Iterated Local Search for the Workload Balancing Problem in Service Enterprises

Thanh-Ha Nguyen*, Mike Wright

Lancaster University Management School, Lancaster, LA1 4YX, UK

Abstract

In this paper, we consider a telecommunication service company facing seasonal demand and time-varying capacity. A uniform lead-time, which is the maximum time span a customer has to wait before receiving the required service, is quoted to all customers. We present a quadratic integer programming model for the problem of scheduling jobs to meet the promised lead-time with the objective of balancing the workload across time. Since in practice solving such a problem to optimality can be very difficult, two variants of an iterated local search approach are proposed, which iteratively apply local search to perturbations of the current search point, leading to a randomized walk in the space of local optima, instead of sampling the space of all possible candidate solutions. These heuristics may be seen as simple variants of variable neighborhood search, which, in its basic form, exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them. Extensive computational tests show that our heuristics are able to provide high quality solutions efficiently.

Keywords: Scheduling, Quadratic integer programming, Iterated local search, Variable neighborhood search, Meta-heuristic, Capacity management

*Corresponding author

Email addresses: n.thanh-ha@lancaster.ac.uk (Thanh-Ha Nguyen), m.wright@lancaster.ac.uk (Mike Wright)

1. Introduction

In the telecommunications industry, a common practice is to use uniform lead time, which is the maximum time span a customer has to wait before receiving the required service. For services (e.g. installation of broadband at telecom exchanges) where customers do not have to participate in the service delivery process, the advantages of quoting a uniform lead time are twofold. On the one hand, all customers are guaranteed a uniform delivery lead time. On the other hand, the firm can flexibly choose the best time within the promised lead-time to carry out the work. We consider a telecommunication operator with seasonal demand for such a service. The available capacity per period also follows a cyclic pattern. In order to improve service efficiency, i.e. to avoid unnecessary capacity over-utilization in certain periods and under-utilization in other periods, the workload must be balanced across time. Thus, we are interested in how to match the firm's capacity to customer demand in such a way that: a) the quoted lead-time is satisfied and b) the capacity utilization rate per period is distributed as equally as possible along the time line. This problem will be referred to as the Workload Balancing Problem (WBP) in the rest of the paper.

A clear definition for the measure of balance is necessary for the study of workload balancing. Naturally, variance, standard deviation or *sum of squared deviations (SSD)* are good measures, as they tend to penalize larger deviations at a higher rate. We model the WBP in terms of minimizing the SSD of capacity utilization rates across time periods from their mean. The WBP can be seen as a Quadratic Integer Programming Problem (QIP or QIPP) due to the formulation of the problem as an integer problem with a quadratic objective function. The QIP is, in general, difficult to solve. Hence mathematical programming techniques may fail to deliver exact solution in reasonable time. To practically solve the QIP, heuristic algorithms which find high quality solutions in short computation time have been proposed. Such heuristic algorithms are iterated local search [1], variable neighborhood search [2, 3, 4], simulated annealing [5, 6, 7], tabu search [8, 9, 10], genetic algorithms [11, 12, 13, 14, 15], evolution strategies [16, 17], ant algorithms [18, 19, 20, 21], and scatter search [22, 13]. Among them, iterated local search (ILS) is a simple and powerful stochastic local search method for solving combinatorial problems. ILS is reported to be among the best performing algorithms for a number of problems such as the Traveling Salesman Problem (TSP)[23] and the Quadratic Assignment Problem (QAP)[1]. The latter is

NP-hard and is considered as one of the hardest QIP [24]. The QAP can be described as the optimization problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities in order to minimize the sum of the product between flows and distances. The test results in [1] show that ILS algorithms have excellent performance when compared to robust tabu search and MAX-MIN ant system, which are known to perform well for the QAP [19, 25]. For this reason, we propose two variants of an ILS method to tackle the WBP, which iteratively apply local search to perturbations of the current search point, leading to a randomized walk in the space of local optima, instead of sampling the space of all possible candidate solutions. Thus, ILS may be seen as a simple variant of variable neighborhood search, which in its basic form, exploits *systematically* the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them[2].

The WBP was only once investigated by investigated by Li and He [26]. They proposed two greedy local search algorithms to tackle the problem. However, these algorithms are time consuming and because of the greedy strategy, they can often get stuck in local optima. Another problem is that without providing a mathematical model and thus an exact method, in their experiments, the previous authors were only able to compare CPU times, but not obtain any information about solution quality in terms of loss of optimality.

This paper proposes several improvements to the previously published work by Li and He [26] as follows: Section 2 presents a mathematical model for the WBP, based on which an exact solution can be derived. In Section 3, besides an exact method, two new heuristics based on ILS are proposed to improve search speed and solution quality. In Section 4, to illustrate the proposed heuristic algorithms, a concrete numerical example is provided. Section 5 shows computation results for 140 test problem instances. Section 6 concludes and discusses potential extensions and directions for further research.

2. Problem formulation

2.1. Assumptions

We consider a major telecommunication service company. The company is using an integrated planning system, based on hierarchical planning concepts that allow to decompose the entire planning problem into partial plan-

Table 1: Notation for problem formulation

M:	length of one demand-cycle
N:	length of one capacity-cycle
τ:	finite planning horizon, $\tau = lcm(M, N)$
I:	set of job arrival dates, $I = \{1, \ldots, \tau\}, i \in I$
J:	set of job completion dates, $J = \{1, \ldots, \tau\}$, $j \in J$
J_i:	set of feasible completion dates of jobs arriving in period i
λ_i:	demand in period i
c_j:	available capacity in period j
x_{ij}:	number of jobs that arrive at period i and are assigned to completion date j
X:	assignment scheme $X = [x_{ij}]_{\tau \times \tau}$
ℓ:	the minimum time span a job has to wait before it can be processed
L:	uniform lead-time, $\ell < L \leq \tau + \ell$
u_j:	used capacity in period j
μ_j:	capacity utilization rate in period j
$\overline{\mu}$:	mean of capacity utilization rates during the planning horizon τ
r_i:	internal release date of jobs that arrive in period i
d_i:	due date of jobs that arrive in period i

ning tasks but to still consider their interdependencies and to coordinate their solutions. This planning system consists of different modules such as demand forecasting, resources planning, work scheduling, which are interlinked with each other. It makes use of solution approaches known as mathematical programming and meta-heuristics and provides supports at different levels for planning tasks along the company's service chain, from long-term strategic decision making to short-term operational decisions. The levels for planning may overlap or may be distinct. Either way there is a flow of information from strategic to operational planning and then to operational planning and vice versa.

Demand. The firm faces seasonal demand for a particular service, e.g. broadband installation. The estimated demand data are provided by the responsible module for forecasting demand. Thus, the start and the end point of the seasonal cycle are known. Further, the seasonal demand pattern, which repeats itself every cycle, is also given. By dividing the seasonal cycle into time periods $1, \ldots, M$, the demand pattern can be expressed by a vector $[\lambda_1, \ldots, \lambda_M]^T$. Each element of this vector represents the demand (measured by the number of jobs) that occurs through a particular time period.

