9-2021

The multi-skilled multi-period workforce assignment problem

Haibo Wang

Bahram Alidaee

Jaime Ortiz
The University of Texas Rio Grande Valley, jaime.ortiz@utrgv.edu

Wei Wang

Recommended Citation

Wang, H., Alidaee, B., Ortiz, J. and Wang, W., 2021. The multi-skilled multi-period workforce assignment problem. International Journal of Production Research, 59(18), pp.5477-5494. https://doi.org/10.1080/00207543.2020.1783009

The Multi-skilled Multi-period Workforce Assignment Problem

Haibo Wang,[a,1*] Bahram Alidaee[b] , Jaime Ortiz [c], Wei Wang[d*]

[a] A.R. Sánchez Jr. School of Business, Texas A&M International University, Laredo, TX 78041, United States
[b] School of Business Administration, University of Mississippi, University, MS 38677, United States
[c] Global Strategies and Studies, University of Houston, Houston, TX 77204, United States
[d] College of Economics and Management, Chang'An University, Xi'an, Shaanxi, 710064,China

Abstract

Seasonal business operations hire workers within a time window that depends on environmental conditions and market prices. For example, in agricultural business, during the growing and harvest seasons, multiple workers are deployed in the jobs, performing activities such as tilling soil, sowing seed, spreading fertilizer, spraying pesticides, removing weeds, and threshing crops. This study proposes two mixed-integer programming (MIP) models with an effective heuristic to solve the problem of simultaneously assigning multiple multi-skilled workers to the multiple tasks requiring different skill sets during single-period and multiple-period operations. The MIP models are NP hard in the strong sense, and it seems unlikely that large sized realistic instances could be solved efficiently by exact algorithms directly except for some instances with very sparse tasks and skill sets. Thus, this study presents a heuristic algorithm using k-*Opt* as a diversification strategy embedded within the Tabu search for this complex problem. To assess the solution quality of k-*Opt* heuristic, we solved two sets of instances with different sizes by running the exact solver Gurobi and the proposed heuristic algorithm with single processor as well as running Gurobi with multiple processors. This heuristic is applicable to other multitasking situations where a large number of workers with multiple capabilities are deployed.

Keywords: Multi-period; Mixed-Integer Programming; Multitasking; Multi-skilled; K-*Opt* Strategy

* Corresponding authors: A. R. Sánchez Jr. School of Business, Texas A&M International University, Laredo, TX 78041, United States and College of Economics and Management, Chang'An University, Xi'an, Shaanxi, 710064,China. E-mail addresses: hwang@tamiu.edu (H. Wang), balidaee@bus.olemiss.edu (B. Alidaee), jortiz22@uh.edu, (J. Ortiz), wwang@chd.edu.cn (W. Wang).

1. Introduction

Seasonal business operations hire workers within a time window that is restricted by environmental conditions and market prices. For example, in agricultural business, during the growing and harvest seasons of agricultural business, multiple workers are hired and deployed to perform activities such as tilling soil, sowing seed, spreading fertilizer, spraying pesticides, removing weeds, and threshing crops. The multiple-period production cycle in agriculture demands highly complex production planning and effective supply chain risk management due to random yields and market demands (Lowe and Preckel 2004; Allen and Schuster 2004; Van Elderen 1980; Wijngaard 1988; Basnet, Foulds, and Wilson 2006; Edwards et al. 2015; He, Li, and Wang 2018; He and Li 2019).

The decision to hire seasonal workers for the sole purpose of lowering labor costs without investing in training might lead to a higher overall cost if it results in less utilization of labor resource. Consequently, a firm operating with a low utilization of labor skills might not remain competitive. In other type of seasonal business operations such as traveling and sport tourism businesses, cross-training not only provides flexibility to the business and improve customer satisfaction, but also help employee retention and service quality (Salem and Abdien 2017; Mabert and Showalter 1990). The mathematical models for decision-making under uncertain conditions are recently developed for cross-training (Olivella and Nembhard 2016).

The strategy for addressing the challenge under uncertain conditions in seasonal business is to develop multi-skilled workforce using cross-training. Cross-training both improves workers' productivity and increases their welfare (Eden and Gaggl 2018), and it becomes core competitiveness in retail industry. For example, the German Retailer Aldi uses cross-training to achieve lower operational cost by deploying four to five employees per store (Gerhard and Hahn 2005). In recent years, cross-training have attracted interest from academicians and practitioners for their potential to improve productivity as well as the socioeconomic welfare of workers.

