens, transportation cost, inventory cost and demand shortage cost. Four strategies of chick ordering are considered: (1) Strategy 0: It is the case that there are no chicks to be ordered or the chick order quantity is zero. (2) Strategy 1: *Lot Round Down* strategy: In this case, the decrement of the chick ordering is one lot. (3) Strategy 2: *Lot Round Up* strategy: In this case, the increment of the chick ordering is one lot. (4) Strategy 3: *Shifting order* strategy: This case is performed to find the new solution of chick ordering in the Re-order operation. In this step, the number of hens to be ordered is derived from the minimum total cost between two strategies, strategy 1: *Lot Round Down* strategy and strategy 2: *Lot Round Up* strategy. However the maximum number of chicks to be ordered for either Strategies 1 or 2 must not exceed the maximum capacity of the firm's slaughterhouse. In the last step, the number of chicks purchased and delivered to the industry in each week is given by adapting the lot for lot technique with lead time of K weeks. The example of initial chick ordering from the first step to the final step is shown in Fig. 5.

The second part of a particle: chick and pullet allocation part

In the encoding step in the second part of the particle, the chick and pullet allocation to farms is randomly generated by a uniform number within [0,1] for the position of $I+J$ column and T row.

2) Decoding step

In the decoding step, we present an ROV rule to convert the position in the second part of a particle to a sequence of the chick and pullet allocation to farms. The ROV rule is described as follows. For the first I positions (total number of pullet houses), the largest position value of a particle is first selected and assigned the smallest rank value 1, then the second largest position value is selected and assigned rank value 2. All of the I position values are handled to a sequence of chick allocations to farms. And the next J positions (total number of hen houses) are processed in the same way.

Illustrations of an initial particle obtained in the encoding step and the decoding step are shown in Fig. 6(a) and (b), respectively. Fig. 7 gives an illustration of an example solution derived by Fig. 6.

***Step 2:* Local search**

If chicks and hens are randomly allocated, the solution space may be enlarged, and a better solution may be found at the expense of computation time. Thus the local search algorithm is adopted to decide the new sequence of chicks and pullets allocated in each time period to rapidly converge to a better solution.

A local search procedure with neighborhood search solutions was used and is embedded in the PSO algorithm. The neighborhood search solution has various types of operations, including insert operation, swap operation, and 2-opt operation to solve problems. These operations are commonly embedded in PSO and other metaheuristics. Let set $N(X)$ be defined as the set of solutions neighboring a solution X from the allocation sequence of chicks or hens (the second part of the particle). For each particle, the next solution Y is randomly selected from $N(X)$ either by an insert operation, swap operation or

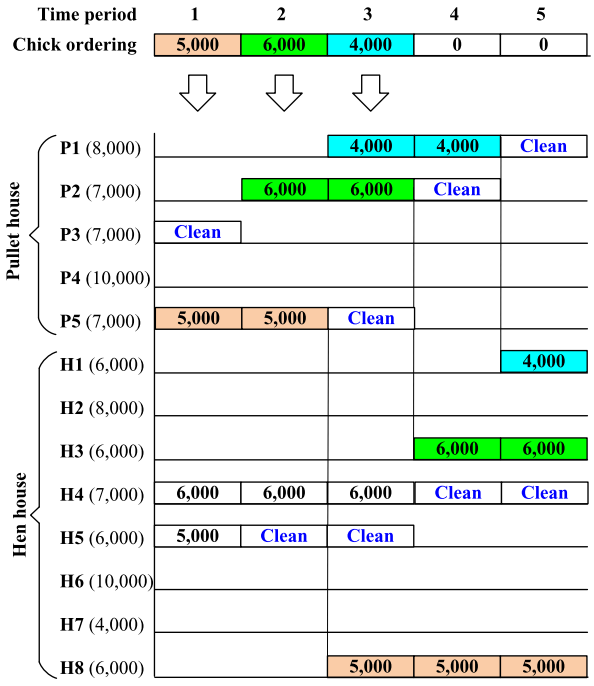


Fig. 7. The illustration of example solution given in Fig. 6.

2-opt operation. This process continues until the last neighborhood solution of X is reached.

The three solutions of the neighborhood search solution can be explained as follows: (1) the insert operation is performed by selecting the first position of X and inserting it into the position immediately before the randomly selected ith position of X (see Fig. 8(a)), (2) the swap operation is carried out by a randomly selected ith position of X and exchanged with the first position of X (see Fig. 8(b)) and (3) the 2-opt operation is implemented by selecting the first position of X and the random ith position of X, and then reversing the substring in the solution representation between them (see Fig. 8(c)).

In this paper the development of the local search algorithm is divided into two phases. In the first phase, chick allocation to pullet houses was considered and then pullet allocation to hen houses was performed in the second phase. The two parameters V^C and V^P are randomly generated between 0 or 1 as the probability to perform the local search for chick allocation phase and pullet allocation phase, respectively. If the local search has been done, time period (t) is randomly selected, and then the solution selected by randomly generating V^R in the range [0,1] for neighborhood search in order to generate a new solution (Y_t) with the three cases: (1) by the insert operation, if $V^R \leq 1/3$, (2) by the swap operation, if $1/3 < V^R \leq 2/3$ and (3) by the 2-opt operation, if $2/3 < V^R \leq 1$. The local search solution for the GLNPSO algorithm can be explained step by step as follows.