Capacity:. The firm has a fixed number of permanent employees and a number of seasonal technicians with repeated fixed term contracts. The latter are retained in order to meet peaks in demand (e.g. surge of demand for broadband installations at the beginning of school terms). The information concerning availability of the workforce per time period is provided by means of the medium-term, anticipatory deployment plan. As it is possible to estimate the average time a technician needs to complete a job, we represent capacity during a time period in terms of the number of jobs to better match it with customer demand. Capacity levels are assumed to follow a cycle of N time periods with the pattern $[c_1, \ldots, c_N]^T$.

Planning horizon. The planning horizon τ is the minimum time interval after each of which both demand and capacity pattern will repeat themselves. Thus, τ is determined as the least common multiplier of M and N, $\tau = lcm(M, N)$.

Lead time. Uniform lead-time L is quoted to all customers. We assume that L is bounded by $\ell+1$ and $\tau+\ell$ ($\ell < L \leq \tau+\ell$), where ℓ denotes the minimal time span a job has to wait before it can be processed.

Job. All jobs are the same and can be completed within one time period. Each job is characterized by its arrival i and the time period j, when it is completed.

For the ease of notation, let I and J where $I = J = \{1, \ldots, \tau\}$ denote the set of job arrival dates and job completion dates respectively. The demand and the available capacity during the planning horizon are represented by the vectors $\lambda = [\lambda_i]$ and $c = [c_j]$, where $i \in I$ and $j \in J$. The vector λ is obtained by τ/M-times concatenation of $[\lambda_1, \ldots, \lambda_M]^T$, and the vector c by τ/N-times concatenation of $[c_1, \ldots, c_N]^T$.

We want to find a job assignment scheme $X = [x_{ij}]_{\tau \times \tau}$ that determines how many jobs of each demand λ_i are to be completed in which time period j, so that all demands are met within L periods and the workload is balanced over time. In an ideal case, the capacity utilization rate in every time period would be the same.

In the following, the lead-time constraints and the objective function of the WBP are specified. Note that, an explicit formulation of capacity constraints is not needed. The reasons are as follows: (a) If the total capacity is sufficient to accommodate the total demand over τ periods (that is, $\sum_{i \in I} \lambda_i \leq \sum_{j \in J} c_j$) and the quoted lead-time L is long enough, the objective function will ensure that workload is distributed across periods as equally as possible and the capacity utilization rate in each period is less or equal 100%. (b) Otherwise, the firm will not be able to accommodate the total demand within the promised lead-time without exceeding the total capacity. In this case, the firm is better off increasing capacity by hiring more labor or outsourcing part of its service activities, but modeling these aspects is out of the scope of this paper.

2.2. Lead-time constraints

We use internal release and due dates to indicate the time window in which a job must be completed. The internal release date is the earliest possible time a job can start. The internal release date $r_i \in \{1, \ldots, \tau\}$ of jobs, that arrive in period i, is computed as

$$r_i = (i + \ell + 1) \bmod \tau \tag{1}$$

The modulo operation (mod τ) is used here and in the rest of the paper to model the cyclical behavior of demand and capacity that recurs every τ periods. The due date $d_i \in \{1, \ldots, \tau\}$ of jobs arriving in period i is the latest time these jobs must be completed, and is calculated as arrival time plus quoted lead-time.

$$d_i = (i + L) \bmod \tau \tag{2}$$

Note that if $r_i < d_i$, the due date d_i simply represents the d_i-th time period in the same planning horizon as the internal release date r_i. Otherwise, if $r_i \geq d_i$, d_i represents a time period in the subsequent planning horizon.

Example If the considered planning horizon τ is a week, the quoted lead-time L is 5 days, and $\ell = 0$, for jobs arriving on Thursday (day 4), we calculate $r_i = (4 + 1) \bmod 7 = 5$ and $d_i = (4 + 5) \bmod 7 = 2$. That is, the internal release date is Friday of the current week and the due date is Tuesday of the following week. In this case $r_i > d_i$. Another example, if a job arrives on Monday (day 1), we have $r_i = 2$, $d_i = 6$ and $r_i < d_i$.

To satisfy the guaranteed lead-time, the completion date j of any job that arrives in period i must be within the time intervals below:

$$J_i = \begin{cases} \{r_i \dots d_i\} & \text{if } r_i < d_i \\ \{r_i \dots \tau\} \cup \{1 \dots d_i\} & \text{if } r_i \geq d_i \end{cases} \tag{3}$$

Hence, a feasible assignment scheme X must satisfy the condition: $x_{ij} = 0$ for any $i \in I$ and $j \notin J_i$, or equivalently

$$\sum_{i=1}^{\tau} \left(\sum_{\substack{d_i < j \leq \tau \\ r_i < d_i}} x_{ij} + \sum_{\substack{1 \leq j < r_i \\ r_i < d_i}} x_{ij} + \sum_{\substack{d_i < j < r_i \\ r_i \geq d_i}} x_{ij} \right) = 0 \tag{4}$$

2.3. Objective function

In order to formulate the objective function $f(X)$ that minimizes the SSD of capacity utilization rates, we first compute the used capacity $u_j(X)$ and the capacity utilization rate $\mu_j(X)$ in each period $j \in J$. For notation simplicity, $u_j(X)$ and $\mu_j(X)$ will be written as u_j and μ_j.

The used capacity u_j in period j is determined by adding all the allocated jobs in this period, and can be mathematically written as

$$u_j = \sum_{i=1}^{\tau} x_{ij} \tag{5}$$

The capacity utilization rate μ_j in period j is the used capacity in that period divided by the available capacity in the same period.

$$\mu_j = u_j / c_j = \sum_{i=1}^{\tau} x_{ij} / c_j \tag{6}$$

Let $\bar{\mu}$ denote the average utilization rates during τ periods, the objective

function is given by

$$f(X) = \sum_{j=1}^{\tau} [\mu_j - \bar{\mu}]^2 = \sum_{j=1}^{\tau} \mu_j^2 - \tau \cdot \bar{\mu}^2 \tag{7}$$

Combining Equations (6) and (7) and $\bar{\mu} = \frac{1}{\tau} \sum_{j=1}^{\tau} \mu_j$ we can write the objective function as directly depending on $X = [x_{ij}]_{\tau \times \tau}$

$$f(X) = \sum_{j=1}^{\tau} \left(\frac{\sum_{i=1}^{\tau} x_{ij}}{c_j} \right)^2 - \frac{1}{\tau} \left(\sum_{j=1}^{\tau} \frac{\sum_{i=1}^{\tau} x_{ij}}{c_j} \right)^2 \tag{8}$$

2.4. Mathematical model

The model for the workload balancing problem WBP can be formulated as

$$\textbf{(WBP)} \quad \min_X f(X) \tag{9}$$

$$s/t \quad \sum_{j=1}^{\tau} x_{ij} = \lambda_i \qquad \forall i \in I \tag{10}$$

$$\sum_{i=1}^{\tau} \left(\sum_{\substack{d_i < j \leq \tau \\ r_i < d_i}} x_{ij} + \sum_{\substack{1 \leq j < r_i \\ r_i < d_i}} x_{ij} + \sum_{\substack{d_i < j < r_i \\ r_i \geq d_i}} x_{ij} \right) = 0 \tag{4}$$

$$r_i = (i + \ell + 1) \bmod \tau \qquad \forall i \in I \tag{1}$$

$$d_i = (i + L) \bmod \tau \qquad \forall i \in I \tag{2}$$

$$x_{ij} \in \mathbb{N}_0,\ i \in I,\ j \in J \tag{11}$$

Some explanations shall be given as follows. The decision variable is $X = [x_{ij}]_{\tau \times \tau}$. The objective (9) is to minimize the SSD of capacity utilization rates over τ periods. Constraints (10) force that all demands are to be filled. Constraints (4) secure that each job can only be processed after the internal release date, and has to be completed within the quoted lead-time. Equations (1) and (2) compute the internal release date and the due date of any job

respectively. Finally variables are restricted to be integer in (11).