In this paper, we advance this line of research on two fronts. We examine the economic feasibility of hiring and cross-training seasonal workers to perform multitasking. We integrate both objectives and propose new mixed-integer programming (MIP) models for multi-skilled workforce management (MSWM) under single-period and multiple-period operations. We also present a heuristic algorithm to solve this complex problem and numerically evaluate the performance. We use Gurobi to solve two sets of instances with different sizes under both sequential and parallel computing environment and compare the results of the proposed k-Opt heuristic.

The remainder of the paper is structured as follows: In section 2, we provide the theoretical and

conceptual background and the state of the art in managing a multi-skilled workforce in a multitasking environment. We introduce our models in section 3. Section 4 presents the results of a substantial simulation exercise, and finally, in section 5, we discuss the managerial implications along with limitations and future development.

2. Literature review

2.1 Cross-training

Many firms found cross-training effective approaches to create a multi-skilled workforce to meet the demand with rising labor costs and aging workforce (Gerhard and Hahn 2005; Luce 2013; Gnanlet and Gilland 2014; Malhotra and Ritzman 1994). It is operational-intensive to manage the workers, and plan cross-training the multi-skilled workforce in these firms. In the meantime, others proposed conceptual models for empirical studies and investigated the managerial issues of multi-skilled workers under resource-constrained environment. The models and methods on cross-training and management of multi-skilled workforce of multitasking assignment are summarized in Table 1.

Table 1

Models and methods on cross-training and multitasking assignment

Reference	Application	Model	Solution Method/Solver
Ebeling and Lee (1994)	cross-training	MILP	GAMS solver
Campbell (1999)	cross-training	MILP	Dynamic Programming
Shakeri and Logendran (2007)	Multitasking assignment	MILP	Tabu search
Kim and Nembhard (2013)	cross-training	MILP	CPLEX solver
Lusa, Corominas, and Pastor (2008)	cross-training	MINLP	MINLP solver without specific detail
Olivella, Corominas, and Pastor (2013)	cross-training	MILP	CPLEX solver
Telhada (2014)	cross-training	MILP	CPLEX solver
Hall, Leung, and Li (2015)	Multitasking assignment	MILP	Dynamic programming algorithms
Hall, Leung, and Li (2016)	Multitasking assignment	MILP	Dynamic programming algorithms

Cheng et al. (2017)	Multitasking assignment	Nonlinear programming with continuous variables	Particle Swarm algorithm
Liu et al. (2017)	Multitasking assignment	MILP	Dynamic programming algorithms
Zhu, Zheng, and Chu (2017)	Multitasking assignment	MILP	Dynamic programming algorithms(Hall, Leung, and Li 2015)
Ji et al. (2019)	Multitasking assignment	MILP	Dynamic programming algorithms

The development of seasonal business operation requires a multi-skilled workforce able to perform in a multitasking environment. Seasonal business operation often faces these challenges in dynamic multitasking environments and depends on human operators with multiple skills. Hiring seasonal workers and training them in such skills requires extensive resources. In addition, uncertain environments create more complexity in scheduling seasonal workers. It is critical to make employment decisions months before the season starts.

2.2 *K-Opt* heuristic

In this study, we choose k-*Opt* strategies for solving improvement processes. k-*Opt* is the most widely cited local optimization method originally used to solve sequential problems such as TSP (Croes 1958; Lin 1965; Lin and Kernighan 1973), (Knox 1994; Potvin 1996; Helsgaun 2009; Blazinskas and Misevicius 2011; Mladenović et al. 2012; Taillard and Helsgaun 2019). k-*Opt* is also integrated with other heuristic methods such as Tabu Search, Genetic Algorithm, GRASP, Simulated Annealing, Ant Colony and Particle Swarm, for solving applications in network optimization problems such as Fixed Charge Transportation Problem (Sun et al. 1998; Rodríguez-Martín and Salazar-González 2010), Facility Location and P-Median problem (Hansen and Mladenović 2001), Cluster Analysis(Hansen and Mladenović 2001), Bilinear QP with bilinear constraints (Hansen and Mladenović 2001), Maritime Inventory Routing (Papageorgiou et al. 2018), Maximum Clique (Katayama, Hamamoto, and Narihisa 2005), Machine Scheduling (Sadfi et al. 2005; Wang and Alidaee 2018, 2019; He, Zhong, and Gu 2006), and Mobile Network Design (Fournier and Pierre 2005). The list of recent development and applications of k-*Opt* is given in Appendix A. We can observed that k-*Opt* is not limited to TSP and can be applied to some new applications such as MSWM in this study.