Algorithm Local search

1. **Input:** Θ_l, Ψ_l , Ψ_g
2. **For** $l = 1 : L$
3. Random number generation V^C and V^P of 0 or 1
4. **If** $V^C == 1$ // Perform local search for chick allocation to farm
5. **Then** Random integer number generation t = random [1,T]
6. Generate a new solution Y_t based on allocation sequence of chicks at time period t by generating V^R= random [0,1] to find the neighborhood search solution

Endif

7. **If** $V^P == 1$ // Perform local search for pullet allocation to farm
8. **Then** Random integer number generation t = random [1,T]
9. Generate a new solution Y_t based on allocation sequence of hens at time period t by generating V^R= random [0,1] to find the neighborhood search solution

Endif

10. **If** $Z(\Theta_l) < Z(\Psi_l)$ **Then** Update $\Psi_l = \Theta_l$ **Endif**
11. **If** $Z(\Psi_l) < Z(\Psi_g)$ **Then** update $\Psi_g = \Theta_l$ **Endif**
12. $l \leftarrow l + 1$
13. **Endfor**
14. **Output:** Θ_l, Ψ_l and Ψ_g

***Step 3:* Update the velocity and the position**

In the next iteration, every particle moves from one position to the next position by the summing of five terms: (1) the current velocity term, (2) the personal best position term, (3) the global best position term, (4) the local best position term, and (5) the nearest neighbor best position term. In the current velocity term, the value of inertia weight (w) is used to improve the search ability by linearly decreasing it (Shi & Eberhart, 1998) as Eq. (26), and then Eq. (27) is applied as the updating velocity of each particle in each iteration and the next new position of the particle is updated by Eq. (25).

$$w(\tau) = w(\Gamma) + \frac{\tau - \Gamma}{1 - \Gamma}[w(1) - w(\Gamma)] \tag{26}$$

$$\begin{aligned}\omega_{lh}(\tau+1) = {} & w(\tau)\omega_{lh}(\tau) + c_p u(\psi_{lh} - \theta_{lh}(\tau)) + c_g u(\psi_{gh} - \theta_{lh}(\tau)) \\ & + c_l u(\psi^L_{lh} - \theta_{lh}(\tau)) + c_n u(\psi^N_{lh} - \theta_{lh}(\tau))\end{aligned} \tag{27}$$

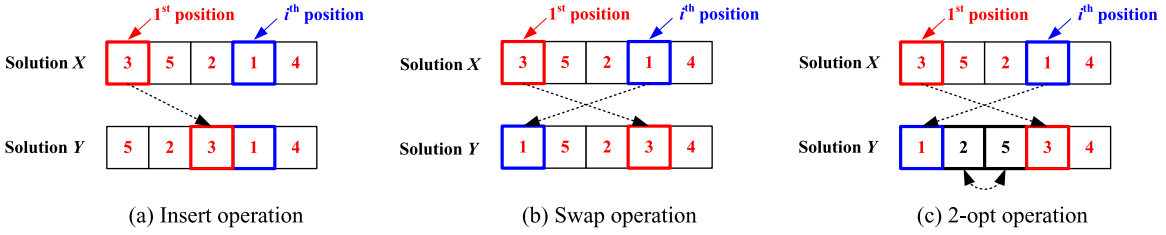
where $w(\tau)$ is the inertia weight in the τ^{th} iteration. $w(1)$ and $w(\Gamma)$ are the inertia weight in the first and the last iteration, respectively. If the next new position of particle reaches more than its the maximum ($\theta^{\max}$) or less than its minimum ($\theta^{\min}$) position value, then the next new position is updated by $\theta_{lh}(\tau+1) = \theta^{\max}$ or $\theta_{lh}(\tau+1) = \theta^{\min}$, respectively. The process of GLNPSO is repeated until the maximum number of iterations (Γ) is reached.

***Step 4:* Re-initialization and Re-order**

During training, if the maximum number of re-initialization iterations of particles reaches (RI_{max}), the Re-initialization solution will be performed to increase the ability to avoid being trapped in a local optimal solution by randomly generating a new solution in the second part of all particles.

In the case of the maximum number of re-order's iteration of chick ordering reaches (RO_{max}), the Re-order of chick ordering operation will be performed to find the new solution of chick ordering. We consider two strategies, *Shifting order* and *Reverse order*, to determine the chick ordering (the first part of the particle) in order to obtain a new solution by randomly applying the two strategies.

The key concept of the *Shifting order* strategy is to reduce the chick ordering cost. After the *Shifting order* operation is performed to the number of chicks ordered, the strategy in that period is represented



(a) Insert operation (b) Swap operation (c) 2-opt operation

Fig. 8. Example of the neighborhood search solutions.

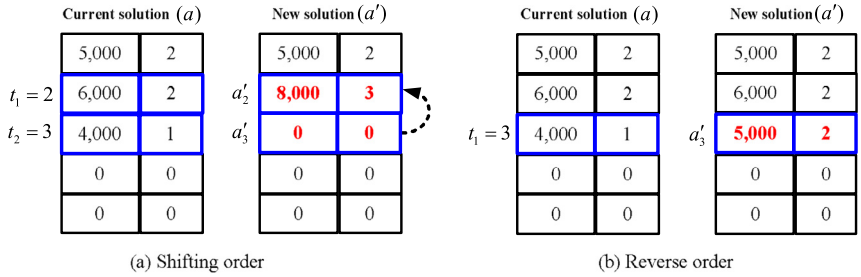


Fig. 9. Example of Re-order strategy.