3. Solution approaches

3.1. Exact method

We use CPLEX (see http://www.ilog.com/ for details) to solve the WBP exactly for small-scale problems. The WBP can be written in the following standard matrix form, as required by CPLEX

$$\begin{aligned} \min_{y} \quad & f(y) = \frac{1}{2} y^T F y \\ s/t \quad & Ay = b \\ & y^- \leq y \leq y^+, \quad y_k \in \mathbb{N} \quad \forall k \in \{1, \ldots, \tau^2\} \end{aligned}$$

where $F \in \mathbb{R}^{\tau^2 \times \tau^2}, A \in \mathbb{N}^{\tau \times \tau^2}$ and $y^-, y^+ \in \mathbb{N}^{\tau^2 \times 1}$. The decision variable y is bounded by the lower bound y^- and upper bound y^+. Each element y_k of the vector y corresponds to one and only one element x_{ij} of the allocation matrix X defined in the previous section, with k satisfying $k = i \cdot \tau + j$. The vector y has τ^2 elements, but only $\tau \cdot (L - \ell)$ out of them are positive. All other elements are set to zero according to the lead-time constraints specified in Section 2.2. Even so, in practice, solving such a quadratic problem to optimality can be very difficult. Our experiments show that, unfortunately, CPLEX may take several hours on a modern workstation to obtain a globally optimal solution even for small-scale problems. Problems of medium or large size cannot be solved by CPLEX, the solver usually runs out of memory.

3.2. Heuristics

3.2.1. Notations and definitions

For solving the WBP, Li and He [26] propose two greedy search algorithms named Single Day Shift (H1) and Multiple Day Spread (H2). Their experiments show that H2 outperforms H1 in run-time performance, but do not reveal any insight into the solution quality of the heuristics in terms of loss of optimality. The basic idea is to shift jobs within the valid time window in order to achieve workload balancing. Following this idea, in Section 3.2.2, we design and develop two new local search procedures S-Shift and M-Shift, which we then embed separately into a metaheuristic framework based on iterated local search (ILS) in Section 3.2.3. Two variants are developed,

S-ILS and M-ILS, which use S-Shift and M-Shift in their local step, respectively. The new algorithms improve the run time and the solution quality significantly as exhibited by the numerical experiments reported in Section 5. Before discussing these algorithms, we introduce some further notations and definitions.

Table 2: Additional notation for S-Shift and M-Shift heuristic

j :	source period, current completion date
t :	target period, completion date after a shift
J_j:	set of feasible target periods t, which jobs from source period $j \in J$ can be shifted to
I_j:	set of possible arrival dates of jobs that can be completed in period $j \in J$
I_{jt}:	set of possible arrival dates of jobs that can be completed in periods $j \in J$ and $t \in J$
$S_j(t, \Pi)$:	shift of quantity Π of jobs from source period j to target period t
$O_{ij}(t, \pi_i)$:	shift operation of quantity π_i of jobs, from the cell $[i, j]$ to the cell $[i, t]$ in the assignment scheme X

Feasible assignment scheme. A feasible job assignment scheme X is a $\tau \times \tau$ matrix, that satisfies the constraints of the WBP (see Section 2.4).

Better assignment scheme. Given two feasible assignment schemes X and X', X' is better than X if X' has a lower sum of squares of the utilization rate across τ time periods, that is $f(X') < f(X)$ where f is defined in Equations (7) and/or (8).

Operation. An operation is described by the expression $O_{ij}(t, \pi_i)$, where O symbolizes the movement of quantity π_i of jobs from the cell $[i, j]$ to the cell $[i, t]$ in the matrix X. This means that, for π_i jobs with arrival time i, the completion date is changed from j to t. We will refer to j and t as source period and target period of shift respectively. An operation would

- reduce x_{ij} by π_i jobs

- increase x_{it} by π_i jobs

Shift. A shift $S_j(t, \Pi)$ is a set of operations $O_{ij}(t, \pi_i)$, which describes the shift of a total of Π jobs with different arrival times i, from the completion date j to another completion date t. Such a shift would

- reduce the used capacity u_j in source period j by Π jobs
- increase the used capacity u_t in target period t by Π jobs

Positions of jobs to be shifted. As the length of any feasible time window for completion is $L - \ell$ periods, jobs currently scheduled in period j can be moved at most $L - \ell - 1$ periods earlier or later. Hence, the range J_j of possible target periods, which jobs can be moved to is between periods t^- and t^+ where

$$t^- = [j - (L - \ell - 1)] \bmod \tau$$
$$t^+ = [j + (L - \ell - 1)] \bmod \tau$$

Thus,

$$J_j = \begin{cases} \{t^-, \ldots, t^+\} & \text{if } t^- < t^+ \\ \{1, \ldots, t^+\} \cup \{t^-, \ldots, \tau\} & \text{if } t^- > t^+ \end{cases} \tag{12}$$

To make sure that the solution obtained after an operation is feasible, we only select jobs to move that can be assigned to both completion days j and t without violating the lead-time constraints. Let i^- and i^+ denote, respectively, the earliest and the latest arrival time of jobs that can be "legally" completed in period j. i^- and i^+ can be computed as

$$i^- = (j - L) \bmod \tau$$
$$i^+ = (j - \ell - 1) \bmod \tau$$

Thus, the set I_j of all possible arrival times of jobs, for which period j is a feasible completion time is given by

$$I_j = \begin{cases} \{i^-, \ldots, i^+\} & \text{if } i^- < i^+ \\ \{1, \ldots, i^+\} \cup \{i^-, \ldots, \tau\} & \text{if } i^- > i^+ \end{cases} \tag{13}$$

The set of all possible arrival times of jobs, that can be completed in both periods j and t is therefore

$$I_{jt} = I_j \cap I_t \tag{14}$$

Quantity of jobs to be shifted. Suppose we want to move jobs from a source period j to a target period t, the maximum quantity of jobs to be shifted out of period j in a shift $S_j(t, \Pi)$ is defined as

$$\Pi_{max} = \left\lfloor c_j \left(\mu_j - \frac{u_j + u_t}{c_j + c_t} \right)^+ + 0.5 \right\rfloor \tag{15}$$

Equation (15) says that the shift aims to balance out the gap of capacity utilization rate between periods j and t, where $\frac{u_j+u_t}{c_j+c_t}$ represents the average capacity utilization rate in periods j and t. Further, the formula (15) is edited to round the result to the nearest integer. We denote $a^+ = \max\{a, 0\}$ and $\lfloor a + 0.5 \rfloor$ as the nearest integer to a.

