Scheduling Model Design of Logistics Service Supply Chain

Yaoxuan Li 517201910672 lyxtracy1@sjtu.edu.cn
Tianyao Shi 517021910623 sthowling@sjtu.edu.cn
Kuangjin Wu 517021910732 kuangjin@sjtu.edu.cn

Department of Computer Science,
Shanghai Jiao Tong University, Shanghai, China

Abstract. The implementation of the logistics service supply chain requires good time scheduling. In this paper we mainly analyze whether to consider the relationship between time windows of supplier operation and customer requirement will cause different results. We use discrete values to represent the time range provided by the supplier, find feasible solution and use local search to find a better solution for multi-objective programming. When looking for a better solution, we derive a single objective as the Euclidean distance from the present point to the ideal point and minimize it, which make all of the three objectives have good approximate feasible solutions.
Besides, we also analyze the effectiveness and complexity of the algorithm, and use the results obtained by the algorithm to analyze the data and derive the performance of the algorithm. Our algorithm always gives good time scheduling at most of time, so that the three objectives required by the project are as good as possible. In general, our model and algorithm have handled these problems very well and have reached satisfactory conclusions.
Keywords: Time scheduling, Logistics service supply chain, Multi-objective programming, Local search.

1 Introduction

1.1 Problem Background

With the accelerated pace of life, people's demand for logistics has gradually increased, and the logistics supply chain has emerged in this context. In the logistics service supply chain, main participants include customers, functional logistics service providers and logistics service integrators. As we all know, completion time is an important index of logistics service performance and then the time scheduling problem becomes one of the key problems in logistics service scheduling. As far as is known, many uncertain factors such as weather, traffic conditions and vehicle breakdown will have a certain impact on the logistics supply chain. Therefore, we need to build models and analyze these factors through algorithms to enable the logistics service supply chain to provide more effective services.

1.2 Restatement of the Problem

We need to solve two main problems:

1 **Scheduling Model Design without the Relationship between Time Windows of Supplier Operation and Customer Requirement**
In this problem, there is a two-echelon logistics service supply chain composed of one logistics service integrator and several functional logistics service providers. A logistics service integrator can receive multiple orders at the same time, and each order consists of many service processes. And each logistics service order consists of multiple service processes, which could be divided into two types, the customization service process and the mass service process. We need to analyse the orders and generate a scheduling scheme so that we can minimize the difference between the total expected service time and customer requirement time, while minimizing the total order operation cost of the logistics service supply chain and maximizing the satisfaction of customers.

2 **Scheduling Model Design with the Relationship between Time Windows of Supplier Operation and Customer Requirement**
In this problem, we need to consider the relationship between time windows of supplier operation and customer requirement. Based on this condition, we modify the scheduling model and recalculate the new solution.

1.3 Our work

(1) At the beginning we assume that the distribution given by the FLSP(Functional Logistics Service Supplier) is continuous, but the problem is too complicated to solve, so we try to simplify the problem and set the given range to discrete. We design a multi-objective programming to solve this problem. The execution time distribution of each process is discrete. We look for a suitable CODP(Customer Order Decoupling Point), traverse each process, find feasible solutions and continuously optimize our feasible solutions during the traversal process until we find a satisfactory solution.

(2) After finding a CODP, we do a local search for the processes in all orders. In this part we design a good algorithm to solve the problem.

(3) When considering the relationship between time windows of supplier operation and customer requirement, we have added new constraints to the planning of the original algorithm. At this time, the difference between the expected service time and the expected operation time will affect the extra operation cost, which is also the impact of considering the new relationship.

(4) After designing the model, we implement it with the corresponding algorithm. At the same time, we analyze the time complexity and efficiency of the algorithm and compare it with the results of the heuristic algorithm we find to see if the result of the algorithm is practical.

2 Notations

Here we list the symbols and their meanings the are used frequently in our model. Other symbols that are used only once will be described later.

