INTEGRATED ORDER ACCEPTANCE AND SCHEDULING DECISION MAKING IN PRODUCT SERVICE SUPPLY CHAIN WITH HARD TIME WINDOWS CONSTRAINTS

BIN DAN*,a,b, HUALI GAOa,c, YANG ZHANGa,b
RU LIUa,b AND SONGXUAN MAa,b

aSchool of Economics and Business Administration
Chongqing University, Chongqing 400044, China
bResearch Center of Business Administration & Economic Development
Chongqing University, Chongqing 400030, China
cSchool of Management, Southwest University of Political Science & Law
Chongqing 401120, China

Abstract. A product service supply chain (PSSC) supplies customers with product-service systems (PSS) consist of integrated products and services. The product manufacturing should match the service supply in the order delivery planning. For PSS orders are usually delivered under time window constraints, this paper is concerned with the integrated order acceptance and scheduling (OAS) decision of the PSSC. Defined the PSS orders by their revenues, product processing times, serving offering times and hard time window constraints, we formulate the OAS problem as a MILP model to optimize total revenue of PSSC and propose two effective value for big-M to solve the problem with small size optimally. The simulated annealing algorithm based on the priority rule of servable orders first (SOF-SA) and the dynamic acceptance and scheduling heuristic (DASH) algorithm are presented. The performance of the model and the two algorithms are proved through simulating instances with different order sizes. Computational tests show that the SOF-SA algorithm is more effective when used for small size problems while the DASH algorithm is more effective for problems with larger size; negotiating with customers to make reasonable delivery time windows should be beneficial to increasing total revenue and improving the decision efficiency.

1. Introduction. After the importance of product-service systems (PSS) in the sustainable development was reported by the United Nations Environment Programme at the beginning of this century [23], researches and discussions on PSS are becoming important issues in both the industrial and academic domains. Accordingly, the supply chain mode is changed from physical product supply chain or service supply chain to product service supply chain (PSSC) which supplies the integration of products and services to customers. A PSSC is a supply chain network where the system integrator acts as the core enterprise, providing customized PSS

2010 Mathematics Subject Classification. Primary: 49K30, 49K35; Secondary: 90C11.

Key words and phrases. Product service supply chain, product-service systems, order acceptance and scheduling, hard time windows, simulated annealing algorithm.

This research was supported by the National Natural Science Foundation of China (Grant Number: 71272086), the National Science and Technology supporting Program of China (Grant Number: 2015BAF05B01), and the Specialized Research Fund for the Doctoral Program of Higher Education of China (20120191110042).

* Corresponding author: danbin@cqu.edu.cn (B Dan).
for customers, controlling the product flow, service flow, information flow, capital flow and value flow systematically, implementing customer participation and mutual cooperation of all member enterprises (the product supplier, the system integrator and the service supplier) [13]. It usually is a build-to-order system [1]. Restricted by the cost, resource, the ability to customize and other factors, neither the production capacity nor the service capacity of the PSSC can be infinite. In addition, the service offering in PSS orders is quite different from the production of corporeal products. Since services are immaterial and cannot be stored, the relevant offering must be strictly JIT. The supply and delivery of services must be caught on simultaneously, i.e. services cannot be fulfilled ahead to wait for delivery. So the PSS order delivery usually faces time window constraints quoted by customers. The service offering cannot have an early start or a delay completion. It may be impossible to meet all order requirements when the PSSC is confronted with multiple customer PSS orders. Therefore, how to select and schedule the PSS orders arriving, in order to make full use of both the production capacity and service capacity of the PSSC and maximize its total revenue has become an urgent problem in the operation management area.

This study deals with the integrated order acceptance and scheduling (OAS) problem of a PSSC that consists of a system integrator, a product supplier and a service supplier. The production and the service offering of the PSSC are initiated by customers' PSS orders. Take an elevator PSSC composed of an elevator manufacturer and an elevator installation enterprise for example, the elevator manufacturer acts as the system integrator and produces elevators for customers, and then the elevator installation enterprise installs elevators for customers. Each order contains only one type of elevators and the corresponding installation service with different characteristics in terms of rated load, rated speed, shape, the total height of the travel, and so on. It is known that producing an elevator and installing it for the customer are both time consuming. The installing service offering and its delivery must be carried on simultaneously. Consequently, the installing process is usually limited by a time window, and any installation of an order beyond its time window may not be allowed, e.g., customers may request installing and debugging their elevator during their idle time. So the elevator PSSC confronts the OAS problem to maximize the total revenue.

Therefore, taking the coordination of the production and service offering into consideration, this paper studies the OAS decision of the system integrator faced with multiple PSS order applications to maximize the total revenue of the PSSC. We structure the rest of this paper as follows. In Section 2, the integrated OAS problem in a PSSC system is formally described. Then the integrated problem is formulated as a mixed-integer linear programming (MILP) model in Section 3. In Section 4, simulated annealing algorithm based on the priority rule of servable orders first (SOF-SA) is developed for solving the problem. In Section 5, a dynamic acceptance and scheduling heuristic (DASH) algorithm is developed. Section 6 presents the details of the data generation, the performances and analysis of the MILP model and the two algorithms. Section 7 presents conclusions and directions for future researches.

2. Literature review. The operation management problems of PSSC have been studied in recent years. For example, Maull et al. [16] focused specifically on the inter-relationship between the products and the services in the PSSC, and developed
an IDEF0 model to highlight the importance of coordinating product and multiple service concepts for integrated PSSC. Xu et al. [28] explored the structure and the unique features of PSSC and developed a comprehensive framework for it. Xie et al. [27] studied how to effectively provide PSS for customers in the supply chain of service-oriented manufacturing with asymmetric information. They developed three forms of contracts and conducted a detailed comparison of the incentives provided by the three contract models. They analyzed the equilibrium and gave some incentive countermeasures of information sharing in the supply chain.