Moving Π_{max} jobs may introduce infeasibilities. In order to avoid this problem, we shift no more than the total amount $\sum_{i \in I_{jt}} x_{ij}$ of jobs, which are currently assigned to completion date j and can be moved to another completion date t within the valid time window. Thus, the total quantity of jobs to be shifted is adjusted to

$$\Pi = \min \left\{ \left(\sum_{i \in I_{jt}} x_{ij} \right), \Pi_{max} \right\} \tag{16}$$

Feasible shift. Now we define a feasible shift as

$$S_j(t, \Pi) = \left\{ O_{ij}(t, \pi_i) \ : \ i \in I_{jt}, j \in J, t \in J_j, \pi_i \geq 0 \wedge \Pi = \sum_{i \in I_{jt}} \pi_i \right\} \tag{17}$$

Given j, t and Π for a shift $S_j(t, \Pi)$, the number of jobs to be shifted in an operation $O_{ij}(t, \pi_i)$ is determined by

$$\pi_i = \begin{cases} \min\{x_{ij}, \Pi\} & \text{if } i = \min I_{jt} \\ \min \left\{ x_{ij}, \left(\Pi - \sum_{\substack{k \in I_{jt} \\ k < i}} \pi_k \right)^+ \right\} & \text{if } i \in I_{jt} \setminus \{\min I_{jt}\} \end{cases} \tag{18}$$

The shift operations are carried out in ascending order of job arrival, that

is, first-arrive, first-shifted. Equation (18) ensures that in each operation, no more than the current number of jobs in cell [i, j] are shifted, and the total number of jobs shifted from source period j to target period t is exactly Π.

3.2.2. Local search methods S-Shift and M-Shift

S-Shift. The search iteratively considers all possible pairs of completion dates (j, t), where j has a higher capacity utilization rate than t, i.e. $\mu_j > \mu_t$. S-Shift starts with the pair that has the highest gap of utilization rates and continues in descending order of the gap $[\mu_j - \mu_t]$. For each pair (j, t), the algorithm checks whether t is a valid target period for a shift from j. If yes, i.e. $t \in J_j$, the total number of jobs to be shifted Π is determined using Equations (15) and (16). The set of shift operations $O_{ij}(t, \pi_i)$ to be processed is found using Equation (17). The shift operations are applied in ascending order of job arrival $i \in I_{jt}$, until the total quantity Π of jobs are completely moved from source period j to target period t. In each shift operation, π_i jobs with arrival i are to be reassigned from completion date j to completion date t, see Equation (18). The algorithm stops when a better assignment scheme X' is obtained, or after having gone though all possible pairs (j, t) and no better solution can be found.

M-Shift. M-Shift differs from S-Shift in that, whilst S-Shift moves jobs from a source period j to the best suitable target period t in each search iteration, M-Shift spreads jobs from the source period j to all feasible target candidates proportionally.

Algorithm 1 S-Shift

Require: λ, c, ℓ, L, X
Ensure: X'

Sort the set J of job completion dates j in descending order of utilization rate μ_j, into DRU
for $k = 1$ to $\tau - 1$ **do**
  Set $j \leftarrow DRU[k]$; {Source period}
  for $l = T$ to k **do**
    Set $t \leftarrow DRU[l]$; {Target period}
    if $t \in J_j$ **then**
      Set $u' \leftarrow u; u'_j \leftarrow u_j - \Pi; u'_t \leftarrow u_t + \Pi;$
      Set $\mu' \leftarrow u'./c; f(X') \leftarrow \sum_{j=1}^{\tau} \left(\mu'_j - \bar{\mu'}\right)^2$;{Compute $f(X')$}
      if $f(X') < f(X)$ **then**
        Set $m \leftarrow 1$;
        repeat
          Set $i \leftarrow I_{jt}[m]$;
          Set $x_{ij} \leftarrow x_{ij} - \pi_i; x_{it} \leftarrow x_{it} + \pi_i$; {Shift operation $O_{ij}(t, \pi_i)$}
          Set $\Pi \leftarrow \Pi - \pi_i; m \leftarrow m + 1$;
        until $\Pi = 0$;
        Set $X' \leftarrow X$;
        return
      end if
    end if
  end for
end for

Algorithm 2 M-Shift

Require: λ, c, ℓ, L, X
Ensure: X'
1: $existBetterAS \leftarrow 0$;
2: Sort the set J of job completion dates j in descending order of utilization rate μ_j, into DRU
3: **for** $k = 1$ to τ **do**
4: $j \leftarrow DRU[k]$; {Source period}
5: Sort the set J_j of feasible target periods t in ascending order of utilization rate μ_t, into LDS
6: **for all** $t \in LDS$ **do**
7: Set $u' \leftarrow u; u'_j \leftarrow u_j - \Pi; u'_t \leftarrow u_t + \Pi$;
8: Set $\mu' \leftarrow u'./c; f(X') \leftarrow \sum_{j=1}^{\tau} \left(\mu'_j - \bar{\mu'}\right)^2$;{Compute $f(X')$}
9: **if** $f(X') < f(X)$ **then**
10: $existBetterAS \leftarrow 1$;
11: $m \leftarrow 1$;
12: **repeat**
13: $i \leftarrow I_{jt}[m]$;
14: $x_{ij} \leftarrow x_{ij} - \pi_i; x_{it} \leftarrow x_{it} + \pi_i$; {Shift operation $O_{ij}(t, \pi_i)$}
15: $\Pi = \Pi - \pi_i; m \leftarrow m + 1$;
16: **until** $\Pi = 0$;
17: **end if**
18: **end for**
19: **if** $existbetterAS = 1$ **then**
20: Set $X' \leftarrow X$;
21: **return**
22: **end if**
23: **end for**

3.2.3. ILS heuristic

In order to escape from local optima, S-Shift and M-Shift are embedded, each separately, in an ILS framework. We denote the meta-heuristics as S-ILS and M-ILS respectively.

The three main components of the ILS heuristic are as below:

Initialization. Similar to the approach in [26], the initial solution in Step 1 is generated deterministically by allocating each daily demand λ_i across the time interval between the internal release date r_i and the due date d_i, as evenly as possible.

Algorithm 3 Variable neighborhood search S-ILS/M-ILS

Require: $\lambda, c, \ell, L, t_{max}, k_{max}$
Ensure: X_{opt}
1: *Initialization:* Select an initial solution X
2: set $X_{opt} \leftarrow X; f_{opt} \leftarrow f(X)$
3: **repeat**
4: Set $k \leftarrow 1$
5: **repeat**
6: *Exploration of neighborhood:* Find a better solution X' with $f(X') < f(X)$ using S-Shift or M-Shift;
7: **if** $f(X') < f_{opt}$ **then**
8: Set $X \leftarrow X'; X_{opt} \leftarrow X'; f_{opt} \leftarrow f(X'); k \leftarrow 1;$
9: **else**
10: *Perturbation:* Set $X \leftarrow perturbation(X_{opt}, k); k \leftarrow k + 1;$
11: **end if**
12: **until** k_{max}
13: **until** t_{max}

Exploration of neighborhood. The core of the search procedure is, of course, the exploration of the neighborhood. In each iteration, the search starts from a solution X and attempts to find a better solution X' using the local search heuristic S-Shift or M-Shift (see Section 3.2.2).