symbol	description
LLSC	Logistics Service Supply Chain
LSI	Logistics Service Integrator
FLSP	Functional Logistics Service Supplier
CODP	Customer Order Decoupling Point
k	Value of CODP: before which process all are mass service processes, and from which process on all are customization service processes
N	Size of the set of customer orders being scheduled by LSI
N_j	Number of processes in the jth order
c_i	Normal cost per unit time of the ith mass service process
c_{ij}	Normal cost per unit time of the ith customization service process in the jth order
U_i	Completion time of ith mass service process as a Uniform Distribution $U_i \sim [a_i, b_i]$
U_{ij}	Completion time of ith customization service process in the jth order as a Uniform Distribution $U_i \sim [a_{ij}, b_{ij}]$
t_i	Actual completion time of ith mass service process, $f(t_i) = \frac{1}{b_i - a_i}$
t_{ij}	Actual completion time of ith customization service process in the jth order, $f(t_{ij}) = \frac{1}{b_{ij} - a_{ij}}$
T_i	Expectation of t_i, $\mathbb{E}(t_i) = T_i = \frac{a_i + b_i}{2}$
T_{ij}	Expectation of t_{ij}, $\mathbb{E}(t_{ij}) = T_{ij} = \frac{a_{ij} + b_{ij}}{2}$
T_j^r	Required completion time of the jth customer order
c_{j1}	Possible ahead of schedule of the required time T_j^r of the jth customer order
c_{j2}	Possible postpone of the required time T_j^r of the jth customer order
T_{ij}^e	Expected completion time of ith process in jth order as a decision variable used by LSI
T_i^e	Expected completion time of ith process if process i is selected as a mass service process
L_{i+1}^p	Upper limit of the time postponed in the ith service process
L_{i+1}^a	Upper limit of the time head of schedule in the ith service process
S_i	Satisfaction of the ith mass service process
S_{ij}	Satisfaction of the ith customization service process in jth order
s_i	Capability of generating satisfaction of the ith mass service process
s_{ij}	Capability of generating satisfaction of the ith customization service process in jth order
P_{ij}	Penalty cost per time unit paid by FLSP to LSI for the difference between t_{ij} and T_{ij}^e
E_{ij}	Extra cost per time unit paid by LSI to FLSP when $T_{ij}^e \notin [a_{ij}, b_{ij}]$
X_1	Total cost of LSSC as one optimizing objective to be minimized
X_2	Time difference between the total expected service time and customer requirement time as one optimizing objective to be minimized
X_3	Total satisfaction as one optimizing objective to be maximized
w_i	Weight of objective i in the multi-objective programming
X	Single objective converted from the multi-objective programming problem using Ideal Point Method

3 Scheduling Model Design without the Relationship between Time Windows of Supplier Operation and Customer Requirement

3.1 Assumptions and Definitions

In our model that does not consider the relationship between time window of supplier operation and customer requirements, we assume that

1. **The service orders consist of serveral service processes that happens strictly in predetermined time order.** That is, a service order O_j may consist of process $1, 2, \cdots, N_j$, where N_j is the total number of processes in O_j. Process i can start only after process $i-1$ is completely finished.

This is determined by the rigid requirement caused by upstream and downstream operations of LSSC (Logistics Service Supplier Chain).

To be specific, assume that the Cai Niao logistics service center in SJTU is to deliver a batch of goods that SJTUers ask it to, to places all over China. The first process is to collect all the goods and target locations from SJTUers. The second process is to pack them so that their destinations are clearly shown on the packet by attaching a bar code to every packet. Then it sends the batch of goods to a larger logistics service scheduling center, where they are decided to which collecting and distributing center on their way to destination they should first go. Before they reach the collecting and distributing center, they can do nothing but follow the scheduled route. And nor can it happen that they are sent to the scheduling center before they are packed.

By the way we strongly criticize the explanation that these processes can happen out of order in the wechat group, which is misleading and confusing since it is contraditory to the written text in the 6th and 7th data description of Section 2.1 in the Project Description file. Out-of-order arrangement is also not compatible with the real condition in logistics, as we just demonstrated. We take long time trying to explain the contradiction and understand the problem context under this kind of conflict. And this kind of conflict is not alone, see our 3rd and 4th assumption.

2. **The service processes in orders continue one after another without any stagnation.** In other words, process i starts immediately after process $i-1$ finishes. This is also realistic as there is no such stagnation in a real LSSC, everything keeps moving to avoid overstock in any tranporting node.
3. Service processes could be divided into two types, the customization service process and the mass service process. **Each service process can be selected to be a mass service process or a customization service process By LSI (Logistics Service Integrator)**, rather than letting the customers to explicitly state which processes need be massive and which processes need be customized.

   By making this assumption we further clarify the semantics of customization service process and mass service process. Though the word customization may imply that customers can customize their orders as they wish, in logistics context, the word actually refers to the order-specific processes. That is to say,*a customized process is a process that is not to be integrated with other processes to form a mass process* . To be more specific, consider the Cai Niao example where the goods are to be delivered all over China. Though they have different destinations, they share the first serveral processes as mass processes until they reach the scheduling center, after which they may go to different collecting and distributing centers, as a customized process.
4. The customers need not know the details of what processes their orders consist of. They only provide a destination, and the processes generate automatically to the problems's nature. **The only thing in their requirement is desired completion time of their orders.**

   This is also very right in real life: we do not ask a logistics service provider to use Tesla Model S to transport our goods when our goods go from Shanghai to Suzhou, then change to horse before they reach Nanjing. We only specify the destination, and if some order is too slow, we make a call to complain. For this reason we strongly disagree with the explanation in the wechat group that customers need to explicitly state which processes in their order is to be customized. In fact they do not have a clue what a process is.
5. **Each scheduling task performed by LSI aims at only one set of customer orders.** One set of customer orders is the basic unit in scheduling work, once the set has been decided, the new arriving orders cannot insert and must wait for the next set of order. We denote N the size of this set. This is also to make the problem we work on more realistic.
6. **There exists a CODP in each set of customer orders.** Though the scheduler can select any process in the set of orders to be mass service process or customization service process, it would be better to select a point *before which all processes are mass service processes, and from which on all processes are customization service processes.*

   This assumption also agrees with the nature of a logistics problem situation, like the Cai Niao problem we mentioned above. And it simplifies the problem greatly. Though "customized-massive-customized" would be more common in most logistics problems, we can view the first customized phase together with part of the massive phase to be the reverse of our "mass-customization" model here. Then it suffice to use the scheduling method we are to introduce later in this paper to solve both problem situations.