The other relevant researches are mainly on PSS and are mainly the qualitative researches on the origin, concept, connotation and value creation mechanism of PSS. For example, Mont [17] studied the concept, connotation, nature and classification of PSS. They established a theoretical framework for PSS, and analyzed the main factors that had major influence on the feasibility and applicability of the enterprises’ PSS strategies. Their study laid the theoretical foundation for following researches. Manzini and Vezzoli [15] described some examples to underline the importance of PSS strategy, and proposed a general framework to describe the sustainable potentials and to derive the design discipline of the PSS strategy. Kuo [7] constructed a renting system in the reverse logistics environment based on PSS theory and established a simulation model to determine the feasibility of such PSS strategy. The simulation model established was analyzed by using three distinct scheduling and dispatching rules. Garetti et al. [6] discussed a new approach named Life Cycle Simulation to support the design of PSS, identified the common characteristics and prioritized next steps to be done for a comprehensive implementation of the new approach and proposed a reference architecture. Based on the system dynamics theory and the triple bottom line theory, Lee et al. [10] proposed a new approach to effectively measure PSS sustainability which has dynamic and multidimensional characteristics. Beuren et al. [2] presented a literature review on papers of PSS published from 2006 to 2010, discussed specific features, positive and negative issues of PSS, and pointed out that the relevant researches were mainly theoretical work. Sakao et al. [20] adopted an explorative longitudinal in-depth case study and analyzed the existing literature to reveal benefits and risks of integrated product service offering. Hence, additional quantitative researches are required. There is only a small amount of literature applying quantitative methods to PSS researches. For example, Kuo and Ling [8] analyzed different types of maintenance service, discussed the overall utility of integrated maintenance service by using multi-attribute utility analysis, and developed a mathematical model to enable enterprises to determine the optimal maintenance system. Taking additional service capacity and impatient customers into consideration, Li and Jiang [11] studied the modeling and optimization decision of a basic product-service system from the perspective of operation management and modeled the system as a block structure Markov chain to obtain its stationary distribution. Thus, additional quantitative researches on planning and scheduling problems of PSSC are still required.

After decades of development, studies on the selection and scheduling problem of physical product orders become extensive and intensive as the number of build-to-order enterprises increases. Researches on order scheduling problems are mainly focused on time optimization. For example, Zhang et al. [29] studied the order rejection and scheduling decision to minimize the sum of the makespan and the total penalty in single machine environment. Lin and Chong [12] presented a tabu search algorithm with a restart approach to minimize total weighted tardiness for
the scheduling problem where jobs had non-identical due dates in single machine environment. Lee [9] addressed the order scheduling decision to minimize total tardiness in multiple machine environments, taking the orders’ processing time in every stage and due dates into consideration. With the objective to minimize the maximum lateness, Su et al. [21] studied the order scheduling problem where orders are dispatched in batches and scheduled on multiple parallel machines with the orders’ processing time and due dates considered. As both order selection decision and order scheduling decision influence the enterprise’ capacity utilization, some scholars research on OAS synthetically, focusing on the revenue optimization under the influence of tardiness penalty. For example, having defined orders in a single machine environment based on their processing time, sequence dependent setup time, release dates, due dates and deadlines, Oguz et al. [19] formulated the simultaneous OAS decision as a MILP model to maximize total revenue and developed three heuristic algorithms to solve the problem. Nobibon and Leus [18] devised two branch-and-bound algorithms to obtain the optimal solution of the above problem with up to 50 orders within reasonable CPU times. Zhao et al. [30] studied the OAS and due date assignment problem in a single-machine environment. Firstly they considered the problem with common due date assignment and developed a polynomial-time algorithm to solve it. Secondly, they formulated a unified model for the OAS problem with position-dependent processing times. Zhong et al. [31] studied the OAS problem with machine availability constraints. They took the make span of all accepted orders and the penalties of all rejected/outsources orders into account and formulated the problem as a model. The approximability and some important special cases of the model were researched in their work. Xiao et al. [26] formulated the OAS problem in the permutation flow shop environment as an IP model to maximize the total net profit influenced by weighted tardiness penalty, and developed a heuristic algorithm to obtain its near-optimal solutions. Wang et al. [24] formulated the OAS problem in a two machine flow shop as MILP models to maximize the firm’s total net revenue and developed a heuristic algorithm and a branch-and-bound algorithm to solve the problem. And they proposed a modified artificial bee colony algorithm to obtain good solutions for the problem of large size [25]. Later, Esmaeilbeigi et al. [5] formulated two new integer programing formulations for the OAS problem in two machine flow shops which perform better than the formulation developed by Wang et al., and presented several enhancements to speed up the solution procedure.

Although quantitative researches on the operation management problems of traditional physical supply chains have been studied in-depth and maturely, the researches mainly focus on physical product orders and cannot be used to solve the OAS problems in the PSSC. Because every PSS order contains not only physical products but also services that go with the products, and customer participation might be needed during the whole process of order delivery especially when the service is offered, customer requirements have significant influence on the delivery of PSS order. In addition, service offering is usually limited by a time window, namely, the PSSC must offer service to the customer within the time window, and any offering without the time window will not be permitted. Time window constraints increase the difficulty of the planning and sequencing of the PSS orders. Some researchers study the planning and sequencing problems with service offering. For example, considering the time window constraints, Chen et al. [4] formulated production scheduling and vehicle routing for perishable food products as a nonlinear
mathematical model to determine the optimal production quantities, the optimal start time for producing and the optimal vehicle routes that maximize the supplier’s expected total profit. With the objective to minimize the completion time of production, delivery and return, Low et al. [14] formulated the production and vehicle routes problem with time window constraints as a nonlinear mathematical model to obtain the optimal sequence and vehicle routes and designed two heuristic algorithms to solve the large-scale problems. Ullrich [22] studied the production and outbound distribution scheduling simultaneously to minimize total tardiness influenced by processing time, service time and delivery time windows, and designed a genetic algorithm and two classic decomposition approaches to solve the problem.

The service offering in the above researches are mainly transportation services. The transportation services in different orders are not unique, because vehicles are available for different customer orders, i.e. each vehicle can deliver kinds of products to different customers in the same trip. Unlike the relationships between the products and the transportation services above, the products and services of a PSS order need to be matched and integrated. So the planning and scheduling of PSS orders in the PSSC is more complex. In addition, both the production capacity of the product supplier and the service capacity of the service supplier restrict demand satisfaction as both the two capacities are usually limited. When either of the two capacities is overloaded, it may cause problems of capacity waste, cost increase and income reduction. So the matching use of the two capacities should be taken into account when the PSSC makes the planning and scheduling decisions.

3. The problem description and model formulation.

3.1. Problem description. We focus on a PSSC consisting of a system integrator, a product supplier in a single production environment and a service supplier in a single service environment. The integrated system provides various PSSs for customers according to their different demands. The system integrator is the core enterprise of the PSSC, dealing with customers’ needs and connecting the product supplier and the service supplier. The production capacity of the product supplier and service capacity of the service supplier are both limited. At the beginning of each planning period, the system integrator sets production tasks and service tasks respectively to the product supplier and the service supplier according to the PSS orders accepted. Each supplier must accomplish the tasks distributed in accordance with the system integrator’s requirements to ensure all selected PSS orders are delivered on time. The delivery of each PSS order is restricted by a hard time window. Any earliness or tardiness will not be allowed. Additionally, there is no preemption for any order. The PSSC gains revenue if the product production and service offering of an order are both completed. The service offering can begin only after the product production is completed. If the product of an order is completed but the relevant service offering cannot be finished within the time window, the PSSC will not obtain any revenue. And worse yet, the customized product completed may not be suitable for other customers, so the cost will be increased. In addition, the service offering must be strictly JIT for services cannot be stored. So the service offering and the delivery of a PSS order are carried on simultaneously, which may easily cause the backlog of service tasks. As a result, the PSSC must start and complete service offering within the time windows admitted to deliver PSS orders on time. It is quite different from physical products which
can be produced in advance. Hence, when confronted with multiple PSS orders that overpass the capacity of the PSSC in a planning period, the system integrator should make integrated order acceptance and scheduling decision to make full use of the product supplier’s capacity and the service supplier’s capacity, and maximize the total revenue of the whole PSSC. The problem we proposed above is depicted in Fig. 1.