1. If no better solution X' than X can be found, a new starting point for the next iteration is randomly generated by perturbing the current best solution X_{opt}. In the following we denote $f_{opt} = f(X_{opt})$ as the objective function value of X_{opt}.
2. Otherwise, a random solution from the neighborhood of X_{opt} is generated using perturbation of X_{opt} to restart the next round of search.

Perturbation. ILS uses perturbation techniques to enable the search to escape the valley of a local optimum and find another better solution. In this step, a solution is generated by perturbing the current best, from which the search continues to explore the solutions landscape. The solutions are perturbed in a random way in order to avoid cycling which might occur if deterministic rules are used. First, a non-zero element $x_{ij} > 0$ is randomly chosen in the assignment scheme X. Second, a random target period $t \in J_j$ is determined. Third, the quantity π_i of jobs to be shifted is generated as an integer random number between 1 and x_{ij}. Finally, the perturbed solution is obtained after the shift operation $O_{ij}(t, \pi_i)$ which reassigns π_i jobs arriving in period i from completion date j to completion date t.

We increase the perturbation strength in a similar way as done in VNS [2] by repeating the described perturbation k times. We vary k between 1 and k_{max} starting at 1. If after the perturbation and the subsequent local search no better solution is found, k is increased by one, otherwise it is set to 1. If k_{max} is reached, we set k back to 1. The search stops when the maximum CPU time allowed t_{max} runs out.

4. Illustrative example

In this section, we present a numerical example to illustrate S-ILS and M-ILS.

Example We have $M = 7$ days, $N = 7$ days, $\tau = lcm(M, N) = 7$ days, $\ell = 2$ days. The weekly demand and capacity are given as

- $\lambda = [\lambda_i] = [524, 391, 523, 490, 372, 254, 126]^T$,
- $c = [c_j] = [480, 400, 260, 320, 370, 490, 480]^T$ respectively.

Further, L is set to 4 days.

Table 3: Initial solution

	j / i	1 Mon	2 Tue	3 Wed	4 Thu	5 Fri	6 Sat	7 Sun	λ_i
	1 Mon				262	262			524
→	2 Tue					196	195		391
	3 Wed						262	261	523
	4 Thu	245						245	490
	5 Fri	186	186						372
	6 Sat		127	127					254
	7 Sun			63	63				126
	u_j	431	313	190	325	458	457	506	
	c_j	480	400	260	320	370	490	480	
	μ_j	0.90	0.78	0.73	1.02	1.24	0.93	1.05	
	$f(X)$	0.1771				↑	↑		

In Table 3, the dark gray cells represent the time span a job has to wait before it can be processed. The light gray cells indicate the time window

Table 4: Comparisons between S-ILS and M-ILS by means of an example

Iter.	S-ILS					M-ILS				
	$j \to t$	i	π_i	Π	$f(X)$	$j \to t$	i	π_i	Π	$f(X)$
1	$5 \to 6$	2	64	64	0.11977	$5 \to 6$	2	64	64	0.11977
						$5 \to 4$	1	8	8	0.11922
2	$5 \to 4$	1	8	8	0.11922	$6 \to 7$	3	2	2	0.11920
3	$6 \to 7$	3	2	2	0.11920	$7 \to 1$	4	39	39	0.10663
4	$7 \to 1$	4	39	39	0.10633	$6 \to 7$	3	20	20	0.10309
5	$6 \to 7$	3	20	20	0.10309	$5 \to 6$	2	5	5	0.10221
46	$6 \to 7$	3	1	1	2.9×10^{-5}	$1 \to 2$	5	1	1	1.2×10^{-5}
47	$1 \to 2$	5	1	1	1.2×10^{-5}	–	–	–	–	–
61	$4 \to 5$	1	94	94	1.2×10^{-5}	$2 \to 1$	5	97	97	7.0×10^{-6}
328	$7 \to 6$	3	193	193	7.0×10^{-6}	$7 \to 6$	3	171	171	3.6×10^{-6}
810	$6 \to 7$	3	193	193	3.6×10^{-6}	$3 \to 2$	6	66	66	3.6×10^{-6}

between the internal release date r_i and the due date d_i of jobs arriving on day i. It can be seen that e.g. the feasible time window for completing jobs arriving on Monday, are Thursday and Friday. As another example, for any job arriving on Thursday, the completion date has to be Sunday of the same week as the arrival date, or Monday of the following week. In both heuristics, to create an initial solution, the daily demands are allocated across the feasible 2-days completion time interval respectively, as evenly as possible. Table 3 shows the initial state ($x_{14} = x_{15} = 262$; $x_{25} = 196$, $x_{26} = 195$ etc.). The used capacity per day is determined using Equation (5) $u = [u_j] = [431, 313, 190, 325, 458, 457, 506]^T$. According to Equation (6) the daily capacity utilization rates are calculated and can be written in the vector form $\mu = [\mu_j] = [0.90, 0.78, 0.73, 1.02, 1.24, 0.93, 1.05]^T$. Thus, the objective function value of the initial solution is 0.1771.

Starting from the initial solution, the search process of both heuristics is recorded. The CPU time limit t_{max} is set to 50 seconds, which is long enough for solving this small test instance to optimality. Here only the results of some search iterations are indicated in Table 4 to take up less space. In each of these iterations, source day j and target day t of each shift operation, as well

as arrival date and number of shifted jobs π_i are shown.

In the initial solution, Friday has the highest capacity utilization rate, Saturday has the lowest rate among the feasible target candidates, and only jobs arriving on Tuesday are feasible to be moved from Friday to Saturday. Therefore, the very first shift operation changes the completion date of 64 jobs arriving on Tuesday from Friday to Saturday.

Unlike S-ILS, M-ILS can spread jobs from a source day to more than one target day as can be seen in the first iteration, and thus converges faster than S-ILS. An example of neighborhood change can be seen in the 47th iteration, in which no better solution is found. Here, a solution generated at random is taken as the starting point for the next iteration to escape the valley of a local optimum. S-ILS obtains the best solution in the 810th, and M-ILS in the 328th round of search. The heuristics terminate when t_{max} is reached and no better solution is found. The best objective function values provided by S-ILS and M-ILS are both $3.6{\times}10^{-6}$ which correspond to that of the optimal solution obtained by CPLEX.

5. Computational study

In this section, some computational results regarding the solution approaches proposed will be given. The test problems are categorized into 7 different classes (each containing 20 test instances) according to the size of the test problems as depicted in Table 5. The test instances in classes 1–4 are of rather small and common problems, where the planning horizon T can be 7 days, 30 days, 12 or 24 hours. Less common problems are represented by the test instances in classes 5–7, in which demand cycle M and resource cycle N are set to be different prime numbers, which causes a significant increase in the problem size. Demand and capacity data in each time period are generated from a uniform distribution $\lambda_i \sim U(0, 500)$ and $c_j \sim U(0, 500)$.

All computations were carried out on a PC with an Intel Core 2 Duo CPU E6550 @ 2.33 GHz 1.98 GHz and 3.25 GB RAM under Windows XP. As a MQIP solver, CPLEX 11.2 is used, which is a commercial solver employing branch-and-cut methods. We applied it to the formulation of the WBP in Section 2.4 without any further specific tailoring, i.e. CPLEX's standard configuration parameters were used, except that the default relative and absolute optimality tolerance of 10^{-4} and 10^{-6} were changed to 0. These changes are necessary, because the objective function value of the WBP is near zero, and hence the default tolerance gaps would significantly influence

the solution quality. CPLEX terminates after either an optimal solution is found or because of an error. The heuristics are coded in C.