7. **The normal service cost per unit time in mass service is cheaper than that of customization service.** This is intuitive and common sense, but contradictory to the 2nd data description of Section 2.1 in the Project Description file. But if mass service has more expensive unit cost, it is meaningless to use it: the time scheduling capability is weaker compared to customized service, a change in the mass service requirement changes time schedule in all orders to the same extent, while scheduling a customization service process is more flexible. If by scheduling a mass service process LSI can satisfy all orders' time requirements, changing it to a customization service process LSI can also manage to satisfy all orders' time requirements, but not vise versa.

   To keep this opimization problem meaningful, we decide to violate the project descriptions and assume mass service processes to be cheaper in unit cost. And we synthesize the unit cost of mass service for this project using this principle.
8. **Each service process is handled by a single FLSP (Functional Logistics Service Supplier).** The LSI we are to play has strong integrating capability, such that it can always find more FLSPs in contract with it to meet the incremented service requirement derived from changing a mass service process into serveral customization service processes. That is to say, **there is not any capacity constraint in any service process**. In customized phase, serveral FLSPs respectively handles serveral orders, and their work is concurrent. This would be pretty resonable in real life except for special periods like Double Eleven.
9. Each process handled by an FLSP takes some time to complete, and the completion time is a time window. To be specific, **the completion time of process i is assumed to be a Uniform Distribution** $U_i \sim [a_i, b_i]$. Here we assume that selecting i as a mass or customization service process does not change the distribution, which is diffrent with in Section 4.1. For different FLSPs, their completion time is independent. We denote t_{ij} the actual completion time of process i in the jth order, where we have $\mathbb{E}(t_{ij}) = T_{ij} = \frac{a_i+b_i}{2}$ and $f(t_{ij}) = \frac{1}{b_i-a_i}$. If process i is selected as a mass service, we denote t_i the actual completion time it, $\mathbb{E}(t_i) = T_i = \frac{a_i+b_i}{2}$ and $f(t_i) = \frac{1}{b_i-a_i}$.

   This assumption is also very true for real life as many uncertain factors including weather, traffic condition and vehicle troubles, possibly lead to difference between actual service time and standard time spending on completing a certain service.
10. **The required completion time of the jth customer order is also an acceptable time window which is an interval** $[T_j^r - c_{j1}, T_j^r + c_{j2}]$**.** Here c_{j1}, c_{j2} are the possible ahead of schedule and postpone of the required time T_j. The absolute time is assumed to start at 0.
11. **The LSI does scheduling by determining the expected completion time T_{ij}^e for the ith process in jth order.** Here "expected" means LSI expects corresponding FLSP to complete service with such a time. There is nothing to do with the expectation of the completion time as a Uniform Distribution. If process i is selected as mass service process, then $\forall j$, $T_{ij}^e = T_i^e$.
12. **Expected completion time T_{ij}^e must fall in the time window of FLSP operation.** That is, $T_{ij}^e \in [a_i, b_i]$ $\forall i, j$. LSI cannot expect FLSPs to do something beyond their capabilities.
13. We define **the time postponed in the ith service process** in jth order to be the difference between the expected completion time T_{ij}^e and the expectation of completion time $T_{ij} = \frac{a_i+b_i}{2}$, when $T_{ij}^e > T_{ij}$. Literally, it evaluates the amount of time that LSI expects FLSP to postpone upon normal service time.

    We then define **the upper limit of the time postponed in the ith service process** that can be endured by $(i+1)$th process by L_{i+1}^p. **We assume that $T_{ij}^e - T_{ij} \leq L_{i+1}^p$ must be satisfied.**
14. We define **the time ahead of schedule in the ith service process** in jth order to be the difference between the expectation of completion time $T_{ij} = \frac{a_i+b_i}{2}$ and the expected completion time T_{ij}^e, when $T_{ij}^e < T_{ij}$. Literally, it evaluates the amount of time that LSI expects FLSP to be ahead of normal service time.

    We then define **the upper limit of the time ahead of schedule in the ith service process** that can be endured by $(i+1)$th process by L_{i+1}^a. **We assume that $T_{ij}^e - T_{ij} \geq L_{i+1}^a$ must be satisfied.**
15. Once a service process i in jth order is finished, it generates a satisfaction S_{ij} or S_i if it is a mass service process. **Satisfaction could be the highest if the work is arranged in normal service time ($T_{ij}^e = T_{ij}$ or $T_i^e = T_i$) and could be impaired if the work is assigned with a difference from the normal service time.** And for different processes, they possess different capabilities of generating satisfaction. We denote the capabilities by s_{ij} and s_i.