![Graphical representation of the OAS problem in PSSC](image)

**Figure 1.** Graphical representation of the OAS problem in PSSC

In this problem setting, it is assumed that the production capacity and the service capacity of each planning period cannot be transferred to any other planning period. Demand and price of each order are determined when it arrives. A PSS order contains only one product and its corresponding service. The product and service demand in an order must be accepted or rejected as a whole, namely, neither accepting the product in a PSS order only nor accepting the service in a PSS order only are admitted. Each accepted PSS order must be delivered within the time window, i.e., any overload is not admitted. The product production of an order can be started only after that of the preceded order being completed, so does the service offering.

There is an optimal property of the OAS problem in PSSC, which is evident and stated without proof as Proposition 1. And we can use Proposition 1 to validate our formulation.

**Proposition 1.** For the accepted PSS orders, there must be an optimal schedule in which each PSS order is produced and serviced in the same sequence.

### 3.2. Notation.

The symbols used are defined as follows:

- \( N_0 = \{1, 2, ..., n\} \): a set of \( n \) independent PSS orders, available at time zero;
- \( r_i \): the revenue of PSS order \( i \);
- \( t_{pi} \): the processing time of PSS order \( i \);
- \( t_{si} \): the servicing time of PSS order \( i \);
- \( \sigma_m = (o_1, o_2, ..., o_m) \): a sequence of the order subset \( O \), \( m \leq n \), \( O \subseteq N_0 \);
- \( d_{ei} \): the earliest time allowed for service supplier to start the service offering of PSS order \( i \);
- \( d_{li} \): the latest time allowed for service supplier to complete the service offering of PSS order \( i \), \( d_{li} \geq d_{ei} + t_{si} \).

We also define some sets of decision variables to describe the model that will be formulated.
The mathematical model. According to the above assumptions and definitions, the model (named MA) of the integrated PSS order acceptance and scheduling decision with hard time window constraints can be built as follows:

\[
\text{(MA)} \quad \text{Max} \quad \Pi = \sum_{i=1}^{n} y_i r_i
\]

s.t. \( y_i = \begin{cases} 
1, & \text{if PSS order } i \text{ is accepted} \\
0, & \text{otherwise}
\end{cases}, \forall i = 1, 2, ..., n; \)

\[
x_{ij} = \begin{cases} 
1, & \text{if both order } i \text{ and } j \text{ are accepted and order } i \text{ preceds order } j \\
0, & \text{otherwise}
\end{cases}, \quad i, j = 1, ..., n, i \neq j;
\]

\( C_j : \) the completion time of PSS order \( j. \)

3.3. The mathematical model. According to the above assumptions and definitions, the model (named MA) of the integrated PSS order acceptance and scheduling decision with hard time window constraints can be built as follows:

\[
\text{(MA)} \quad \text{Max} \quad \Pi = \sum_{i=1}^{n} y_i r_i \quad (1)
\]

s.t. \[x_{ij} + x_{ji} \leq y_i, \forall i, j = 1, ..., n, i \neq j\] (2)

\[x_{ij} + x_{ji} \geq y_i + y_j - 1, \forall i, j = 1, ..., n, i < j\] (3)

\[C_i + t_{sj} = (x_{ij} - 1)M \leq C_j, \forall i, j = 1, ..., n, i \neq j\] (4)

\[\sum_{i=1, i \neq j}^{n} t_{pi} x_{ij} + (t_{pj} + t_{sj}) y_j \leq C_j, \forall j = 1, ..., n\] (5)

\[C_i \geq y_i (d_{ei} + t_{si}), \forall i = 1, ..., n\] (6)

\[C_i \leq d_{ii}, \forall i = 1, ..., n\] (7)

\[y_i \in \{0, 1\}, \forall i = 1, ..., n\] (8)

\[x_{ij} \in \{0, 1\}, \forall i, j = 1, ..., n, i \neq j\] (9)

In the above model, formula (1) defines the objective function which represents total revenue of all the selected orders. Constraint sets (2) imply that, each PSS order \( i \) is accepted if it is preceded or followed by another PSS order \( j \). Constraint sets (3) ensure that if PSS order \( i \) and PSS order \( j \) are both accepted, then order \( j \) is produced and serviced after order \( i \) or vice versa. Constraint sets (4) imply that if order \( j \) is preceded by order \( i \), the completion time of order \( j \) should not be shorter than that of order \( i \) plus the servicing time of order \( j \). If order \( i \) does not precede order \( j \) in the sequence, \( 0 \leq C_j \) should be the only restriction. Constraint sets (5) imply that if order \( j \) is accepted, then its completion time is no shorter than the sum of the processing times of orders preceding order \( j \) plus the processing time and servicing time of \( j \). Constraint sets (6) imply that the service offering of each PSS order accepted must be started later than the earliest time allowed, and constraint sets (7) imply that the service offering of each PSS order accepted must be finished earlier than the latest time allowed, i.e. each PSS order accepted cannot beyond its due window. Constraint sets (6) - (9) specify the domains of decision variables. In the model, \( M \) is a sufficiently large positive number.

It can be seen from Esmaeilbeigi et al. [5] that setting \( M = \sum_{i=1}^{n} (t_{pi} + t_{si}) \) is a simple way, and compute smaller value for \( M \) may improve the performance of the formulation. Because the completion time of each PSS order \( i \) must be no later than \( d_{ii} \), so we can replace \( M \) in constraints (4) with \( \widetilde{M}_{ij} = d_{ii} + t_{sj} \), where \( i, j = 1, ..., n \), and \( i \neq j \). \( \widetilde{M}_{ij} = d_{ii} + t_{sj} \) are obviously suitable values since the new constraints getting from the replacement hold for each PSS order, \( i, j = 1, ..., n \), and \( i \neq j \).

3.4. The complexity analysis. The size complexity of MA formulated for the OAS problem with \( n \) PSS orders can be seen from the following parameters, the number of binary variables is \( n^2 \), the number of continuous variables is \( n \), the number of constraints is \( 2.5n^2 + 0.5n \), and the number of disjunctive constraints is
So the size complexity of MA model is \( o(n^2) \). And our computational results show that MA formulation can be solved by lingo software to obtain the optimal solution for instances with small number of PSS orders in reasonable times.