Table 5: The configuration of test problems

$M = N = \tau$								
Class	M	N	τ	ℓ	L	No. of [a] variables	No. of constraints	No. of test instances
1	7	7	7	2	5	49 (21)	28	20 (1-20)
2	12	12	12	2	5	144 (36)	48	20 (21-40)
3	24	24	24	2	5	576 (72)	96	20 (41-60)
4	30	30	30	2	5	900 (90)	120	20 (61-80)
$M \neq N$								
5	7	13	91	2	7	8,281 (455)	364	20 (81-100)
6	11	13	143	2	7	20,449 (715)	572	20 (101-120)
7	13	17	221	2	7	48,841 (1,105)	884	20 (121-140)

[a]The column indicates the total number of variables and the number of positive variables given in brackets

Firstly, the heuristic algorithms are used to solve the small size test instances of classes 1–4. The run times for finding feasible solutions are effectively instantaneous. However, in order to verify the solution quality of our algorithms and those of Li and He, namely whether they find an optimum or not, we compare the heuristic solutions with those provided by CPLEX. We set $k_{max} = 1,000$ and $t_{max} = 200$ seconds. The quality of a solution obtained by S-ILS and M-ILS is indicated in Table 6 as a percentage gap between the objective value of this solution and either the optimum solution value (if available) or the objective value of the best solution found by the MQIP solver. For each problem class of 20 different test instances, the minimum, mean and maximum values of the percentage gap to CPLEX's solution are recorded. Further, CPU times are also presented in Table 6. CPLEX was able to solve 75%, 60%, 60%, and 55% of instances of classes 1–4 to optimality respectively, but the computation is for most of the problem instances time consuming. The same problems are solved in much shorter

CPU time by S-ILS and M-ILS. The final solutions provided by the heuristics are, on average, very close to those obtained by CPLEX. The average gap to CPLEX's solutions is between 0.48% and 1.56% for S-ILS and between 0.40% and 0.77% for M-ILS. Furthermore, it can be observed that on average M-ILS converges faster and provides better solution quality than S-ILS. Figure 1 illustrates the comparison results between S-ILS, M-ILS and CPLEX in more details.

Table 6: Gaps to best known solution and solution times

		% Gap		CPU Time		
Class		S-ILS	M-ILS	CPLEX	S-ILS	M-ILS
1	Min	0,00	0,00	0,01	200 s	
	Mean	0,75	0,77	2.173,53		
	Max	8,49	8,49	17.394,23		
2	Min	0,00	0,00	0,01		
	Mean	0,48	0,40	4.753,11		
	Max	2,05	2,01	28.995,38		
3	Min	0,12	0,04	0,28		
	Mean	1,29	0,63	1.676,93		
	Max	4,18	2,96	13.668,77		
4	Min	0,16	0,10	1,22		
	Mean	1,56	0,69	807,83		
	Max	5,43	3,08	2.688,63		