In particular, the satisfactions are gernerated according to the following formula:

$$S_i = \begin{cases} \frac{T_i^e}{T_i} \cdot s_i, & T_i^e < T_i, \\ \frac{T_i}{T_i^e} \cdot s_i, & T_i^e > T_i, \\ s_i, & T_i^e = T_i. \end{cases}$$

$$S_{ij} = \begin{cases} \frac{T_{ij}^e}{T_{ij}} \cdot s_{ij}, & T_{ij}^e < T_{ij}, \\ \frac{T_{ij}}{T_{ij}^e} \cdot s_{ij}, & T_{ij}^e > T_{ij}, \\ s_{ij}, & T_{ij}^e = T_{ij}. \end{cases}$$

16. If in practice the actual completion time of process i in the jth order, t_{ij} is different from the completion time T_{ij}^e expected by LSI, FLSP needs to pay LSI with penalty cost P_{ij} per time unit.
17. **The goal of this scheduling problem is to minimize the total cost, minimize the time difference between the total expected service time and customer requirement time, and maximize the total satisfaction.** Each of the goal has weight w_i correspondingly.

3.2 Problem formulation

From Assumption 15 we define the problem clearly as a multi-objective programming problem. Now we give mathemetical description of each goal and the constraints of this problem.

1. Minimizing the total cost X_1
   Here since we have constrained $T_{ij}^e \in [a_i, b_i]$ in Assumption 12, the extra cost is not generated. The total cost consists of two parts: the total normal service cost paid by LSI to FLSPs, and the total penalty cost paid by FLSPs to LSI, which is viewed as a negative value since concpetually they offset each other. Then we have:

$$\text{Min} \quad X_1 = \sum_{i=1}^{k-1} \int_{a_i}^{b_i} (c_i \cdot t_i - |t_i - T_i^e| \cdot P_i) f(t_i) dt_i + \sum_{j=1}^{N} \sum_{i=k}^{N_j} \int_{a_i}^{b_i} (c_{ij} \cdot t_{ij} - |t_{ij} - T_{ij}^e| \cdot P_{ij}) f(t_{ij}) dt_{ij} \tag{1}$$

   Here c_i refers to the normal cost per unit time of the ith mass service process, c_{ij} refers to the normal cost per unit time of the ith customization service process in the jth order, and k is the value of CODP.
2. Minimizing the total time difference X_2
   This is an easy one to derive with our sufficient assumptions and definitions:

$$\text{Min} \quad X_2 = \sum_{j=1}^{N} \left| \sum_{i=1}^{k-1} T_i^e + \sum_{i=k}^{N_j} T_{ij}^e - T_j^r \right| \tag{2}$$

3. Maximizing the total satisfaction X_3
   This is to evaluate te average level of satisfaction. We have to be very careful because the mass service process is handled by a single FLSP and is seen a single process. Therefore the quantity of mass service processes and customization service processes are very different. We have

$$\text{Max} \quad X_3 = \frac{\sum_{i=1}^{k-1} S_i + \sum_{j=1}^{N} \sum_{i=k}^{N_j} S_{ij}}{k - 1 + \sum_{j=1}^{N} (N_j - k + 1)} \tag{3}$$

4. Constraints
   Equations (1), (2) and (3) are subject to the following constraints:

$$\begin{cases} T_j^r - c_{j1} \leq \sum\limits_{i=1}^{k-1} T_i^e + \sum\limits_{i=k}^{N_j} T_{ij}^e \leq T_j^r + c_{j2} & \forall j,\ \forall 1 \leq k \leq N_j \\ L_{i+1}^a \leq T_{ij}^e - T_{ij} \leq L_{i+1}^p & \forall i, j \\ T_i^e \in [a_i, b_i] & \forall i < k \\ T_{ij}^e \in [a_i, b_i] & \forall i \geq k, \forall j \end{cases} \tag{4}$$

Now we have formulated the the problem formally as a multi-objective programming. To solve a multi-objective programming problem, we have to synthesize the three goals to convert the multi-objective problem into a single-objective one.

To do the converting we adpot the idea of *Ideal Point Method.* That is, we set up an ideal solution point, where the values of three objective functions are respectively their optimal values $X_{1\min}$, $X_{2\min}$ and $X_{3\max}$ when configured that one of them is the only objective of the programming. Then we derive the single objective X as the Euclidean distance from the present point to the ideal point to minimize:

$$\text{Min} \quad X = \sqrt{w_1 \cdot (1 - \frac{X_{1\min}}{X_1})^2 + w_2 \cdot (1 - \frac{X_{2\min}}{X_2})^2 + w_3 \cdot (1 - \frac{X_3}{X_{3\max}})^2} \tag{5}$$

But still this programming is not an LP (Linear Programming) nor ILP (Integer Programming). The constraints of programming are linear, yet the objective function is non-linear. And it is easy to see that it is impossible to convert the problem into LP, since the Uniform Distribution involves calculating expectations, which is an integral.