An NP-complete problem named (Partition Problem) should be mentioned firstly to prove the NP-hardness of the integrated OAS problem above, which can be described as follows: given a set of \( n \) (an even number) positive numbers \( \{c_1, c_2, ..., c_n\} \) with \( \sum_{i=1}^{n} c_i = T \), find a subset \( |S| = n/2 \) of these numbers such that \( \sum_{i \in S} c_i = T/2 \).

When we set \( N_0 = \{1, 2, ..., n\} \), \( r_i = 1, \ t_{pi} = 0, \ t_{si} = c_i, \ d_{ei} = 0, \ d_{li} = T/2 \), \( \forall i = 1, 2, ..., n \), then the (Partition Problem) is true if and only if the corresponding OAS problem has an optimal solution \( n/2 \). That is to say, the (Partition Problem) can be reduced as a special case of the integrated OAS problem we studied. So the integrated OAS problem is also NP-hard, which cannot be solved in polynomial time unless \( P = NP \).

As there is no exact algorithm for obtaining the optimal solution of the above problem, developing an effective approximate algorithm is the key point. Hence, we develop two heuristic algorithms for the above OAS problem and present them below.

We defined two dummy orders named order 0 and order \( n+1 \) for the algorithm design [19]. And \( r_0 = \max_{i=1, ..., n} \{r_i\}, \ r_{n+1} = \min_{i=1, ..., n} \{r_i\}, \ t_{p0} = t_{s0} = t_{p,n+1} = t_{s,n+1} = 0, \ d_{e0} = d_{l0} = 0, \ d_{e,n+1} = d_{l,n+1} = \max_{i=1, ..., n} (d_{li}) \), it means that order 0 and order \( n+1 \) are respectively assigned to the first position and the last position in any order sequence.

4. The SOF-SA algorithm. In order to greatly reduce the number of iterations, we can develop some reasonable priority rules according to the features of scheduling. Motivated by this, we propose a priority rule called “servable orders first” (SOF) to generate initial solutions. In order to deal with the uncertainty of the number of orders in the scheduling path and the various constraints during the optimization process, we adopt the simulated annealing (SA) algorithm to improve the initial solutions since this algorithm is less likely to be affected by the initial solution and is good at searching for the global optimal and strongly adaptable solutions. The SA algorithm is an iterative, adaptive, heuristic and probabilistic search algorithm for solving combinatorial optimization problems. It is based on the similarity between the physical annealing process in solids and the general combinatorial optimization problems. In the Metropolis sampling process, this algorithm follows the Metropolis rule to accept deterioration solutions at a certain probability, hence, it can jump off a local optimum traps and eventually get the global optimal solution with a great probability.

4.1. Preconditioning operation. Because reducing the search space can improve the efficiency of the decision making, we first perform the preconditioning operation to remove some infeasible orders before execution of the algorithm. Firstly, remove the infeasible order \( i \) from the order set when inequality \( d_{li} - d_{ei} < t_{si} \) or \( d_{li} < t_{pi} + t_{si} \) is satisfied. Secondly, perform selecting and scheduling operations on the remaining feasible orders.

4.2. Algorithm based on the SOF for generating an initial solution. In this section we describe the basic idea and the details of the SOF algorithm. Firstly, the algorithm starts the scheduling path with the dummy order 0. Secondly, it finds
out the earliest servable PSS order in the remaining orders when the current order is completed and inserts it into the current scheduling path. Finally, it repeats the insertion process until no order can be found. The details of the SOF algorithm can be described as follows:

**Step 1.** Define the set of all the feasible orders including the imaginary order $N = \{0, 1, ..., n + 1\}$, the set of all accepted and scheduled orders $A = \phi$, the set of orders to be inserted into the scheduling path $B = \phi$, and the set of orders to be accepted and scheduled $A = N \setminus (A \cup B)$; compute the service provision start time window $[d_{ei}, d_{li} - t_{si}]$ and the width of the service provision start time window $l_i = d_{li} - t_{si} - d_{ei}$.

**Step 2.** Construct the initial scheduling path, and let $\sigma_{now} = 0 \rightarrow n + 1$ be the current scheduling path and order 0 be the current order. Set $A = \{0, n + 1\}$.

**Step 3.** Give priority to the earliest servable order $k$ which satisfies the equality $d_{ek} = \min_{i \in A} \{d_{ei}\}$, and let $k \in B$:  

3.1. If $|B| = 1$, insert order $k$ right behind the current order, then update order $k$ as the current order, and update the current scheduling path as $\sigma_{now} = 0 \rightarrow ... \rightarrow k \rightarrow n + 1$. Remove order $k$ from $B$ and reset $B = \phi$, then go to Step 3.

3.2. If $|B| = 1$, select order $k$ according to $l_i = \min_{i \in B} \{l_i\}$, and insert order $k$ right behind the current order, then update order $k$ as the current order, and update the current scheduling path as $\sigma_{now} = 0 \rightarrow ... \rightarrow k \rightarrow n + 1$. Remove order $k$ from $B$. Repeat this step until $B = \phi$.

**Step 4.** Repeat Step 3 until $A = \phi$, then the current scheduling path $\sigma_{now} = 0 \rightarrow ... \rightarrow k \rightarrow n + 1$ obtained is the initial scheduling path needed.

4.3. **Construction of the cost function.** We add penalty to the objective function to deal with the time window constraints of service offering. The corresponding principle is that the orders in the scheduling path which can be completed within their time windows will not cause penalty increase, and the other orders in the scheduling path will cause penalty increase. We add a penalty value to the current scheduling path, so that it has a smaller adaptability. Based on this, we formulate the cost function of the scheduling path as

$$
    f(\sigma) = \sum_{i \in \sigma} y_i(r_i - a \max(d_{ei} + t_{si} - C_i, 0) - b \max(C_i - d_{li}, 0)),
$$

where a and b are constant large enough.

4.4. **Simulated annealing algorithm based on SOF (SOF-SA).**

**Step 1.** Load the data and perform the preconditioning process.

**Step 2.** Initialize control parameters: set the start temperature $T_{max}$, the cooling rate $q$, the end temperature $T_{end}$ and the iteration number $L$ for each $T$; set the current temperature $T_{now} = T_{max}$.