by Strategy 3. Fig. 9(a) illustrates this strategy. In the first step, the new solution (a') is performed by randomly selecting the first time period with non-zero chicks ordered (t_1) between $t = 1$ to T - 1 from the current solution (a). Then the next time period with non-zero chicks to be ordered (t_2) is selected by incrementing 1 time period. Finally, the new solution is obtained by shifting the number of chicks ordered from period t_2 to period t_1 by the Eq. (28).

$$a'_{t_1} = a_{t_1} + \left(\left\lceil (a_{t_2}/LS) \times (1/(1 + (t_2 - t_1))) \right\rceil \times (LS)\right) \quad (28)$$

Then update the number of chicks to be ordered $a'_{t_2} \leftarrow 0$ and also the Strategies for t_1 and t_2 to 3 and 0, respectively. If t_2 reaches the maximum for the time period ($t_2 = T$) and the number of chicks to be ordered is zero, the new solution is obtained by $a'_{t_1} \leftarrow a_{t_1}$ and also updating the Strategy for t_1 to 3.

For the second strategy, called *Reverse order*, an example of this strategy is illustrated in Fig. 9(b). In the first step, the new solution (a') is generated randomly by selecting the time period with non-zero chicks to be ordered (t_1) between $t = 1$ to T from the current solution (a).Then reverse the number of chicks to be ordered between Strategies 1 and 2. If the number of chicks to be ordered is Strategy 3, the new solution is performed by randomly selecting lot size quantity from either Strategies 1 or 2, and then updating the new solution by that selected Strategy.

The GLNPSO algorithm can be explained step by step as follows:

1. **Input:** problem data and GLNPSO parameters
2. $\tau \leftarrow 1, ri \leftarrow 1, ro \leftarrow 1$
3. **For** $l = 1, 2, \ldots, L$, Generate the lth particle with the initial chick ordering and the random position $\Theta_l(\tau)$ in the range $[\theta^{\min}, \theta^{\max}]$, velocity $\Omega_l = 0$ and personal best $\Psi_l = \Theta_l$ **Endfor**
4. **While** $\tau \leq \Gamma$
5. **For** $l = 1, 2, \ldots, L$, decode $\Theta_l(\tau)$ to a solution R_l and compute the performance measurement of R_l, and set this as the fitness value of Θ_l, represented by $Z(\Theta_l)$ **Endfor**
6. **For** $l = 1, 2, \ldots, L$, update $\Psi_l = \Theta_l$, if $Z(\Theta_l) < Z(\Psi_l)$ and update $\Psi g = \Theta_l$, if $Z(\Psi_l) < Z(\Psi_g)$ **Endfor**
7. **For** $l = 1, 2, \ldots, L$, perform the local search solution on Θ_l, update $\Psi_l = \Theta_l$, if $Z(\Theta_l) < Z(\Psi_l)$ and update $\Psi_g = \Theta_l$, if $Z(\Psi_l) < Z(\Psi_g)$ **Endfor**
8. **For** $l = 1, 2, \ldots, L$, among all *pbest* from G neighbors of the lth particle, set the personal best which obtains the least fitness value to be Ψ_l^L **Endfor**
9. **For** $l = 1, 2, \ldots, L$, and $h = 1, 2, \ldots, H$, set $\psi_{lh}^N = \psi_{oh}$ that maximizes fitness-distance ratio (FDR) for $o = 1, 2, \ldots, L$, where FDR is defined as: $FDR = \frac{Z(\Theta_l) - Z(\Psi_o)}{|\theta_{lh} - \psi_{oh}|}$; $l \neq o$ **Endfor**
10. Update the inertia weight as: $w(\tau) = w(\Gamma) + \frac{\tau - \Gamma}{1 - \Gamma}[w(1) - w(\Gamma)]$
11. Update the velocity and position of each lth particle as:
$\omega_{lh}(\tau + 1) = w(\tau)\omega_{lh}(\tau) + c_p u(\psi_{lh} - \theta_{lh}(\tau)) + c_g u(\psi_{gh} - \theta_{lh}(\tau))$
$+ c_l u(\psi_{lh}^L - \theta_{lh}(\tau)) + c_n u(\psi_{lh}^N - \theta_{lh}(\tau))$
$\theta_{lh}(\tau + 1) = \theta_{lh}(\tau) + \omega_{lh}(\tau + 1)$
12. **If** $\theta_{lh}(\tau + 1) > \theta^{\max}$ **Then** $\theta_{lh}(\tau + 1) = \theta^{\max}$; $\omega_{lh}(\tau + 1) = 0$ **Endif**
13. **If** $\theta_{lh}(\tau + 1) < \theta^{\min}$ **Then** $\theta_{lh}(\tau + 1) = \theta^{\min}$; $\omega_{lh}(\tau + 1) = 0$ **Endif**
14. **If** $Z(\Psi_g(\tau)) \geq Z(\Psi_g(\tau - 1))$ **Then** $ri \leftarrow ri + 1, ro \leftarrow ro + 1$ **Endif**
15. **If** $ri = RI_{max}$ **Then** Generate the lth particle with random position $\Theta_l(\tau)$ in the range $[\theta^{\min}, \theta^{\max}]$, velocity $\Omega_l = 0$ and personal best $\Psi_l = \Theta_l$ for $l = 1, 2, \ldots, L$ and set $ri \leftarrow 1$ **Endif**
16. **If** $ro = RO_{max}$ **Then** Re-order operation to chick ordering and set $ro \leftarrow 1$ **Endif**
17. $\tau \leftarrow \tau + 1$ **Endwhile**
18. **Output:** R_l and $Z(\Theta_l)$

Table 2
12 test problems with different sizes.