3.3 Problem Analysis and Algorithm Design

So far we have successfully formulated the original vague and ambiguous descriptions into a clear and well-defined optimizing programming problem. And we claim this problem to be in NP class.

Proof of claim. The set of decision variables T_{ij}^e and a value k of CODP act as a certifier of this problem. And the certificate is to compute the values in Equation /eqrefconstraint and check whether the constraints are satisfied. These are all summations and comparisons that can be done in linear time. Therefore the certifer is polynomial time. According to the definition of NP class, this problem is in NP.

But as we can see, due to the nature of this problem, this is a complex programming problem. What we have learned in course mainly focus on LP and ILP, but does not give us a clue on how to solve complex programming problems. We viewed many literature online and find most algorithms solving multi-objective programming to be heuristic algorithms, which does not guarantee the quality of solutions obtained. It is hard to design a non-heuristic algorithm to solve problem of this difficulty level efficiently. The only option seems to be submitting the problem constraints and objective to optimizers like CPLEX and hoping its inner opimizing algorithms will work.

To deal with this obstacle, we choose to step a little backward by modifying one of our assumptions, Assumption 9 :

9. Each process handled by an FLSP takes some time to complete, and the completion time is a time window. To be specific, **the completion time of process i is assumed to be a Dicrete Random Distribution $\{a_i, a_i + 1, \cdots, b_i - 1, b_i\}$. The completion time equally likely equals to all possible values.** Here we assume that selecting i as a mass or customization service process does not change the distribution, which is diffrent with in Section 4.1. For different FLSPs, their completion time is independent. We denote t_{ij} the actual completion time of process i in the jth order, where we have $\Pr[t_{ij} = x] = \frac{1}{b_i - a_i + 1}$, $x \in \mathbb{Z}$, $a_i < x < b_i$. And we have $\mathbb{E}(t_{ij}) = T_{ij} = \frac{a_i + b_i}{2}$. If process i is selected as a mass service, we denote t_i the actual completion time it, $\Pr[t_i = x] = \frac{1}{b_i - a_i + 1}$, $\mathbb{E}(t_i) = T_i = \frac{a_i + b_i}{2}$.

By adopting this modification we are not simplifying the problem—intuitively discrete version of a problem is no simpler than the continuous version, like ILP to LP. This is because we have no idea how to optimize continuous conditions. Actually we have not learned the content of Simplex or Kramarkar's Algorithm either, we only tried to use CPLEX to solve problems back then. But with discrete, integer-value problems, we are able to take advantage of **Local Search** and try designing our own method.

Before we demonstrate how we implement Local Search, we first modify the first objective to adapt to the modification in assumptions. The new total cost X_1 takes shape

$$\text{Min} \quad X_1 = \sum_{i=1}^{k-1} \sum_{x=a_i}^{b_i} (c_i \cdot t_i - |t_i - T_i^e| \cdot P_i) \Pr[t_i = x] + \sum_{j=1}^{N} \sum_{i=k}^{N_j} \sum_{x=a_i}^{b_i} (c_{ij} \cdot t_{ij} - |t_{ij} - T_{ij}^e| \cdot P_{ij}) \Pr[t_{ij} = x] \tag{6}$$

Then we define the **neighborhood** $\mathcal{N}$ of a current solution $S = (T^e, k)$, where

$$T^e = \left\{T_1^e, \cdots, T_{k-1}^e, T_{k1}^e, \cdots, T_{N_N N}\right\}, \tag{7}$$

$$\mathcal{N}(S) = \left\{ S' = (T^{e'}, k) \;\middle|\; \exists i, j : T^{e'} \setminus T^e = T_i^e \pm 1 \;\; or \;\; T_{ij}^e \pm 1 \right\} \tag{8}$$

Then in Algorithm 1 we give our approximate scheduling algorithm by doing local search on $\mathcal{N}(S)$.

Algorithm 1: Local Scheduling

Input: N, the set of all N_j, the set of all T_j^r, c_{j1}, c_{j2};
Output: A solution $S = (T^e, k)$;