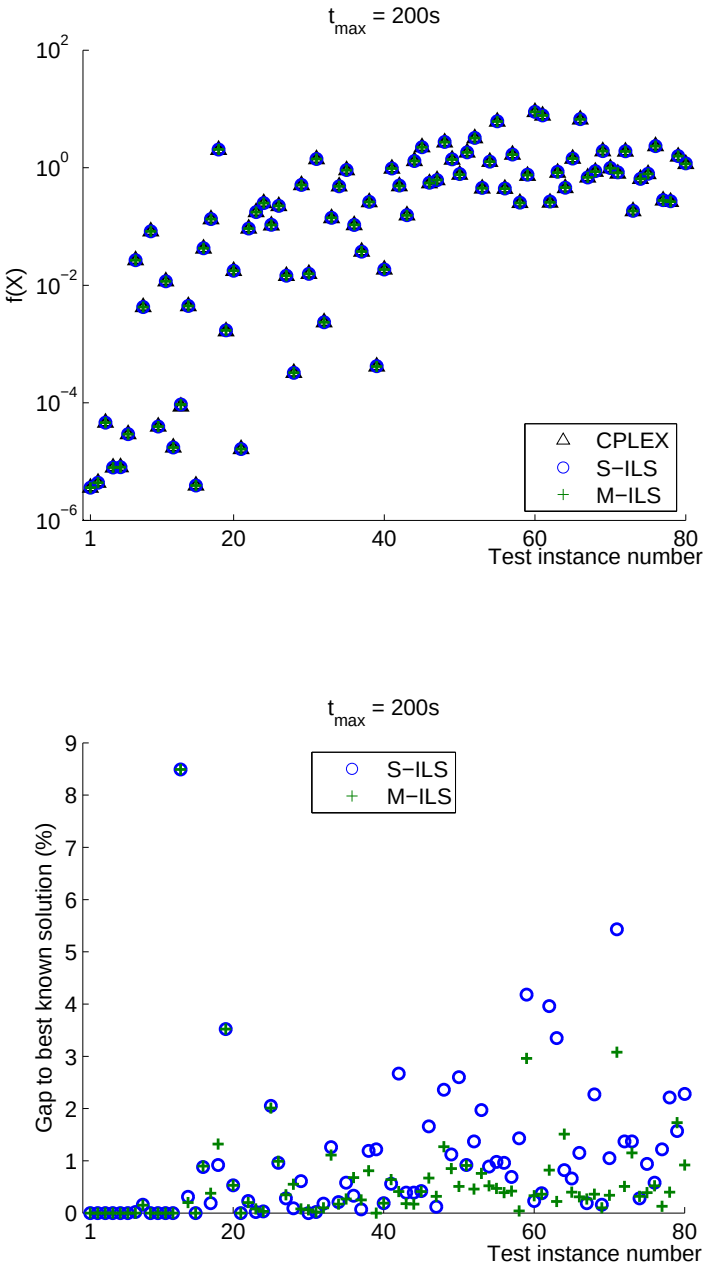

Figure 1: Test problem instances of class 1–4, $t_{max} = 200s$

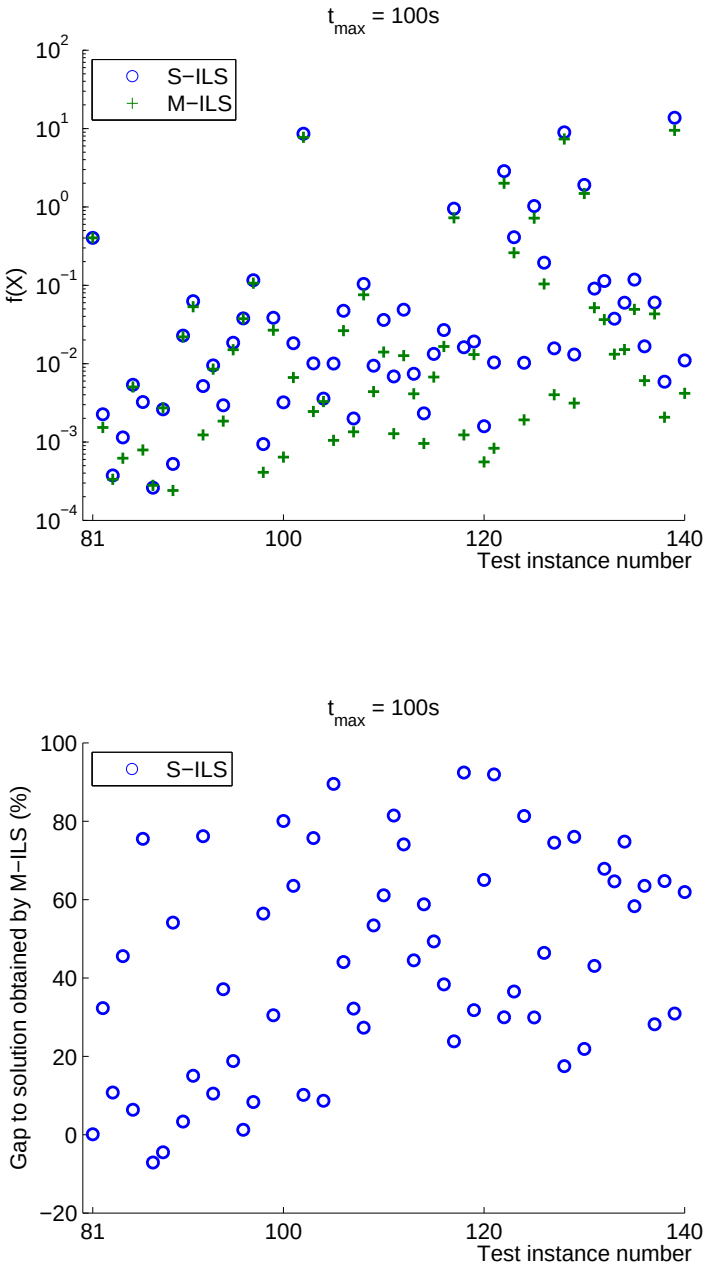

Figure 2: Test problem instances of class 5–7, $t_{max} = 100s$

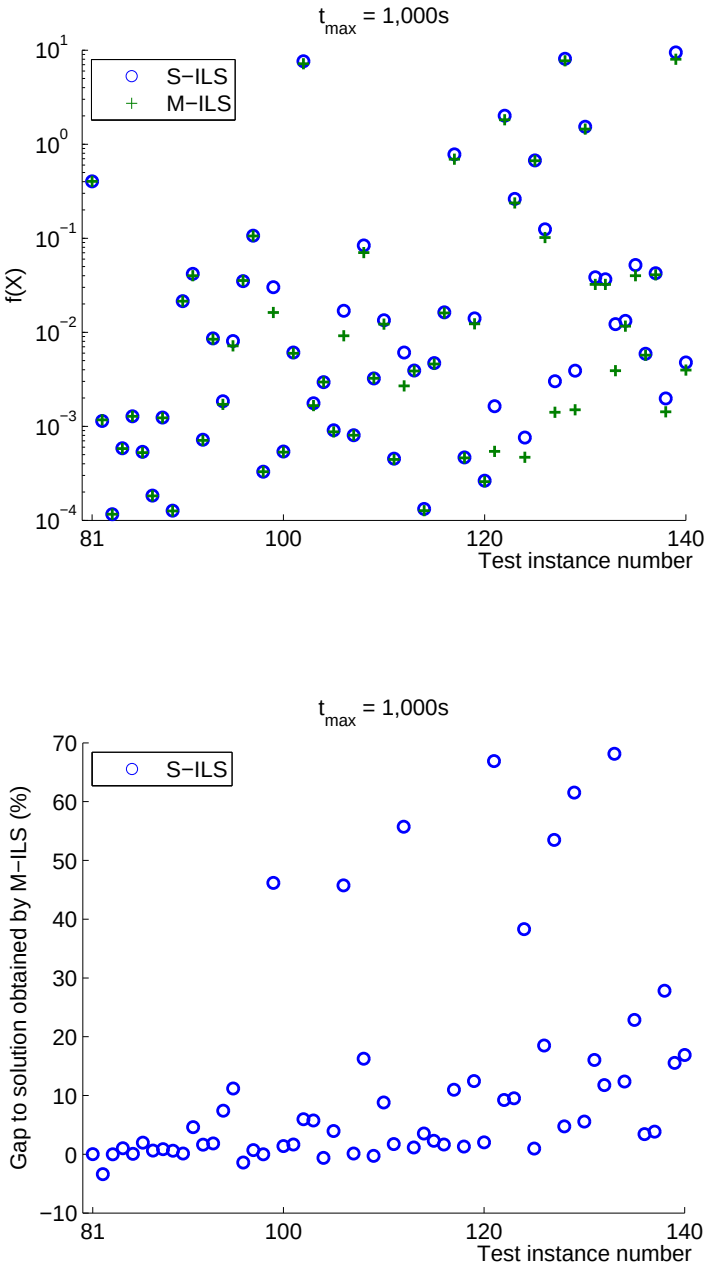

Figure 3: Test problem instances of class 5–7, $t_{max} = 1,000s$

Because of the large problem size, test instances of classes 5–7 cannot be solved by CPLEX. Here, S-VNS and M-VNS are compared on the basis of equivalent CPU times. We run both heuristics with t_{max} equal to 100 sec and 1,000 sec for each instance of classes 5–7. The objective function values of final solutions are depicted in Figure 2-3. The results show that considering the solution quality, in most of the cases M-VNS outperforms S-VNS, but the gaps decrease with increasing t_{max}. Generally, it can be stated that the longer the planning horizon τ is, and the shorter the ratio L/τ is, the more rapidly M-VNS converges than S-VNS.

The experiments also show that our heuristics yield substantial improvements in CPU time compared to those of Li and He. The CPU times for solving three test instances of class 4 by using their heuristics, H1 and H2, lie between 39 and 67 min (as indicated in [26] without any information regarding the solution quality and the used computer), whereas our maximum CPU time is 200 sec. The main reasons for this big gap are as follows. The number of jobs to be shifted in H1 and H2, based on which a decision will be made whether it is better to shift, is not always feasible. This may lead to unnecessary search iterations without solution improvement. Another feature, which makes H1 and H2 less efficient is that jobs are shifted individually and for each job a check has to be made to ensure the lead-time constraints. Therefore, when a large number of jobs are to be analyzed, these approaches are very time consuming. In contrast, S-VNS and M-VNS employ feasible shift operations $O_{ij}(t, \pi_i)$, in each of which a batch of π_i jobs are moved to the target position. Further, while H1 and H2 can easily get stuck in local optima due to their greedy strategy, our stochastic search heuristics explore the neighborhood more thoroughly.

6. Conclusion

This paper proposes several improvements for the previously published work by Li and He [26]. First, a mathematical formulation of the WBP was proposed. The mathematical model helps us to better understand the structure of the problem and justifies the use of heuristics. Second, two variants of a new heuristic, S-ILS and M-ILS, are proposed to improve the solution quality as well as the computational efforts. A further improvement to the previous work is that insights into the quality of solutions obtained by using heuristic algorithms are provided by comparing those with CPLEX's best known solutions for small problems. The empirical results indicate that

the use of CPLEX for solving the WBP is limited due to the complexity of the problem, whereas both heuristic variants are able to provide very good results efficiently. For large scale problems, M-ILS turns out to be more efficient than S-ILS. Using the proposed solution algorithms we have developed a prototype of a software application for jobs scheduling (Figure 4) at the company under investigation. The software application can be implemented as a component in the company's integrated planning system. Such a system is described in Section 2. This component receives data inputs regarding demand and capacity from the modules for demand forecasting and capacity deployment planning and provides a (near-) optimal solution and graphical visualization for the WBP when data for lead-time are entered.