Problem number	Problem size			
	# of pullet houses	# of hen houses	Total capacity of slaughterhouses (hens)	# of time periods
1	5	8	10,000	5
2	5	8	20,000	5
3	8	15	10,000	5
4	8	15	20,000	5
5	5	8	10,000	10
6	5	8	20,000	10
7	8	15	10,000	10
8	8	15	20,000	10
9	15	25	30,000	20
10	15	25	30,000	30
11	20	40	40,000	20
12	20	40	40,000	30

5. Computational experiments

5.1. Case study application

This research was motivated by the interest of a semi-industrial production firm in northeastern Thailand. Currently, the total number of laying hens in its farms is about 1,900,000 hens with a daily production of 1,500,000 eggs. Presently, the weekly chicks order volumes from hatcheries of the case-study company depend on egg demand volumes in the planning horizon. Then the chicks are transferred to the contracted pullet-raising farms. When the pullets are 17 weeks old, they are moved to a laying unit (i.e. hen houses). To transport the hens, the firm has a contract with a third-party transport provider. These hens will remain in the hen house until the age of 75 weeks for egg production. After that, they are slaughtered at the firm's slaughter house.

To allocate chicken to pullet houses and hen houses, two major factors are considered, namely, sizes of the pullet houses and distance between the pullet houses and the hen houses. Currently, the company gives priority to the largest pullet house size to minimize the total number of houses used. To allocate the hens, the firm gives priority to the shortest distance between the pullet houses and the hen houses in order to minimize the traveling distance of the truck. If there are several hen houses with the same distance, the hen house will be selected in the order of larger size.

5.2. Test problems and performance measures

In this section, the performance of GLNPSO in the hen egg production planning problem was validated by comparing the optimal solution obtained by the MPL/CPLEX software and the traditional PSO. The performance of the proposed methods in terms of the total cost was tested by using 12 randomly generated test instances. The parameters for each instance are given in Table 2. To solve the problem, the traditional PSO and the GLNPSO algorithms were developed using Matlab

Table 3
The setting of GLNPSO control parameters.

Problem number	Γ	L	RI_{max}	RO_{max}
1	300	100	15	30
2	300	100	15	30
3	300	200	15	30
4	300	200	15	30
5	400	200	20	40
6	400	200	20	40
7	500	300	20	40
8	500	300	20	40
9	600	300	20	40
10	600	300	20	40
11	600	300	20	40
12	600	300	20	40

version 7.9.0.529 on Intel® Core™ i5 processor (2.4 gigahertz), with 3.00 gigabytes of RAM.

For each problem number, the values of weekly order volumes on egg demand in the planning horizon are approximately 50,000–275,000 eggs per week. The chicks are purchased by applying lot size discipline. The number of chicks per lot size (LS) is 1,000 chicks and the pullet-raising farms have heterogeneous capacities ranging from 7,000–14,000 pullets per house. When the pullets are two weeks old ($K = 2$), they are moved to a laying unit (i.e., hen houses) within a radius of 2–30 kilometers from the pullets farms. These hens will remain in the hen house until the age of five weeks ($R = 3$) for egg production with the production rate of eggs from laying hens for all ages being seven eggs per week ($P_1, P_2, P_3 = 7$). The capacity of the hen houses ranges from 4,000–15,000 laying hens per house. Moving pullets or hens from the houses means complete removal. Then the pullet houses and hen houses are cleaned within one ($C^P = 1$) and two ($C^H = 2$) weeks, respectively.

To solve the problems, six major factors of cost are determined: (1) chick ordering cost is 1,000 units per order ($A = 1,000$); (2) feed costs of pullets and hens are one per chicken per week ($F^P, F^H = 1$); (3) farm utilization cost of pullet house and hen house is 1,000 units per house per week ($O^P, O^H = 1,000$); (4) holing cost of eggs is one unit per egg per week ($ES = 1$); (5) transportation cost of pullets from pullet house to hen house depending on transportation distance is one unit per kilometer ($TRC = 1$) and (6) egg demand shortage cost is five units per egg ($LC = 5$).

In order to determine the suitable control parameters of the proposed algorithm (i.e., GLNPSO), the values of these parameters were initially set with respect to the empirical schemes from the study of Ai and Kachitvichyanukul (2007). After performing the experiments, parameter are set as $G = 5$, $c_p = 1$, $c_g = 1$, $c_l = 1$, $c_n = 1$ and the inertia weight(w) is set to linearly decrease from 0.9 to 0.4. Additionally, other control parameters used in the GLNPSO algorithm are presented in Table 3.

In this paper, two quantities are investigated: (1) the performance of the proposed heuristic algorithm, obtained by comparing its solutions to the optimal solution, and (2) the relative improvement of the solutions obtained by the firm's current practice with respect to those of traditional PSO and the GLNPSO algorithms.