Given the challenges of planning multiple dynamic tasks, and deploying multi-skilled employees, we present in the next section an integrated mathematical model as a holistic approach to the multi-skilled

multi-period work force scheduling problem.

3. Model and solution methodology

Seasonal businesses such as agriculture hire workers within a time window that is restricted by weather conditions and market prices. During the seasons, multi-skilled workers are assigned to a set of jobs according to their skills. In this study, we assume that there are enough workers with the required skills to perform the jobs. Managers often assign a worker to the same job sets throughout the production period and each job has one or more tasks. Hence, we investigate the cost challenge of scheduling workers with multi-skills for multitasking environments. First, we present a MIP model for seasonal planning during a single-period production cycle. In this study, we focus on the cost problems related to labor (hiring and cross-training) and cost to perform tasks in both single-period and multiple-period production cycles. The efficient use of worker's skills reduces the cost of labor, thus, the improved scheduling of workers provides seasonal businesses with a better planning horizon.

In subsection 3.1, we present a MIP model to solve the MSWM problem in a single-period production cycle. Then, in subsection 3.2., we introduce the necessary modifications to address the cost problem of multiple-period production for MSWM in a planning horizon. We propose an effective algorithm in subsection 3.3 to solve the MIP models in both scenarios. We begin by defining the input parameters and decision variables in the model with their respective notation for single-period and multiple-period production cycles.

Input parameters for single-period production :

N_j	Set of specific tasks needed to be performed in a job $j = 1 \ldots n$, assuming there are enough qualified workers available in each period of planning horizon
K	Set of all skill-types needed to do all task
M_k	Set of workers possessing skill for task $k \in N_j$
n	Number of jobs in each period of the planning horizon
m	Number of qualified workers available to do the tasks in the jobs
Q_i	Total time available for qualified workers $i = 1 \ldots m$ in a day, 4 hours for a part-time worker, 8 hours for a full-time worker (values randomly chosen between 4 and 8 hours)
d_{jk}	Time required (demand) to do task $k \in N_j$ in a job $j = 1 \ldots n$ (values randomly chosen between 1 and 8 hours)
a_i	Fixed cost (e.g., initial hiring and cross-training of a worker) of deploying workers $i = 1 \ldots m$ (values chosen between \$150 and \$1,000 based on worker's skill set, fixed cost decreases if worker possessing more skills)

c_{jki} Cost of deploying a qualified worker $i = 1 \ldots m$, working on task k $\in N_j$ in a job $j = 1 \ldots n$ (hour rate value chosen between \$7 and \$20 per hour based on U.S. Department of Labor pay scales and the worker's skill set, then multiple by the time required to do task, worker with more skills commands higher wage)

Decision variables for single-period production:

x_{jki} Equal to 1 if task k $\in N_j$ on job $j = 1 \ldots n$ is done by a qualified worker $i = 1 \ldots m$, or 0 otherwise

y_i Equal to 1 if deploying a qualified worker $i = 1 \ldots m$ or 0 otherwise

3.1 Single-period production model

The MIP model for single-period production is formulated as follows:

$$Minimize \quad \sum_{i=1}^{m} a_i y_i + \sum_{j=1}^{n} \sum_{k \in N_j} \sum_{i=1}^{m} c_{jki} x_{jki}$$

s.t.

$$\sum_{i=1}^{m} x_{jki} = 1, \qquad \forall j, k \in N_j \tag{1}$$

$$d_{jk} x_{jki} \leq Q_i y_i \qquad \forall j, k \in N_j, i \tag{2}$$

$$y_i, x_{jki} \in \{0,1\} \qquad \forall j, k \in N_j, i \tag{3}$$

Constraints (2) can be substituted with the following two equations (Laporte, Nickel, and da Gama 2015)

$$d_{jk} x_{jki} \leq Q_i, \qquad \forall j, k \in N_j, i \tag{2.1}$$

$$x_{jki} \leq y_i, \qquad \forall j, k \in N_j, i \tag{2.2}$$

In the single-period production model, Constraints (1) guarantee that every task *k* of job *j* must be done by a qualified worker. Constraints (2) ensure that if a worker *i* works on several tasks, the time of each task must be within the availability of the qualified worker. Constraints (3) impose a binary restriction on the decision variables.

Constraints (2) are capacitated, thus the single-period production problem is NP hard.