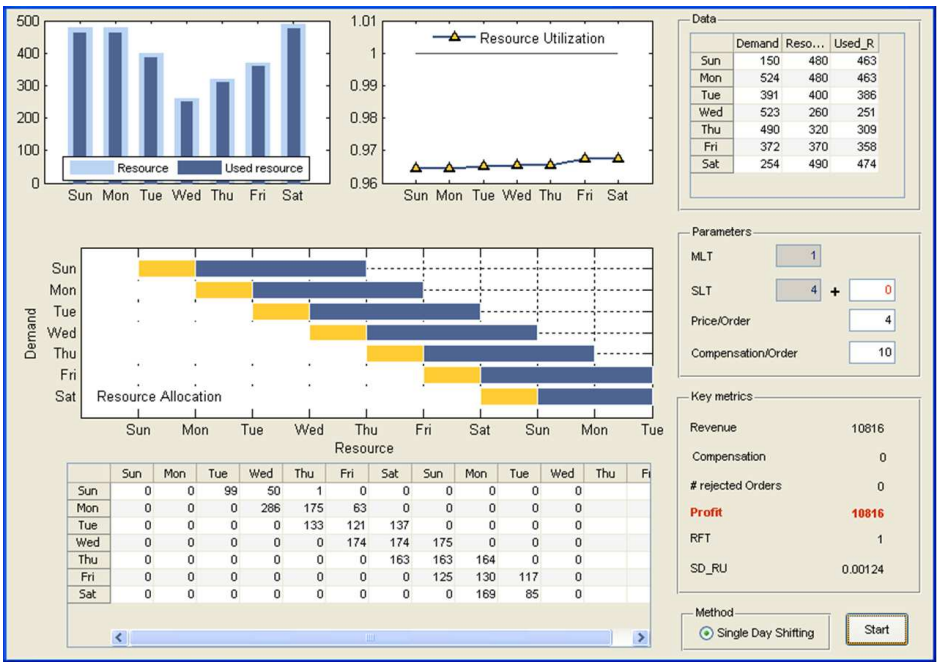

Figure 4: Software application for jobs scheduling

Future research could focus on profit maximization taking into consideration that customers are delay sensitive, i.e. customer demand increases with lower lead-time. Another aspect which is worthy to pay attention on is that the firm may incur lateness penalties when ever the actual delivery time exceeds the quoted lead-time. In addition, demand uncertainty is an interesting topic for further study in this context.

References

[1] T. Stützle, Iterated local search for the quadratic assignment problem, European Journal of Operational Research 174 (2006) 1519–1539.

[2] P. Hansen, N. Mladenovíc, J. Brimberg, J. A. M. Pérez, Variable neighborhood search, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Vol. 146 of International Series in Operations Research & Management Science Volume, Sprin, 2010, pp. 61–86.

[3] P. Hansen, N. Mladenovíc, Variable neighborhood search, in: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer, 2005, pp. 211–238.

[4] N. Mladenovíc, P. Hansen, Variable neighborhood search, Computers Operations Research 24 (11) (1997) 1097–1100.

[5] A. G. Nikolaev, S. H. Jacobson, Simulated annealing, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Vol. 146 of International Series in Operations Research & Management Science, Springer, 2010, pp. 1–39.

[6] D. T. Connolly, An improved annealing scheme for the QAP, European Journal of Operational Research 46 (1990) 93–100.

[7] R. Burkard, F. Rendl, A thermodynamically motivated simulation procedure for combinatorial optimization problems, European Journal of Operational Research 17 (2) (1984) 169–174.

[8] T. James, C. Rego, F. Glover, Multistart tabu search and diversification strategies for the quadratic assignment problem, IEEE Transactions on-Systems, Man and Cybernetics, Part A: Systems and Humans 39 (3) (2009) 579–596.

[9] R. Battiti, G. Tecchiolli, The Reactive Tabu Search, ORSA Journal on Computing 6 (2) (1994) 126–140.

[10] J. Skorin-Kapov, Tabu Search Applied to the Quadratic Assignment Problem, INFORMS Journal on Computing 2 (1) (1990) 33–45.

[11] J. M. S. Valente, M. R. A. Moreira, A. Singh, R. A. F. S. Alves, Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs, The International Journal of Advanced Manufacturing Technology 54 (2011) 251–265.

[12] J. M. Valente, J. F. Goncalves, A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties, Computers & Operations Research 36 (2009) 2707–2715.

[13] D. M. Tate, A. E. Smith, A Genetic Approach to the Quadratic Assignment Problem, Computers & Operations Research 22 (1) (1995) 73–83.

[14] C. Fleurent, J. A. Ferland, Genetic Hybrids for the Quadratic Assignment Problem, in: P. Pardalos, H. Wolkowicz (Eds.), Quadratic assignment and related problems, Vol. 16 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1994, pp. 173–187.

[15] P. C. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem, Journal of the Society for Industrial and Applied Mathematics 10 (2) (1962) 305–313.

[16] A. Auger, N. Hansen, Theory of evolution strategies: a new perspective, in: A. Auger, B. Doerr (Eds.), Theory of Randomized Search Heuristics: Foundations and Recent Developments, Series on Theoretical Computer Science, World Scientific Publishing Company, 2011, pp. 289–325.

[17] V. Nissen, Solving the Quadratic Assignment Problem with Clues from Nature, IEEE Transactions on Neutral Networks 5 (1) (1994) 66–71.

[18] M. López-Ibáñez, T. Stützle, An experimental analysis of design choices of multi-objective ant colony optimization algorithms, Swarm Intelligence 6 (2012) 207–232.

[19] T. Stützle, H. H. Hoos, MAX-MIN Ant System, Future Generation Computer Systems 16 (2000) 889–914.

[20] V. Maniezzo, Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem, INFORMS Journal on Computing 11 (4) (1999) 358–368.

[21] L. M. Gambardella, E. D. Taillard, M. Dorigo, Ant colonies for the quadratic assigment problem, Journal of the Operational Research Society 50 (2) (1999) 167–176.

[22] M. G. Resende, C. C. Ribeiro, F. Glover, R. Martí, Scatter search and path-relinking: Fundamentals, advances, and applications, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Vol. 146 of International Series in Operations Research & Management Science, Springer, 2010, pp. 87–107.

[23] D. S. Johnson, L. A. McGeoch, Experimental analysis of heuristics for the stsp, in: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and Its Variations, Kluwer Academic Publishers, 2002, pp. 369–443.

[24] A. K. M., B. N. W., L. J. Goux J.-P., Solving large quadratic assignment problems on computational grids, Mathematical Programming 91 (3) (2002) 563–588.

[25] E. D. Taillard, Robust taboo search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.

[26] Y. Li, B. He, Optimizing lead time and resource utilization for service enterprises, Service Oriented Computing and Applications 2 (2-3) (2008) 65–78.