To evaluate the proposed heuristic algorithms, two performance measures are used: (1) solution quality, and (2) computational speed. The computational speed of the proposed algorithms is measured by the amount of CPU time required to execute the algorithms. The CPU time is reported in seconds. The quality of a solution generated by the proposed heuristic algorithms (traditional PSO and GLNPSO) is measured in terms of their performance (P) as presented below:

let

$$P = (Sol_{opt}/Sol_{heu}) \times 100 \tag{29}$$

where

P = the proposed heuristic performance (percent)
Sol_{opt} = the optimal solution
Sol_{heu} = the solution obtained from the proposed heuristic algorithms

Moreover, a comparison between the firm's current practice and the proposed heuristic algorithms is also performed. The relative improvement (RI) of the solutions obtained from the current practice after applying the proposed heuristic algorithms is evaluated and presented below:

let

$$RI = ((Sol_{cur} - Sol_{heu})/Sol_{cur}) \times 100 \tag{30}$$

where

RI = the relative improvement (percent) between Sol_{cur} and Sol_{heu}
Sol_{cur} = the solution obtained from the current practice
Sol_{heu} = the solution obtained from the proposed heuristic algorithms

5.3. Computational results

The following Table 4 shows the solution quality in terms of the total cost of the optimal solutions with those of the firm's current practice and the proposed heuristic algorithms (traditional PSO and

Table 4
Computation results of the total cost for each problem number.

Problem number	MPL/CPLEX	Current practice	Traditional PSO					GLNPSO				
			Min. solution	Average solution	Max. solution	Standard deviation	# of optimal solution found	Min. solution	Average solution	Max. solution	Standard deviation	# of optimal solution found
1	238,000	368,000	245,000	245,600	246,000	516	0	238,000	238,300	239,000	483	7
2	238,000	368,000	245,000	246,000	250,000	1,491	0	238,000	238,200	239,000	422	8
3	370,000	495,000	372,000	373,100	377,000	1,663	0	370,000	371,200	372,000	1,033	4
4	352,000	389,000	352,000	353,200	356,000	1,932	7	352,000	352,400	356,000	1,265	9
5	744,000	1,016,000	802,000	809,600	819,000	5,910	0	751,000	755,500	759,000	3,064	0
6	669,000	1,048,000	687,000	694,700	705,000	6,413	0	669,000	676,500	681,000	3,951	1
7	1,074,000[a]	1,747,000	1,216,000	1,221,200	1,228,000	4,467	0	1,149,000	1,158,300	1,170,000	7,543	0
8	1,042,000	1,472,000	1,130,000	1,185,200	1,219,000	41,141	0	1,078,000	1,085,500	1,095,000	5,622	0
9	[b]	2,782,000	1,953,000	1,989,300	2,027,000	20,128	-	1,554,000	1,661,900	1,752,000	63,808	-
10	[b]	4,196,000	2,776,000	2,885,500	2,944,000	51,759	-	2,539,000	2,719,600	2,822,000	83,602	-
11	[b]	3,247,000	2,021,000	2,064,900	2,106,000	27,954	-	1,896,000	1,932,600	1,978,000	28,907	-
12	[b]	4,674,000	3,086,000	3,130,400	3,207,000	45,848	-	2,834,000	2,891,500	2,942,000	36,613	-
Total							7					29

[a] = MPL/CPLEX run out of memory.
[b] = Problem is too large to be solved by MPL/CPLEX.

Table 5
Computation result of the performance and the relative improvement of the solutions from the traditional PSO and the GLNPSO algorithm for each problem number.

Problem number	Traditional PSO				GLNPSO			
	Best performance (percent)	Mean performance (percent)	Worst performance (percent)	*RI* (percent)	Best performance (percent)	Mean performance (percent)	Worst performance (percent)	*RI* (percent)
1	97.14	96.91	96.75	33.26	100.00	99.87	99.58	35.24
2	97.14	96.75	95.20	33.15	100.00	99.92	99.58	35.27
3	99.46	99.17	98.14	24.63	100.00	99.68	99.46	25.01
4	100.00	99.66	98.88	9.20	100.00	99.89	98.88	9.41
5	92.77	91.90	90.84	20.31	99.07	98.48	98.02	25.64
6	97.38	96.30	94.89	33.71	100.00	98.89	98.24	35.45
7	88.32	87.95	87.46	30.10	93.47	92.72	91.79	33.70
8	92.21	87.92	85.48	19.48	96.66	95.99	95.16	26.26
9	-	-	-	28.49	-	-	-	40.26
10	-	-	-	31.23	-	-	-	35.19
11	-	-	-	36.41	-	-	-	40.08
12	-	-	-	33.03	-	-	-	38.14

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	MPL_CPLEX - Traditional_PSO	-50200.00000	62232.76583	22002.60537	-102227.8942	1827.89424	-2.282	7	.057
Pair 2	MPL_CPLEX - GLNPSO	-18612.50000	30278.63167	10705.11289	-43926.06955	6701.06955	-1.739	7	.126

Fig. 10. Paired samples test of the total cost between MPL/CPLEX and the proposed heuristic algorithms (traditional PSO and GLNPSO).

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	Traditional_PSO - GLNPSO	31587.50000	36431.47923	12880.47300	1130.02116	62044.97884	2.452	7	.044

Fig. 11. Paired samples test of the total cost between traditional PSO and GLNPSO algorithms.

GLNPSO) obtained on 10 runs for each instance. The standard deviation indicates the stable performance of the proposed heuristic algorithms during the 10 executions and is also presented in Table 4. From this table, we can see that the GLNPSO yields low standard deviation for small sized instances (Problem 1–8) since the planning horizon is short resulting to no various solutions. Therefore, the solution obtained in each replication is very close to the optimal solution. On the other hand, the traditional PSO yields higher standard deviation since the solution obtained in each replication is scattered due to premature convergence of local minima. Considering large size problems (Problems 9–12), since the planning horizon is very long, there are various solution in search space. Hence, in some replications, the solutions obtained are close to optimal solutions, while the other swarms move far from the optimal ones. For the traditional PSO, it yields lower standard deviation since it has not included a local search algorithm (re-initialization and re-ordering strategies) to diversify solutions in search space.