3.2 Multiple-period production model

Input parameters for multiple-period production :

N_j Set of specific tasks needed to be performed in a job $j = 1 \ldots n$, assuming there are qualified workers available in each period of planning horizon

K Set of all skill-types needed to do all task

M_k Set of workers possessing skill for task $k \in N_j$

n Number of jobs in each period of the planning horizon

m Number of qualified workers available to do the tasks in the jobs

Q_i Total time available for qualified workers $i = 1 \ldots m$

T Number of periods in planning horizon (e.g., 10 weeks or 8 hours per day)

d_{jkt} Time required (demand) to do task k $\in N_j$ in a job $j = 1 \ldots n$ during a period $t = 1 \ldots T$ (values randomly chosen between 1 and 8 hours)

q_{it} Total time a qualified worker is available $i = 1 \ldots m$, during a period $t = 1 \ldots T$

a_{it} Fixed cost (e.g., initial hiring and training of a qualified worker $i = 1 \ldots m$), during a period $t = 1 \ldots T$ (values chosen between \$150 and \$1,000 based on worker's skill set and the specific period t, fixed cost decreases if worker possessing more skills in general)

c_{jkit} Cost of a qualified worker $i = 1 \ldots m$ working on task k $\in N_j$ in a job j $1 \ldots n$ during a period $t = 1 \ldots T$ (hour rate value chosen between \$7 and \$20 per hour based on U.S. Department of Labor pay scales and the worker's skill set, then multiple by the time required to do task, worker with more skills commands higher wage in general with seasonal factor of the specific period t)

Decision variables for multiple-period production:

x_{jkit} Equal to 1 if task k $\in N_j$ on job $j = 1 \ldots n$ is done by a qualified worker $i = 1 \ldots m$ during a period $t = 1 \ldots T$, and 0 otherwise

y_{it} Equal to 1 if a qualified worker $i = 1 \ldots m$ starts first time during a period $t = 1 \ldots T$, and 0 otherwise

v_{it} Equal to 1 if a qualified worker $i = 1 \ldots m$ works in a period $t = 1 \ldots T$, and 0 otherwise

The MIP model for multiple-period production is formulated as follows:

$$Minimize \quad \sum_{i=1}^{m} \sum_{t=1}^{T} a_{it} y_{it} + \sum_{j=1}^{n} \sum_{k \in N_j} \sum_{t=1}^{T} \sum_{i=1}^{m} c_{jkit} x_{jkit}$$

s.t.

$$\sum_{i=1}^{m} x_{jkit} = 1 \qquad \forall j, \mathrm{k} \in N_j, t \tag{4}$$

$$d_{jkt} x_{jkit} \leq q_{it} v_{it} \qquad \forall j, \mathrm{k} \in N_j, i, t \tag{5}$$

$$\sum_{t=1}^{T} q_{it} v_{it} \leq Q_i \qquad \forall i \tag{6}$$

$$v_{it} \leq \sum_{l=1}^{t} y_{il} \qquad \forall i, t \tag{7}$$

$$y_{it}, v_{it}, x_{jkit} \in \{0,1\} \qquad \forall j, \mathrm{k} \in N_j, i, t \tag{8}$$

In this model, Constraints (4) guarantee that every task *k* of job *j* must be done by a qualified worker during a period *t* of the horizon. Constraints (5) ensure that if a worker *i* works on several tasks during a period *t*; then the time of each task must be within the availability of the qualified worker *i* during that period. Thus, for the worker *i* and period *t*, we have $\max_{jk}\{d_{jkt} x_{jkit}\} \leq q_{it} v_{it}$. Note that Constraints (5) also ensure that the assignment of a qualified worker *i* for a task during period *t* is only possible if the qualified

worker *i* is deployed in that period. Constraints (6) ensure that the total working time of the worker *i* during the planning horizon is limited by its availability Q_i. Constraints (7) ensure that if worker *I* works in period *t*, then he/she must have been trained earlier or during time *t* and enforce the first time a worker starts to work will have some fixed cost. The fixed cost is related to worker's skill set. Constraints (8) impose a binary restriction on decision variables.

Constraints (5) and constraints (6) are both capacitated. MSWM problem is the special case of an uncapacitated facility location (UFL) problem, which is a NP hard in the strong sense (Galvão and Raggi 1989; Garey and Johnson 1990). The comparison between MSWM and UFL problem is given in Appendix B. Therefore, it seems unlikely that large sized realistic instances could be solved efficiently by exact algorithms directly except for some instances with very sparse tasks and skill sets. Thus, we provide a new algorithm to solve the multiple-period production problems as well as the single-period production problems where *t* is equal to 1.