$S_{\min} \leftarrow$ MAX;
for $k \leftarrow 1$ *to* $\min N_j$ **do** // iterate over all CODP value
  foreach i, j **do**
    $T_{ij}^e \leftarrow \max\left\{L_{i+1}^a + T_{ij}, a_i\right\}$; `// initialize the solution, start from where the latter 3 constraints are satisfied`
  end
  foreach $S' \in \mathcal{N}(S)$ **do**
    if S *satisfies all constraints* **then**
      break; `// feasible solution found, now start optimizing`
    end
    if $X_2(S') < X_2(S)$ *and* S' *satisfies the latter 3 constraints* **then**
      $S \leftarrow S'$; `// move to more feasible solutions`
    end
  end
  if S *does not satisfies all constraints and* $k = 1$ **then**
    return *"No feasible solution"*; `// tells if the instance does not have s feasible solution`
  else
    return $S_{\min}$; `// larger` k `cannot have a feasible solution`
  end
  foreach $S' \in \mathcal{N}(S)$ **do**
    if $X(S') < X(S)$ **then**
      $S \leftarrow S'$;
    end
  end
  if $X(S) < X(S_{\min})$ **then**
    $S_{\min} \leftarrow S$
  end
end
return $S_{\min}$;

3.4 Theoretical Analysis and Performance Evaluation

Theoretical Analysis

– **Time Complexity**
For each k, our algorithm does two things:

1. setting up a solution that is close to be a feasible one, and move to more feasible ones in its neighborhood, until we find a feasible solution. Or in the end there is no 'more feasible' solutions in the neighborhood, then we can tell that the problem instance does not have a feasible solution. The initializing part costs $\mathrm{O}(N \cdot \max N_j)$, and while moving, with each iteration $X_2(S)$ decreases at least 1. Let us denote l to de the largest length of the completion time window, then it at most moves $\mathrm{O}(N \cdot \max N_j \cdot l)$ times, with each move it compares with at most $\mathrm{O}(2N \cdot \max N_j)$ solutions in its neighbor, with each compare costing $\mathrm{O}(N \cdot \max N_j)$ time to compute $X_2(S')$. So moving costs $\mathrm{O}(N^3 \cdot \max {N_j}^3)$. If we treat the total number of processes of all orders as the input n, then finding a feasible solution costs $\mathrm{O}(n^3 + n)$=$\mathrm{O}(n^3)$.

2. moving towards solutions in its neighbors that has smaller value of X. Consider the ideal point in $\mathbb{R}^3$: $(X_{1\min}, X_{2\min}, X_{3\max})$, and spot a point representing feasible solution in the space to form an imaginary cubiod. Then moving towards the ideal point draws out a 3-D path. The length of the cubiod is $\mathrm{O}(\max c_{ij} \cdot \max b_i \cdot n)$, the width is $\mathrm{O}(l \cdot n)$, and the height is $\mathrm{O}(l \cdot n)$ as well since for a certain k the denominator of the expression of X_3 is constant, and the smaller the difference is between all T_{ij}^e and T_{ij}, the bigger the satisfaction is. Since we follow the rule that X always decreases while moving, the length of path is at most 2-dimensional product of the cuboid's parameter, that is at most $\mathrm{O}(c^2 \cdot n^2)$ moves, where c is a constant like l. And in each move it compares with at most $\mathrm{O}(2n)$ solutions in its neighbor, with each compare costing $\mathrm{O}(3n)$ to compute $X(S')$. So the optimizing process costs $\mathrm{O}(6c^2 \cdot n^4)$.

And the outermost loop iterates at most N times which is $\mathrm{O}(\sqrt{n})$. In conclusion the time complexity of Local Scheduling is $\mathrm{O}(c^2 n^{\frac{9}{2}})$, which is in polynomial time w.r.t. input size.

– **Feasibility**

We claim that if for a instance of this problem there exists a feasible solution, then our algorithm can always find it; if there does not exist a feasible solution, then our algorithm will return "no feasible solution".

Proof of claim.

1. First half.
   Suppose not. Then the feasible solution must have smaller X_2 than that of the current solution, since we can easily see that X_2 and the first constraint decribles exactly the same thing. In that case according to our algorithm, we would move to the feasible solution.
2. Second half.
   Suppose not. That is, for a instance of this problem there does not exist a feasible solution, but our algorithm returns one solution but not "no feasible solution". This is impossible as all solution will fail when judging whether the 4 constraints are satisfied, then the "no feasible solution" must be triggered. □

Performance Evaluation Though it is hard to derive an approximation ratio, we can compare our algorithm's result with that of other widely-used, good-in-practice heuristic algorithm to see how good our algorithm is. Here we choose **genetic algorithm** as the benchmark, and assuming the result it provides to be optimal.

4 Scheduling Model Design with the Relationship between Time Windows of Supplier Operation and Customer Requirement

4.1 Model Modification

We will add new assumptions to the model.

1. **The customization service has a new uniform distribution $U_i \sim (a_{ij}, b_{ij})$.** When we design the scheduling model without considering the relationship between time windows of supplier operation and customer requirement, we assume that mass service's completion time set for a service process of FLSP obeys uniform distribution $U_i \sim (a_i, b_i)$. After considering the relationship between time windows of supplier operation and customer requirement, since the time windows and customer requirement time provided by FLSP will affect each other, we set the customization service obeys a new uniform distribution $U_i \sim (a_{ij}, b_{ij})$. Then their relationship is expressed by setting a different uniform distribution between the mass service and the customization service.
2. **We add an extra cost based on the original cost.** Because when we don't consider the relationship between time windows of supplier operation and customer requirement, each service has a uniform distribution $U_i \sim (a_i, b_i)$, so the original cost has no extra cost. After we consider their relationship, the time schedule given by LSI may exceed the distribution range provided by FLSP, so LSI must pay extra cost per unit time which out of range.