**Step 3.** Set parameters $f_{max} = 0$, $f_{best} = 0$.

**Step 4.** Generate initial solutions.

4.1. Use the SOF algorithm to generate initial scheduling path $\sigma_{now}$, and let the initial scheduling path be the current best solution path, namely, let $\sigma_{best} = \sigma_{now}$;

4.2. Compute the completion time $C_j$ of each order in $\sigma_{best}$;

4.3. Compute the number $m$ of orders which are beyond their time windows in $\sigma_{best}$;
4.4. Compute \( f_{\text{best}} = f(\sigma_{\text{best}}) = \sum_{i \in \sigma_{\text{best}}} y_i(r_i - a \max(d_{ei} + t_{si} - C_i, 0) - b \max(C_i - d_{li}, 0)) \);

**Step 5.** When \( m > 0 \):
5.1. Perform the following operations based on SA cyclically on the current scheduling path for \( k = 1, \ldots, L \):
5.1.1. Randomly generate two different integers \( a_1 \) and \( a_2 \) in \([1, n]\);
5.1.2. Exchange the positions of orders \( a_1 \) and \( a_2 \) in \( \sigma_{\text{best}} \), then place the sequence obtained from the exchange as the new scheduling path \( \sigma_1 \);
5.1.3. Compute the completion time \( C_i' \) of each order in \( \sigma_1 \) and the number \( m_1 \) of orders that are not within the time window in \( \sigma_1 \), and set \( m = m_1 \);
5.1.4. Compute \( f_{\text{max}} = \sum_{i \in \sigma_1} y_i(r_i - a \max(d_{ei} + t_{si} - C_i, 0) - b \max(C_i - d_{li}, 0)) \), \( \Delta f = f_{\text{max}} - f_{\text{best}} \), determine whether to accept a new solution according to the Metropolis rule. If \( \Delta f > 0 \), accept \( \sigma_1 \) as the new current solution path and place it as the initial point of the next SA operation, namely, set \( \sigma_{\text{best}} = \sigma_1 \), \( f_{\text{best}} = f_{\text{max}} \). If \( \Delta f \leq 0 \), generate a stochastic number \( a_3 \) from \([0, 1]\), and accept \( \sigma_1 \) as the new current solution path when \( P(\Delta f) > a_3 \), then set \( R_{\text{best}} = R_{\text{max}} \); reject \( \sigma_1 \) and continue to adopt the original best solution path as the initial point of the next SA operation when \( P(\Delta f) \leq a_3 \);
5.1.5. If \( k < L \), set \( k = k + 1 \), and go to Step 5.1.1, otherwise, go to the next step;
5.1.6. If \( T_{\text{now}} > T_{\text{end}} \), update the current temperature as the attenuation function \( T_{\text{now}} = g T_{\text{now}} \), and go to Step 5.1.1, otherwise, continue to the next step;
5.2. Calculate the earnings loading rate \( \omega_i = r_i/(C_i - C_{\text{pre}}) \) for the order \( i \) in \( \sigma_{\text{best}} \) which is not within the time window, where order \( \text{pre} \) is the order before order \( i \);
5.3. Exclude order \( h \) from the current scheduling path \( \sigma_{\text{best}} \) according to \( \omega_h = \min(\omega_i) \), namely, order \( h \) is rejected, then let the new scheduling path obtained be the current best scheduling path \( \sigma_{\text{best}} \);
5.4. Set \( T_{\text{now}} = T_{\text{max}} \) and let \( \sigma_{\text{best}} \) be the initial point of SA, then, return to Step 4.2;

**Step 6.** When \( m = 0 \), \( \sigma_{\text{best}} \) is a feasible solution path, and then perform the following operations for \( k = 1, \ldots, n_1 \):
6.1. Randomly generate two different integers \( a_1 \) and \( a_2 \) from \([1, n]\);
6.2. Exchange the positions of orders \( a_1 \) and \( a_2 \) in \( \sigma_{\text{best}} \), and set the sequence obtained from the exchange as the new scheduling path \( \sigma_1 \);
6.3. Compute the completion time \( C_i' \) of each order in \( \sigma_1 \) and the new \( f_{\text{max}} \);
6.4. If \( \sigma_1 \) is a feasible solution path and \( f_{\text{max}} > f_{\text{best}} \), then set \( \sigma_{\text{best}} = \sigma_1 \), \( f_{\text{best}} = f_{\text{max}} \); Otherwise, keep the original best solution path, then set \( k = k + 1 \), and return to Step 6.1.

**Step 7.** Output the current best solution path and compute the total profit.

5. **The dynamic acceptance and scheduling heuristic (DASH) algorithm.** Since the SA algorithm is inefficient and can only obtain near-optimal solutions for large-scale problems, we develop a heuristic algorithm named DASH based on dynamic acceptance and scheduling rule. The main idea of DASH is that: firstly, verify the start time of product production and that of the service provision’s time window of a PSS order respectively; secondly, accept and schedule order \( j \) in feasible orders which has the maximum revenue-load ratio defined as \( r_j/(C_{2j} - C_{2j, \text{pre}}) \), and place order \( j \) as the following order of the order \( \text{pre} \) in the existing scheduling path. Here, \( C_{1j} \) and \( C_{2j} \) denote the completion time of order \( j \)’s product production and
that of order j’s service offering respectively. The selection and acceptance decision in DASH algorithm is a dynamic process. During the execution of this algorithm, time goes on until at least one feasible order is found. When a feasible order is accepted and scheduled, its completion time at the production stage and service stage can be calculated respectively. Set the completion time of the service, namely the completion time of the order delivery, as the current time point. Any order whose completion time exceeds the time point of the time window is automatically rejected. The specific implementation steps of the DASH algorithm are described as follows:

**Step 1.** Define the set of all orders including the dummy order as \( N = \{0, 1, ..., n + 1\} \), the set of the accepted and scheduled orders as \( A = \phi \), the set of orders to be inserted into the scheduling path as \( B = \phi \), the set of rejected orders as \( B_R = \phi \), and the set of orders to be accepted and scheduled as \( \overline{A} = N \setminus (A \cup B_R) \). Compute the time window \([d_{ei}, d_{li} - t_{si}]\) of the service provision and the length of the time window \( l_i = d_{li} - t_{si} - d_{ei} \) for order \( i \).

**Step 2.** Set \( A = \{0, n + 1\} \), then construct the initial scheduling path \( \sigma_{\text{now}} = 0 \rightarrow n + 1 \) and place it as the current scheduling path. Let order 0 be the current order and its completion time be the current time point. Set \( t_{\text{now}} = C_{20} = 0, d_{\text{max}} = 1, \text{pre} = 0 \).

**Step 3.** If \( t_{\text{now}} \geq d_{\text{max}} \), then the obtained current scheduling path \( \sigma_{\text{now}} \) is the optimal solution. Go to Step 5.