For MSWM problem in multi-period production, we consider an instance with 3 jobs and 3 workers for 2 periods. The tasks of jobs and time of each task as well as the workers' availability are given. The fixed cost for workers who start at any given day are determined by the workers' skill set. The input data and results are given in Table 2 and the values of the decision variables are given in the Supplementary Material ST.14. In the optimal solution, only worker M_1 is deployed to complete three jobs in both periods. The associated variable costs c_{jkit} and fixed cost a_{it} are highlighted in boldface. Before worker M_1 is deployed in period t_1, the fixed cost a_{1t} has to be occurred for hiring and training of worker M_1. The variable costs for worker M_1 to complete the task of Job 1, 2, and 3 in period t_1 are 15, 30, 17, and 15. The variable costs for worker M_1 to complete the task of Job 1, 2, and 3 in period t_2 are 32, 17, 16, and 34. The objective function value is the sum of all variable costs and fixed cost of worker M_1.

Table 2

Illustrated example for MSWM problem in multi-period production

Job	N_1	N_2	N_3	N_4	Worker	M_1	M_2	M_3	M_4
1	1		1		1	1	1	1	1
2			1	1	2		1	1	1
3	1	1			3	1		1	

d_{jkt}	d_{11t}	d_{13t}	d_{23t}	d_{24t}	d_{31t}	d_{32t}	q_{1t}	q_{2t}	q_{3t}
t=1	1	2	1	1			8	8	4
t=2	2	1			1	2	8	4	4

c_{jkit}	c_{111t}	c_{112t}	c_{113t}	c_{131t}	c_{132t}	c_{133t}	c_{231t}	c_{232t}	c_{233t}
t=1	**15**	10	7	**30**	20	14	**17**	13	8

t=2	**32**	22	15	**17**	13	9			
c_{jkit}	c_{241t}	c_{242t}	c_{243t}	c_{311t}	c_{312t}	c_{313t}	c_{321t}	c_{322t}	c_{323t}
t=1	**15**	12	8						
t=2				**16**	14	9	**34**	22	15

a_{it}	a_{1t}	a_{2t}	a_{3t}	Q_1	16
t=1	**125**	240	320	Q_2	12
t=2	135	260	340	Q_3	10

Solution			
Objective Function Value			301

3.3 K-*Opt strategies*

In seasonal businesses, if a set of specific tasks N_j must be performed on job *j*, then for a given task *k* in N_j, each element of N_j needs a different worker to complete it during period *t*. If *t* is equal to 1, we are then addressing single-period production. To determine the assignment of a qualified worker *X(j) = I* who completes this task *k* during period *t*, we need to check all workers $l \neq k$. Thus, we need to have a strategy for determining the order in which to check all workers to produce the greatest efficiency. Following Recent published procedures (Wang and Alidaee 2018, 2019), we borrow the k-*Opt* strategy from sequencing problems such as the traveling salesman problem (TSP) and use it to improve the assignment. In the process, we implement sequencing strategies *2, 3, 4-Opt,* and *ALL* for the assignment of both tasks and workers (see Figures SF1–SF3 and detailed description in Supplementary Material). There are 5 movements for *2-Opt*, 24 movements for 3-*Opt,* and 8 movements for 4-*Opt*. Based on the k-*Opt* strategies we can choose one of those movements to improve the sequence of tasks and the assignment of workers. Therefore, we have a sequential order of workers in the assignment. We apply one k-*Opt* strategy at each step to create a new sequential order in order to check which qualified worker should be evaluated first. The improvement steps based on the k-*Opt* strategies are implemented by the TS algorithm with an embedded strategic oscillation (SO). Glover and Kochenberger (1996) first introduced this method of using the TS algorithm with an embedded SO and later implemented it in the xQx model (Glover, Kochenberger, and Alidaee 1998). The k-*Opt* strategy in this study is implemented as an exact constraint-guided search method where we fix the same set of variables in each iteration of the improvement process. Definitions for the full pseudo code of our improvement procedures in Fig. 1 are as follows:

v	Number of variables
x	A binary starting solution with *v* variables
*x**	The best solution found so far by an algorithm
Z	The value of the objective function for variable *x*