The solution with symbol "[a]" means the best feasible solution given by the MPL/CPLEX, which is an estimated value since the CPLEX optimizer ran out of memory and the solution with symbol "[b]" means the optimal solution cannot be obtained by the MPL/CPLEX since the problem is too large to be solved by MPL/CPLEX. It can be seen that the proposed GLNPSO algorithm can find the optimal solution for 5 out of the 8 problem numbers, and 29 out of 80 replicated runs. Based on these results, the performance of the heuristic algorithm (P) of the total cost and the relative improvement (RI) of the solutions obtained from the current practice after applying the traditional PSO and the GLNPSO algorithms are presented in Table 5. From this table, we can observe that the performance of the heuristic algorithm (P) of the total cost ranges from 92.72 to 99.89 percent, with an average of 98.18 percent for the GLNPSO algorithm, and 87.92 to 99.66 percent, with an average of 94.57 percent for the traditional PSO algorithm.

To evaluate the performance on the total cost of the traditional PSO and the GLNPSO algorithms, the analysis was carried out using SPSS Software V14 for Windows. Fig. 10 shows the paired samples tests of MPL/CPLEX performed to compare the total cost of the traditional PSO and the GLNPSO algorithms. Results of the tests indicate that the solutions obtained from both algorithms were not significantly different at $\alpha = 0.05$ compared to those obtained from the mathematical model. However, there are significant differences on the total cost of the traditional PSO algorithm compared with the GLNPSO algorithms (see Fig. 11). For the relative improvement (RI) which compares the total cost of the solutions obtained from the current practice after applying the traditional PSO and the GLNPSO algorithms, the results show that the GLNPSO algorithm provides better total cost value than the firm's current practice, ranging from 9.41 to 40.48 percent (with an average of 31.67 percent) and the traditional PSO algorithm provides better total cost value than the firm's current practice, ranging from 9.20 to 36.41 percent (with an average of 27.75 percent).

The minimal, average and maximal values of the computation speeds (CPU time) of all problem numbers are shown in Table 6. From this table, with the growth of problem size, the MPL/CPEX approach cannot solve the large size problems due to the memory restriction. Fig. 12 shows that the CPU time of MPL/CPLEX expands rapidly with the scale of the problem size, and the CPU times for the traditional PSO and the GLNPSO are lower than that of MPL/CPLEX. This means that the GLNPSO can save computation time and also has more opportunity to get better solutions. GLNPSO also provides slightly lower total cost values than the traditional PSO algorithm. The convergence behaviors of the GLNPSO and the traditional PSO methods

Table 6
Comparison of the computation speeds between MPL/CPLEX and the traditional PSO and the GLNPSO algorithm for each problem number.

Problem number	MPL/CPLEX (Sec ond)	Traditional PSO			GLNPSO		
		Min. CPU time (Second)	Average CPU time (Second)	Max. CPU time (Second)	Min. CPU time (Second)	Average CPU time (Second)	Max. CPU time (Second)
1	3	35	36	40	38	40	43
2	4	38	40	42	39	42	46
3	7	130	133	147	133	140	145
4	6	130	135	141	134	146	147
5	55	162	164	165	170	181	185
6	22	165	174	181	169	176	191
7	>9,715[a]	541	554	561	555	566	589
8	19,100	529	538	578	542	554	577
9	[b]	1,461	1,467	1,475	1,495	1,497	1,505
10	[b]	2,061	2,074	2,083	2,096	2,111	2,143
11	[b]	2,521	2,558	2,571	2,584	2,607	2,613
12	[b]	3,308	3,318	3,340	3,335	3,343	3,351

[a] = MPL/CPLEX run out of memory.
[b] = Problem is too large to be solved by MPL/CPLEX.

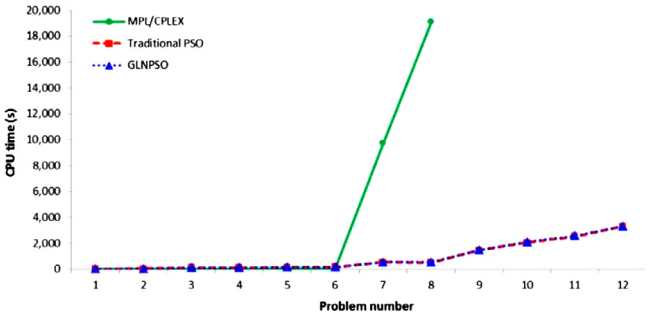


Fig. 12. The comparison graph of CPU time between MPL/CPLEX, Traditional PSO and GLNPSO.

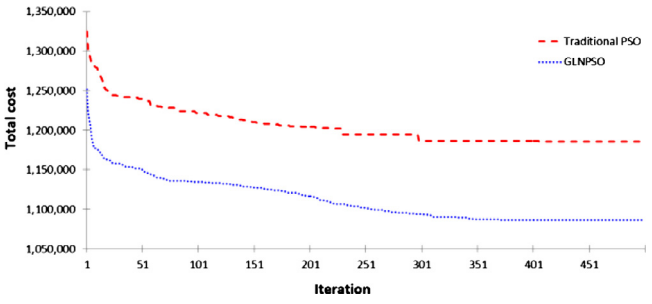


Fig. 13. The average objective function of the iterations for problem number 8.

were tested. Fig. 13, for instance, shows the convergence behaviors of the GLNPSO and the traditional PSO methods in the iteration of solutions for problem number 8. The results indicate that the proposed GLNPSO yields better solutions than those of the traditional PSO for all problem instances, especially for large-sized problems.