**Step 4.** If \( t_{\text{now}} < d_{\text{max}} \), then perform the following actions.

4.1. If \(|\overline{A}| > 0\), find the largest value of the upper limits of the service provision time window of PSS orders in set \( \overline{A} \), and let it be the new delivery due date, namely set \( d_{\text{max}} = \max_{i \in \overline{A}} \{d_{li}\} \), then perform the following actions:

4.1.1. For each order \( j \) in \( \overline{A} \) which satisfies the inequality \( d_{lj} - t_{sj} \geq t_{\text{now}} \), compute the revenue load factor \( e_j = r_j / (C_{2j} - C_{2, \text{pre}}) \):

4.1.2. Find order \( k \) with the largest value of the revenue load factor from \( \overline{A} \) according to \( e_k = \max_{j \in N} \{r_j / (C_{2j} - C_{2, \text{pre}})\} \) and treat it as the order to be inserted. Let \( k \in B \);

4.1.3. When \(|B| > 1\), select order \( k \) according to \( l_k = \min_{i \in B} \{l_i\} \) and perform the following actions;

4.1.3.1. If \( \max(t_{\text{now}}, C_{1, \text{now}} + t_{pk}, d_{ek}) + t_{sk} \leq d_{lk} \), let order \( k \) be the current order in \( \sigma_{\text{now}} \), and update the original order as order \( \text{pre} \) and order \( k \) as the current order so as to obtain the updated current scheduling path \( \sigma_{\text{now}} = 0 \rightarrow ... \rightarrow \text{pre} \rightarrow k \rightarrow n + 1 \), set \( t_{\text{now}} = C_{2k} \), and remove order \( k \) from \( B \), then go to Step 4.1.3;

4.1.3.2. If \( \max(t_{\text{now}}, C_{1, \text{now}} + t_{pk}, d_{ek}) + t_{sk} > d_{lk} \), remove order \( k \) from \( B \), and set \( k \in B_R \), then go to Step 4.1.3;

4.1.4. When \(|B| = 1\): if \( \max(t_{\text{now}}, C_{1, \text{now}} + t_{pk}, d_{ek}) + t_{sk} \leq d_{lk} \), let order \( k \) be the current order in \( \sigma_{\text{now}} \), and update the original order as order \( \text{pre} \) and order \( k \) as the current order so as to obtain the updated current scheduling path \( \sigma_{\text{now}} = 0 \rightarrow ... \rightarrow \text{pre} \rightarrow k \rightarrow n + 1 \), then set \( t_{\text{now}} = C_{2k} \), and reset \( B = \phi \), and go to Step 3; If \( \max(t_{\text{now}}, C_{1, \text{now}} + t_{pk}, d_{ek}) + t_{sk} > d_{lk} \), then set \( k \in B_R \) and go to Step 3;

4.2. If \(|\overline{A}| = 0\), all of the orders have been accepted and scheduled, then go to Step 5.
Step 5. Output the current scheduling path \( \sigma_{\text{now}} = 0 \to \ldots \to k \to n+1 \), which is the optimal solution of the problem.

6. Computational studies. In order to validate the performance and efficiency of the model MA and the algorithms developed, we use lingo 15.0 (Extended Version) to solve the MA formulation for problems with small size, and program the SOF-SA algorithm and DASH algorithm relatively on Matlab2010 for problems with more than 10 PSS orders. Comparison and analysis of the simulation results are carried out. The hardware environment of the numerical simulation is Core i5/2.6 GHz/4.00G RAM. And the software environment is Windows 8 operating system.

6.1. Data generation. There has not been any standard experimental data which can be used to research the PSS OAS problem with time window constraints yet. In practice, decision makers can obtain relevant data according to the characteristics and requirements of customers. The data generation method in our work is referring to Cesaret et al. [3]. The integer parameters of orders in this paper are generated randomly from uniform distributions with different intervals. For each PSS order \( i \), the revenues \( r_i \) is an integer number generated randomly from the distribution \( U[1,20] \). The product processing time \( t_{pi} \) is an integer number generated randomly from the distribution \( U[1,10] \), so does the service offering time \( t_{si} \). The time windows of order delivery and service offering are presented as: \( d_{ei} \in U[0,\tau T] \), \( d_{li} \in U[T(1-\tau - R/2), T(1-\tau + R/2)] \), where \( \tau \) is the relative range factor of \( d_{ei} \), \( R \) is the relative range factor of \( d_{li} \) and \( T = \sum_{i=1}^{n} t_{pi} + \min_{i=1,\ldots,n} \{ t_{si} \} \).

The initial temperature is an important parameter that has great influence on the effect and efficiency of SOF-SA algorithm. We validated the performance of five different initial temperatures: 600, 800, 1000, 1200 and 1600. It is observed that the algorithm performs best when \( T_{\text{max}} = 800 \). Hence, the initial temperature in SOF-SA algorithm is set to 800. The other parameters are set as \( T_{\text{end}} = 0.001 \), \( L = 400 \) and \( q = 0.7 \).

6.2. Analysis of the computational results. \( \tau \) which is the relative range factor of \( d_{ei} \) takes three different values: 0.1, 0.2 and 0.3. \( R \) which is the relative range factor of \( d_{li} \) takes three different values: 0.2, 0.4 and 0.6. Thus, for each problem size, there are 9 combinations of \( \tau \) and \( R \). “CPU(s)” represents the corresponding CPU time of the performances, in terms of seconds.

6.2.1. The computational results of the MA formulation. Our computational results show that all of the randomly generated instances within 12 PSS orders could be solved to optimality in reasonable time with lingo 15.0. When the number of PSS orders increased to 13, only a very few numbers of instances could be solved. None of instances with more than 13 orders could be solved optimally. The performances of the MA formulation with different M-values respectively used for 10 PSS orders instances and 12 PSS orders instances are presented in table 1.

From table 1, we can observe that the optimal solution for small size OAS problem in PSSC can easily be obtained through solving the MA model. Both the two big-M values are effective. More than half of instances can be solved optimally in a shorter time when big-M takes the value \( \hat{M}_{ij} = d_{li} + t_{sj} \), \( i,j = 1,\ldots,n \), and \( i \neq j \). And the time consuming for solving the other instances with big-M taking the value \( \hat{M}_{ij} \) is not much longer than that for cases with big-M taking the value \( M = \sum_{i=1,\ldots,n} (t_{pi} + t_{si}) \). So we can say that the value \( \hat{M}_{ij} \) is more effective than
Table 1. The performance of MA

| $\tau$ | $R$ | $M = \sum_{i=1}^{n} (t_{pi} + t_{si})$ | $M_{ij} = d_{ij} + t_{sj}$ | CPU(s) | CPU(s) |
|--------|-----|-----------------------------------|----------------|----------|---------|
| 0.1    | 0.2 | 19.45                             | 684.96         | 19.55    | 675.23  |
| 0.1    | 0.4 | 1.28                              | 104.66         | 1.19     | 103.99  |
| 0.1    | 0.6 | 0.36                              | 24.52          | 0.34     | 24.75   |
| 0.2    | 0.2 | 2.45                              | 149.15         | 2.30     | 126.21  |
| 0.2    | 0.4 | 0.41                              | 11.61          | 0.39     | 11.5    |
| 0.2    | 0.6 | 0.13                              | 0.07           | 0.28     | 0.07    |
| 0.3    | 0.2 | 1.20                              | 4.57           | 1.19     | 4.08    |
| 0.3    | 0.4 | 1.97                              | 0.16           | 1.91     | 0.23    |
| 0.3    | 0.6 | 0.06                              | 0.54           | 0.06     | 0.54    |

The time spending on solving the MA model increases rapidly as the order number increased from 10 to 12 for $\tau$ taking the value no bigger than 0.3.