After adding the above assumptions, we combine the previously established models to re-analyze the problem taking into account the relationship between time windows of supplier operation and customer requirement.

4.2 Problem Formulation

After modifying the model, we will re-improve this multi-objective programming problem:

1. Minimizing the total cost X_1
   We add an extra cost to the customization service in X_1 and change its distribution from (a_i, b_i) to (a_{ij}, b_{ij}). The total extra cost paid by LSI to FLSPs, which as compensation beyond the given distribution range of the FLSPs.
   If $T_{ij}^e < a_{ij}$, we have:

$$\begin{aligned}\text{Min} \quad X_1 = &\sum_{i=1}^{k-1} \int_{a_i}^{b_i} (c_i \cdot t_i - |t_i - T_i^e| \cdot P_i) f(t_i) dt_i \\ &+ \sum_{j=1}^{N} \sum_{i=k}^{N_j} \int_{a_{ij}}^{b_{ij}} (c_{ij} \cdot t_{ij} + (T_{ij} - T_{ij}^e) \cdot E_{ij} - |t_{ij} - T_{ij}^e| \cdot P_{ij}) f(t_{ij}) dt_{ij}\end{aligned} \tag{9}$$

   If $T_{ij}^e > b_{ij}$, we have

$$\begin{aligned}\text{Min} \quad X_1 = &\sum_{i=1}^{k-1} \int_{a_i}^{b_i} (c_i \cdot t_i - |t_i - T_i^e| \cdot P_i) f(t_i) dt_i \\ &+ \sum_{j=1}^{N} \sum_{i=k}^{N_j} \int_{a_{ij}}^{b_{ij}} (c_{ij} \cdot t_{ij} + (T_{ij}^e - T_{ij}) \cdot E_{ij} - |t_{ij} - T_{ij}^e| \cdot P_{ij}) f(t_{ij}) dt_{ij}\end{aligned} \tag{10}$$

   If $a_{ij} \le T_{ij}^e \le b_{ij}$, we have

$$\text{Min} \quad X_1 = \sum_{i=1}^{k-1} \int_{a_i}^{b_i} (c_i \cdot t_i - |t_i - T_i^e| \cdot P_i) f(t_i) dt_i + \sum_{j=1}^{N} \sum_{i=k}^{N_j} \int_{a_{ij}}^{b_{ij}} (c_{ij} \cdot t_{ij} - |t_{ij} - T_{ij}^e| \cdot P_{ij}) f(t_{ij}) dt_{ij} \tag{11}$$

2. Minimizing the total time difference X_2
   This part is the same as the above definition, we don't need to change it.

$$\text{Min} \quad X_2 = \sum_{j=1}^{N} \left| \sum_{i=1}^{k-1} T_i^e + \sum_{i=k}^{N_j} T_{ij}^e - T_j^r \right| \tag{12}$$

3. Maximizing the total satisfaction X_3
   Since we calculate the satisfaction after finding the corresponding time scheduling, the definition of this value will not change.

$$\text{Max} \quad X_3 = \frac{\sum_{i=1}^{k-1} S_i + \sum_{j=1}^{N} \sum_{i=k}^{N_j} S_{ij}}{k - 1 + \sum_{j=1}^{N} (N_j - k + 1)} \tag{13}$$

4. Constraints
   Equations (9) to (13) are subject to the following constraints:

$$\begin{cases} T_j^r - c_{j1} \le \sum\limits_{i=1}^{k-1} T_i^e + \sum\limits_{i=k}^{N_j} T_{ij}^e \le T_j^r + c_{j2} & \forall j \\ L_{i+1}^a \le T_{ij}^e - T_{ij} \le L_{i+1}^p & \forall i, j \end{cases} \tag{14}$$

   Since we now consider the extra cost, the constraints no longer limit the range of T_i^e and T_{ij}^e.
   Now we have formed a new multi-objective programming. The same reason, we have to synthesize the three goals to convert the multi-objective problem into a single-objective one. Here we use the same method as before, we derive the single objective X as the Euclidean distance from the present point to the ideal point to minimize:

$$\text{Min} \quad X = \sqrt{w_1 \cdot (1 - \frac{X_{1\min}}{X_1})^2 + w_2 \cdot (1 - \frac{X_{2\min}}{X_2})^2 + w_3 \cdot (1 - \frac{X_3}{X_{3\max}})^2} \tag{15}$$

4.3 Problem Analysis and Algorithm Design

In the previous problem we came to the conclusion that it is difficult to calculate a feasible solution with a continuous distribution, so we use discrete values. Similarly, after considering the relationship between Time Windows of Supplier Operation and Customer Requirement, we still use discrete values to calculate feasible solution. Since the algorithm we designed can adapt to the distribution of different completion times, we only need to modify some distribution values on the original algorithm to recalculate the feasible solution.