6. Conclusions and discussion

This paper has dealt with an integrated model for multi-level production lot-sizing and scheduling in order to minimize the total cost. A heuristic solution procedure based on the GLNPSO algorithm was developed. The local search procedure with neighborhood search solutions including insert operation, swap operation, and 2-opt operation were applied to speed up the algorithm. In order to avoid being trapped into a local optimum, the Re-initialization and the Re-order strategy were used to improve the possibility of finding an optimal solution in the searching space.

The proposed heuristic performance (P) was used to measure the performance of the proposed algorithms by comparing their solutions to the optimal solution in terms of the total cost. The experimental results showed that the GLNPSO algorithm obtained a near-optimal solution ranging from 92.72 to 99.89 percent, with an average of 98.18 percent, and with a stable performance as indicated by the standard deviation values. From the numerical experiments, it was found that the proposed GLNPSO method yielded the optimal solution, especially in the small-size problems (problem numbers 1 to 4) ranging between 4–9 runs out of 10 replicated runs. The statistical results obtained from paired sample tests performed to compare the total cost of the traditional PSO and the GLNPSO algorithms show that the proposed GLNPSO was significantly better than the traditional PSO. Moreover, the computational time of the GLNPSO algorithm is lower than that of the CPLEX optimizer, especially for the large-size problem.

Additionally, the performance of the heuristic algorithm was measured using the relative improvement (RI), which compared the total cost of the solutions obtained from the firm's current practice after applying the traditional PSO and the GLNPSO algorithms. The results obtained from this study show that the GLNPSO algorithm provided better total cost values than the firm's current practice, ranging from 9.41 to 40.48 percent (with an average of 31.67 percent), and 0.23 to 16.46 percent (with an average of 5.50 percent) better than the traditional PSO algorithm. The proposed GLNPSO also has convergent behavior and yields better solutions than the traditional PSO method for all test instances. The mathematical and heuristic models developed in this paper are easily adaptable and should prove to be beneficial to other poultry industries and other similar agro-food sectors in Thailand and around the world.

However, there is still much opportunity to extend our work in many aspects. To reduce farm utilization cost, mixing of chickens with different ages in the same hen house should be allowed with a certain maximum age difference. Another valuable avenue for future research is to consider some other parameters of the problem as fuzzy variables. Additionally, although the proposed GLNPSO algorithm has shown an outstanding ability to solve the problem at hand, there is a possibility to use hybrid methods

Acknowledgments

This work was supported by the Higher Education Research Promotion and National Research University Project of Thailand, Office of the Higher Education Commission, through the Food and Functional Food Research Cluster of Khon Kaen University and collaboration with the Research Unit on System Modeling for Industry, Khon Kaen University, Thailand.

References

Ai, T. J., & Kachitvichyanukul, V. (2007). Dispersion and velocity indices for observing dynamic behavior of particle swarm optimization. In *Proceeding of the IEEE Congress on Evolutionary Computation* (pp. 3264–3271).

Berretta, R., & Rodrigues, L. F. (2004). A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics, 87*(1), 67–81.

Clark, A., Almada-Lobo, B., & Almeder, C. (2011). Lot sizing and scheduling: Industrial extensions and research opportunities. *International Journal of Production Research, 49*(9), 2457–2461.

Clerc, M. (2006). *Particle Swarm Optimization*. London: ISTE.

da Silva, A. F., Marins, F. A. S., & Montevechi, J. A. B. (2013). Multi-choice mixed integer goal programming optimization for real problems in a sugar and ethanol milling company. *Applied Mathematical Modelling, 37*, 6146–6162.

Drexl, A., & Kimms, A. (1997). Lot sizing and scheduling - survey and extensions. *European Journal of Operational Researc, 99*(2), 221–235.

Dye, C. Y., & Hsieh, T. P. (2010). A particle swarm optimization for solving joint pricing and lot-sizing problem with flu ctuating demand and unit purchasing cost. *Computers & Mathematics with Applications, 60*(7), 1895–1907.

Dye, C. Y., & Ouyang, L. Y. (2011). A particle swarm optimization for solving joint pricing and lot-sizing problems with fluctuating demand and trade credit financing. *Computers & Industrial Engineering, 60*(1), 127–137.

Franck, B., Neumann, K., & Schwindt, C. (1997). A capacity-oriented hierarchical approach to single-item and small-batch production planning using project-scheduling method. *Operations Research-Spektrum, 19*, 77–85.

Gribkovskaia, I., Gullberg, B. O., Hovden, K. J., & Wallace, S. W. (2006). Optimization model for a livestock collection problem. *International Journal of Physical Distribution & Logistics Management, 36*(2), 136–152.

Guimarães, L., Klabjan, D., & Almada-Lobo, B. (2013). Pricing, relaxing and fixing under lot sizing and scheduling. *European Journal of Operational Research, 230*(2), 399–411.

Harris, F. W. (1913). How many parts to make at once factory. *The Magazine of Management, 10*(2), 135–136.

Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research, 177*, 1855–1875.

Jans, R., & Degraeve, Z. (2008). Modeling industrial lot sizing problems: A review. *International Journal of Production Research, 46*(6), 1619–1643.