6.2.2. The computational results of the SOF-SA algorithm and DASH algorithm. In order to display the performances of the two algorithms to handle large instances of the OAS problem in PSSC, the problem size $n$ takes four different values: 10, 20, 50 and 100. 36 problem instances are generated. For each problem instance, the SOF-SA algorithm and DASH algorithm are performed 20 times respectively. $\pi_{opt}^H$ represents the average objective value obtained from the performing each heuristic algorithm 20 times. Let $\pi_{opt}^*$ represent the optimal revenue obtained from solving MA model. “GAP1(%)” represents the deviation percentage of the objective value for the case with 10 PSS orders, which is defined as $(\pi_{opt}^* - \pi_{opt}^H)/\pi_{opt}^* \times 100\%$. Since we can not obtain the optimal solutions for the cases with 20, 50, 100 PSS orders within reasonable time, the relevant upper bound of the objective value is defined as $\pi_{UB}^* = (\max \pi_{H1} + \max \pi_{H2})/2$, where $\max \pi_{H1}$ is the maximize objective value obtained from performing the SOF-SA algorithm 20 times and $\max \pi_{H2}$ is the maximize objective value obtained from performing the DASH algorithm 20 times. “GAP2(%)” represents the deviation percentage of the objective value for the cases with 20, 50, 100 PSS orders respectively, which is defined as $(\pi_{UB}^* - \pi_{opt}^H)/\pi_{UB}^* \times 100\%$.

“CPU(s)” is the average CPU time of each algorithm, in terms of seconds.

The comparisons between the simulation solutions obtained by the SOF-SA algorithm and DASH algorithm are presented from table 2 to table 5. The comparisons for $n = 10$ are presented in table 2. The comparisons for $n = 20$ are presented in table 3. The comparisons for $n = 50$ are presented in table 4. The comparisons for $n = 100$ are presented in table 5.

Comparing and analyzing the simulation results in table 2, table 3 and table 4, we can find out that:

Firstly, both the SOF-SA algorithm and DASH algorithm can solve instances with no more than 50 PSS orders in a few seconds. Although the SOF-SA algorithm is longer time-consuming compared with the DASH algorithm, the revenue deviation of the optimal OAS decision obtained from the former algorithm is lower, and its running time is within an acceptable range. It is mainly because the SA algorithm can get the optimal solution of the problem with a high probability, and the DASH algorithm is a heuristic algorithm which can usually obtain the approximate optimal
solution fast. As the problem size increases, the advantage in revenue deviation of the SOF-SA algorithm is weakening gradually, and the growth rate of its running time is significantly large.

Secondly, in the same problem size, when the value of \( R \) is fixed, the running time of SOF-SA algorithm increases significantly as \( \tau \) increases, but the running time of DASH algorithm decreases gradually. It is mainly because the relative range of
the PSS order’s time window becomes smaller as $\tau$ increases, and fewer orders are expected to be accepted. So it requires longer time for the SOF-SA algorithm to reject more orders. On the contrary, the DASH algorithm requires shorter time for fewer order selection and insertion operations to find the best solution algorithm.

Thirdly, in the same problem size, the running time of the SOF-SA algorithm becomes shorter as the relative range of $d_{li}$ increases, so does the running time of the DASH algorithm. Comparisons based on table 2, table 3 and table 4 for the fixed value of $\tau$ show that: as $R$ increases, although the narrower time window of each PSS order leads to more rejected orders for the SOF-SA algorithm, the gaps between $d_{li}$ of different orders become larger, which makes refusing an order more easily; as $R$ increases, the narrower time window of each PSS order leads to fewer accepted and inserted orders for the DASH algorithm, and the larger gaps between $d_{li}$ of different orders make accepting an order more easily. Consequently, the running time of the two algorithms becomes shorter.

| $n$ | $\tau$ | $R$ | GAP2(%) | CPU(s) |
|-----|--------|-----|---------|--------|
| 100 | 0.1    | 0.2 | 25.8    | 1365.49 |
| 100 | 0.1    | 0.4 | 30.3    | 954.21  |
| 100 | 0.1    | 0.6 | 19.7    | 996.73  |
| 100 | 0.2    | 0.2 | 25.1    | 1681.32 |
| 100 | 0.2    | 0.4 | 27.9    | 1307.76 |
| 100 | 0.2    | 0.6 | 29.4    | 1178.54 |
| 100 | 0.3    | 0.2 | 32.0    | 1873.55 |
| 100 | 0.3    | 0.4 | 26.5    | 1054.27 |
| 100 | 0.3    | 0.6 | 35.2    | 1175.92 |

Throughout table 5, we can observe that the SOF-SA algorithm spends much longer time obtaining the approximate optimal solution for instances with 100 orders, but the DASH algorithm can obtain a better approximate optimal solution in a very short time when $n = 100$. So the DASH algorithm is more suitable for solving large-scale problems. In addition, the running time of the DASH algorithm becomes shorter as the time window gets wider when the DASH algorithm is used to solve large-scale problems, while the running time of the SOF-SA algorithm does not change regularly.

From the above comparisons and analysis we can observe that:

1. The SOF-SA algorithm can get better solutions in a reasonable time for PSS OAS problems with small size. And it performs poorly when being used for PSS OAS problems of large scale. So it is suitable for solving small size instances.

2. The DASH algorithm can get effective solutions in a short time for both PSS OAS instances in small size and those of large scale. It performs obviously better than the SOF-SA algorithm when being used for solving PSS OAS problem with large size. But it performs poorly for solving PSS OAS problem with small size. So it is suitable for solving large size instances.

Note that we don’t test the formulation and the algorithms for instances in which the value of $\tau$ is larger than 0.3. Because the time windows of the PSS orders will
be too tight so that few orders can be accepted when the value of $\tau$ is greater than 0.3.

7. **Conclusions.** In this paper, we focus on the integrated order acceptance and scheduling decision of the PSSC which supplies customers with various PSS to satisfy their personalized integrated “product + service” demands. The service providing and delivery in the integrated system are carried out synchronously and involve customer participation. So the PSS order delivery is usually restricted by hard time windows determined by customer requirements. And the OAS decision making should take the coordination of the product production executed by the product supplier and the service offered by the service supplier into consideration. Based on this background, we formulate the PSS OAS problem as a MILP model to maximize the total revenue of the PSSC, develop two effective values for the big-M coefficients. Since the MILP model MA can only be used to solve the OAS problem optimally with small number of PSS orders, the SOF-SA algorithm and the DASH algorithm are proposed to solve problem instances of different larger sizes. Numerical simulation is used to verify the effectiveness of the MILP model formulated and the two algorithms developed. The comparison and analysis of the two algorithms’ performance show that: the MILP model and the two value of big-M are effective; the SOF-SA algorithm performs better for solving small size instances with no more than 50 PSS orders while the DASH algorithm performs better for solving large size instances with 50 to 100 PSS orders; the width of time windows and the skip distance of service offering’s upper/lower bound both have obvious influence on the decision efficiency of the PSSC.