After we improve the model, the algorithm also modify the distribution of the input and reduce some constraints. Since our algorithm adapts to different distributions, modifying the distribution does not cause a change in computation time, and reducing the constraint only reduces the condition of the judgment. The overall computational complexity does not change much, so the algorithm can still be obtained in polynomial time. As a result, this problem can still be designed as an NP problem.

Obviously, the pseudo code we designed when we don't consider the relationship between Time Windows of Supplier Operation and Customer Requirement can be used in this problem. We only need to change the input of some distributions and some constraints. The overall idea of the algorithm does not change much, so there is no need to redesign the pseudo code.

4.4 Theoretical Analysis and Performance Evaluation

Theoretical Analysis

Performance Evaluation

5 Strength and Weakness

5.1 Strength

(1) The model we design considers many uncertain factors, and it can be suitable to most of the FLSPs operation time and customer requirement time.
(2) We have simplified the problem and correctly clarified the relationship between the various factors, so the results we get are simple and effective.
(3) Our algorithm can solve the detection problem in polynomial time, so it is no matter if your input size is large or small.
(4) We perform a sensitivity analysis on the model and obtain good results. The data derived from the model is highly similar to the given data.

5.2 Weakness

(1) Our algorithm gets a good feasible solution, not an optimal solution!
(2) Our model considers the expectations given by functional logistics service providers, so the results may differ from the actual time given.
(3) The algorithm we designed can only support a fixed number of customer orders arrived at the same time. When the number of orders changes, we need to find a new time schedule.
(4) For a small amount of special data, the algorithm may not get good results due to the simplification of the model.

5.3 Future Work

(1) Given the realities, we need to improve the algorithm to make timely adjustments to different numbers of customer orders and to handle multiple orders for one customer.
(2) Although our algorithm is polynomial time, it does not run very fast on large data inputs. Maybe we can reduce its time complexity by improving some parts of the algorithm and enable it to adapt to more complex data.

6 Conclusion

In order to obtain a better approximate solution, we turn the continuous problem into discrete. We use the traversal method and use local search to obtain the result. The obtained solution has a high similarity with the example given by the problem. In addition, our algorithm is generally completed in polynomial time. It is convenient to get the results when calculating the problem given by the project, and it has good processing time for different input sizes.

Regardless of whether or not to consider the relationship between Time Windows of Supplier Operation and Customer Requirement, the model we design can analyze problems effectively and get the right results. It may not achieve optimality on all three goals, but try to make each target close to its optimal solution.

All in all, we have designed a good time scheduling for the orders given by our customers to meet the various needs as much as possible, and the results of our model confirm this.

Acknowledgement

First of all, I would like to thank some of the students for their suggestions and the assistants' explanation of the problem. When we did this project, our team encountered a lot of trouble in understanding the problem. We feel that the problem is somewhat vague and even contradictory in the definition of some concepts. For these reasons, We have explored it for a long time.

Of course, we have learned a lot of useful knowledge through the analysis of this problem. In the process of solving the problem, we have a deeper understanding of the multi-objective programming problem. In general, this is a difficult but meaningful project.

References

1. Weihua Liu, Xuan Zhao, and Runze Wu: **Revenue-Sharing Contract Models for Logistics Service Supply Chains with Mass Customization Service**. Hindawi Publishing Corporation, Mathematical Problems in Engineering. Volume 2015, Article ID 572624, 21 pages.
2. Weihua Liu, Yi Yang, Haitao Xu, Xiaoyan Liu, Yijia Wang, and Zhicheng Liang: **A Time Scheduling Model of Logistics Service Supply Chain Based on the Customer Order Decoupling Point: A Perspective from the Constant Service Operation Time**. Hindawi Publishing Corporation, The Scientific World Journal. Volume 2014, Article ID 756178, 22 pages.
3. Weihua Liu, Yi Yang, Shuqing Wang, and Enze Bai: **A scheduling model of logistics service supply chain based on the time windows of the FLSP's operation and customer requirement**. Published online: 28 January 2015.
4. Wang, Y., &Lin, Jie. (2008). **Supply chain model based on multi-CODP in mass dynamic customization**. International Conference on Information Management, Innovation Management and Industrial Engineering, 2,252-255.
5. E. Selvarajah and G. Steiner. **Approximation algorithm for the supplier's supply chain scheduling problem to minimize delivery and inventory holding costs**. Operations Research, vol. 57, no. 2, pp. 426-438, 2009.
6. W. H. Liu, X. C. Xu, Z. X. Ren, and Y. Peng. **An emergency order allocation model based on multi-provider in two-echelon logistics service supply chain**. Supply Chain Management, vol. 16, no. 6, pp. 391-400, 2011.