Karimi, B., Fatemi Ghomi, S. M. T., & Wilson, J. M. (2003). The capacitated lot sizing problem: A review of models and algorithms. *Omega, 31*, 365–378.

Karimi-Nasab, M., & Seyedhoseini, S. M. (2013). Multi-level lot sizing and job shop scheduling with compressible process times: A cutting plane approach. *European Journal of Operational Research, 231*(3), 598–616.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the Fourth IEEE International Conference on Neural Networks* (pp. 1942–1948).

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm Intelligence*. San Francisco: Morgan Kaufmann Publishers.

Kopanos, G. M., Puigjaner, L., & Georgiadis, M. C. (2010). Optimal production scheduling and lot-sizing in dairy plants: The yogurt production line. *Industrial & Engineering Chemistry Research, 49*(2), 701–718.

Kopanos, G. M., Puigjaner, L., & Georgiadis, M. C. (2011). Resource-constrained production planning in semicontinuous food industries. *Computers & Chemical Engineering, 35*(12), 2929–2944.

Kopanos, G. M., Puigjaner, L., Georgiadis, M. C., & Bongers, P. M. M. (2011). An efficient mathematical framework for detailed production scheduling in food industries: The Ice-Cream Production line. *Computer Aided Chemical Engineering, 29*, 960–964.

Lin, T. D., Hsu, C. C., & Hsu, L. F. (2013). Optimization by ant colony hybrid local search for online class constrained bin packing problem. *Applied Mechanics and Materials, 311*, 123–128.

Mahdieh, M., Bijari, M., & Clark, A. (2011). Simultaneous lot sizing and scheduling in a flexible flow line. *Journal of Industry and Systems Engineering, 5*(2), 107–119.

Nearchou, A. C. (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics, 129*, 242–250.

Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming*. New York: Springer.

Pongchairerks, P., & Kachitvichyanukul, V. (2005). A non-homogenous particle swarm optimization with multiple social structures. In *Proceedings of the International Conference on Simulation and Modeling*.

Pongchairerks, P., & Kachitvichyanukul, V. (2006). Particle swarm optimization algorithm with multiple social structures. In *Proceeding of the 36th Conference on Computer and Industrial Engineering* (pp. 1556–1567).

Pratchayaborirak, T., & Kachitvichyanukul, V. (2007). A Two-Stage Particle Swarm Optimization for multi-objective job shop scheduling problems. In *Proceedings of the APIEMS 2007 Conference*.

Ramezanian, R., Saidi-Mehrabad, M., & Fattahi, P. (2013). MIP formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints. *Journal of Manufacturing Systems, 32*(2), 392–401.

Seeanner, F., Almada-Lobo, B., & Meyr, H. (2013). Combining the principles of variable neighborhood decomposition search and the fix & optimize heuristic to solve multi-level lot-sizing and scheduling problems. *Computers & Operations Research, 40*, 303–317.

Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 69–73).

Song, S., Kong, L., Gan, Y., & Su, R. (2008). Hybrid particle swarm cooperative optimization algorithm and its application to MBC in alumina production. *Progress in Natural Science, 18*(11), 1423–1428.

Stockmeyer, L. J., & Meyer, A. R. (2002). Cosmological lower bound on the circuit complexity of a small problem in logic. *Journal of the ACM, 49*(6), 753–784.

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research, 177*(3), 1930–1947.

Toledo, C. F. M., da Silva Arantes, M., de Oliveira, R. R. R., & Almada-Lobo, B. (2013). Glass container production scheduling through hybrid multi-population based evolutionary algorithm. *Applied Soft Computing, 13*, 1352–1364.

Toledo, C. F. M., de Oliveira, R. R. R., & Franca, P. M. (2013). A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers & Operations Research, 40*, 910–919.

Toledo, C. F. M., de Oliveira, L., de Freitas Pereira, R., França, P. M., & Morabito, R. (2014). A genetic algorithm/mathematical programming approach to solve a two-level soft drink production problem. *Computers & Operations Research, 48*, 40–52.

Toledo, F. M. B., & Armentano, V. A. (2006). A Lagrangean-based heuristic for the capacitated lot-sizing problem in parallel machines. *European Journal of Operational Research, 175*, 1070–1083.

van Hoesel, C. P. M., & Wagelmans, A. P. M. (2001). Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Mathematics of Operations Research, 26*(2), 339–357.

Veeramachaneni, K., Peram, T., Mohan, C., & Osadciw, L. A. (2003). Optimization Using Particle Swarms with Near Neighbor Interactions. *Lecture Notes in Computer Science, 2723*, 110–121.

Wagner, H. M., & Whitin, T. M. (1958). A dynamic version of the economic lot size model. *Management Science, 5*(1), 89–96.

Wang, A. C., & Yeh, M. F. (2014). A modified particle swarm optimization for aggregate production planning. *Expert Systems with Applications, 41*, 3069–3077.

Wilson, R. H. (1934). A scientific routine for stock control. *Harvard Business Review, 13*(1), 116–128.

Xiao, Y., Zhang, R., Zhao, Q., Kaku, I., & Xu, Y. (2014). A variable neighborhood search with an effective local search for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research, 235*, 102–114.

Yu, R., Leung, P., & Bienfang, P. (2006). Optimal production schedule in commercial shrimp culture. *Aquaculture, 254*, 426–441.

Yu, V. F., Lin, S. W., Lee, W., & Ting, C. J. (2010). A simulated annealing heuristic for the capacitated location routing problem. *Computer & Industrial Engineering, 58*, 288–299.