**Acknowledgments.** We would like to thank the editors and referees for their valuable comments and suggestions.

**REFERENCES**

[1] T. S. Baines, H. W. Lightfoot, S. Evans, . . . and J. R. Alcock, State-of-the-art in product-service systems, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 221 (2007), 1543–1552.

[2] F. H. Beuren, M. G. G. Ferreira and P. A. C. Miguel, Product-service systems: a literature review on integrated products and services, *Journal of Cleaner Production*, 47 (2013), 222–231.

[3] B. Cesaret, C. Oğuz and F. S. Salman, A tabu search algorithm for order acceptance and scheduling, *Computers & Operations Research*, 39 (2012), 1197–1205.

[4] H. K. Chen, C. F. Hsueh and M. S. Chang, Production scheduling and vehicle routing with time windows for perishable food products, *Computers & Operations Research*, 36 (2009), 2311–2319.

[5] R. Esmaeilbeigi, P. Charkhgard and H. Charkhgard, Order acceptance and scheduling problems in two-machine flow shops: New mixed integer programming formulations, *European Journal of Operational Research*, 251 (2016), 419–431.

[6] M. Garetti, P. Rosa and S. Terzi, Life Cycle Simulation for the design of Product–Service Systems, *Computers in Industry*, 63 (2012), 361–369.

[7] T. C. Kuo, Simulation of purchase or rental decision-making based on product service system, *The International Journal of Advanced Manufacturing Technology*, 52 (2011), 1239–1249.

[8] T. C. Kuo and W. M. Ling, The optimisation of maintenance service levels to support the product service system, *International Journal of Production Research*, 50 (2012), 6691–6708.

[9] I. S. Lee, Minimizing total tardiness for the order scheduling problem, *International Journal of Production Economics*, 144 (2013), 128–134.
[10] S. Lee, Y. Geum, H. Lee and Y. Park, Dynamic and multidimensional measurement of product-service system (PSS) sustainability: a triple bottom line (TBL)-based system dynamics approach, *Journal of Cleaner Production*, 32 (2012), 173–182.

[11] N. Li and Z. Jiang, Modeling and optimization of a product-service system with additional service capacity and impatient customers, *Computers & Operations Research*, 40 (2013), 1923–1937.

[12] Y. K. Lin and C. S. Chong, A tabu search algorithm to minimize total weighted tardiness for the job shop scheduling problem, *Journal of Industrial and Management Optimization*, 12 (2016), 703–717.

[13] H. Lockett, M. Johnson, S. Evans and M. Bastl, Product Service Systems and supply network relationships: An exploratory case study, *Journal of Manufacturing Technology Management*, 22 (2011), 293–313.

[14] C. Low, R. Li and C. Chang, Integrated scheduling of production and delivery with time windows, *International Journal of Production Research*, 51 (2013), 897–909.

[15] E. Manzini and C. Vezzoli, A strategic design approach to develop sustainable product service systems: Examples taken from the ‘environmentally friendly innovation’ Italian prize, *Journal of Cleaner Production*, 11 (2003), 851–857.

[16] R. Maull, A. Smart and L. Liang, A process model of product service supply chains, *Production Planning & Control*, 25 (2014), 1091–1106.

[17] O. K. Mont, Clarifying the concept of product-service system, *Journal of Cleaner Production*, 10 (2002), 237–245.

[18] F. T. Nobibon and R. Leus, Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment, *Computers & Operations Research*, 38 (2011), 367–378.

[19] C. Oguz, F. S. Salman and Z. B. Yalın, Order acceptance and scheduling decisions in make-to-order systems, *International Journal of Production Economics*, 125 (2010), 200–211.

[20] T. Sakao, A. Ö. Rönnbäck and G. Ö. Sandström, Uncovering benefits and risks of integrated product service offerings—Using a case of technology encapsulation, *Journal of Systems Science and Systems Engineering*, 22 (2013), 421–439.

[21] L. H. Su, P. S. Chen and S. Y. Chen, Scheduling on parallel machines to minimise maximum lateness for the customer order problem, *International Journal of Systems Science*, 44 (2013), 926–936.

[22] C. A. Ullrich, Integrated machine scheduling and vehicle routing with time windows, *European Journal of Operational Research*, 227 (2013), 152–165.

[23] UNEP, The role of product service systems in a sustainable society, In. http://www.unep.fr/scp/design/pdf/pss-brochure-final.pdf, (2001).

[24] X. Wang, X. Xie and T. C. E. Cheng, Order acceptance and scheduling in a two-machine flowshop, *International Journal of Production Economics*, 141 (2013), 366–376.

[25] X. Wang, X. Xie and T. C. E. Cheng, A modified artificial bee conoly algorithm for order acceptance in two-machine flowshop, *International Journal of Production Economics*, 141 (2013), 14–23.

[26] Y. Xiao, R. Zhang, Q. Zhao and I. Kaku, Permutation flow shop scheduling with order acceptance and weighted tardiness, *Applied Mathematics and Computation*, 218 (2012), 7911–7926.

[27] W. Xie, Z. Jiang, Y. Zhao and X. Shao, Contract design for cooperative product service system with information asymmetry, *International Journal of Production Research*, 52 (2014), 1658–1680.

[28] Z. Xu, X. Ming, W. Song, M. Li, L. He and X. Li, Towards a new framework: Understanding and managing the supply chain for product-service systems, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 0954405414521189, (2014).

[29] L. Zhang, L. Lu and J. Yuan, Single machine scheduling with release dates and rejection, *European Journal of Operational Research*, 198 (2009), 975–978.
[30] C. Zhao, Y. Yin and T. C. E. Cheng and C. C. Wu, Single-machine scheduling and due date assignment with rejection and position-dependent processing times, *Journal of Industrial and Management Optimization*, **10** (2014), 691–700.

[31] X. Zhong, J. Ou and G. Wang, Order acceptance and scheduling with machine availability constraints, *European Journal of Operational Research*, **232** (2014), 435–441.

Received January 2015; 1st revision August 2016; final revision December 2016.

E-mail address: danbin@cqu.edu.cn
E-mail address: gaohuali1982@163.com
E-mail address: neidanaruto28163.com
E-mail address: 756885541@qq.com
E-mail address: 1042950641@qq.com