Z^*	The value of the objective function for the best solution found
π	The vector for the order of tasks and workers
$J_{next}(X)$	A set of variables *X* where conditions of local optimality are not satisfied
L1, L2	The sequence of tasks, the sequence of workers
p1, p2	Two integer constants where $p1 < p2 \leq v$; their values depend on the problem
Tabu_ten	An integer constant for the number of iterations for subsequent searches in which all moves are permissible (maximum value a variable can remain in Tabu status)
Tabu(i)	For *i = 1… v,* a vector representing Tabu status of tasks

1 **Initialization:**
X= a starting feasible solution, $X^*=X$, $Z^x = \bar{\boldsymbol{Z}}$ *p1, p2, Tabu_ten,*
Tabu(i)=0, for *i=1,…,v,* and $\pi = (\pi_1, \cdots, \pi_{v)}$ calculate the set $J_{next}(X)$
2 **Do while** (until some stopping criteria, e.g., time limit, is reached)
3 **For** *K=p1* to *p2*
4 **Do while** $(J_{next}(X) \neq \emptyset)$
5 **For** *i=1 to v-1*
6 $L1=\pi_1$
7 **For** *j=i+1 to v*
8 $L2=\pi_j$
9 **If** $(L1, L2 \epsilon J_{next})$ **Then**
10 $\bar{\boldsymbol{Z}} = \boldsymbol{Z}$ **for** $\bar{\boldsymbol{x}}_{\boldsymbol{j}} = \boldsymbol{x}_{\boldsymbol{j}}, \boldsymbol{j} \neq \boldsymbol{L1}, \boldsymbol{L2}$, **and** $\bar{\boldsymbol{x}}_{\boldsymbol{j,1}} = \mathbf{1} - \boldsymbol{x}_{\boldsymbol{j,1}}, \bar{\boldsymbol{x}}_{\boldsymbol{j,2}} = \mathbf{1} - \boldsymbol{x}_{\boldsymbol{j,2}}$
11 **If** *((Tabu(L1)=0).and. Tabu(L2)=0) or* $(\bar{Z} > Z^*)$ **Then**
12 $\boldsymbol{x}_{\boldsymbol{i,1}} = \mathbf{1} - \boldsymbol{x}_{\boldsymbol{j,1}}, \boldsymbol{x}_{\boldsymbol{i,2}} = \mathbf{1} - \boldsymbol{x}_{\boldsymbol{j,2}}$
13 *Update:* $J_{next}(X)$ *Tabu (j), for j=1,…,v,*
14 $Z = \bar{Z}$,
15 *If* $(\bar{Z} < Z^*)$ $Z^* = \bar{Z}$
16 **End If**
17 **End If**
18 **End For**
19 **End For**
20 Call *k-Opt()*
21 **End while**
22 Call *rand_var_change(.)*
23 **End For**
24 **End while**

Fig. 1. Pseudo code of TS algorithm using k-*Opt* strategies

Our algorithm randomly chooses a set of tasks N_j on a job *j* and randomly changes the worker assignment for each task (Line 1 in algorithm). When the stopping criteria is reached such as time limit, we reported the final solution (Lines 2 to 24). After the local optimality condition is reached, the function "*rand_var_change*" is invoked to change the worker assignments. We use the following rule for those changes. When a local optimality is reached after k-*Opt* improvement processes (Lines 4 to 21), a random number *p* is generated in the interval $p \in [1, K]$ where *K* is changing in the interval *[p1, p2]* then *p* tasks

are randomly selected and randomly make *rand_var_change* on a different worker (Lines 3 to 23). The concept is closely related to the strategic oscillation used in many TS settings; see (Glover and Kochenberger 1996; Glover, Kochenberger, and Alidaee 1998). However, we use some randomness in our algorithm to implement a form of strategic oscillation. Most importantly, we never remove a feasible solution from the Tabu list. We only make a *rand_var_change* on a set of randomly chosen tasks; thus we always preserve any feasible solutions during the improvement process. During the implementation, we only examine the Tabu status on the tasks, not on the workers. Furthermore, we use a simple aspiration criterion in the TS algorithm. If the new solution is strictly better than the best known solution found so far, we override the Tabu status for the variables to be changed.

4. Numerical study on a synthesized dataset

In order to measure the performance of the proposed models and the algorithm, we first created an illustrative example using U.S. Department of Labor pay scale data (United States Department of Labor 2017) and salary survey to calculate labor costs (Edwards and Johanns 2012). Then we evaluated the models and the algorithm with different parameter settings for mismatched supply of and demand for workers' skills. We coded all *k-Opt* strategies in the proposed algorithm using Fortran language, which was compiled by GNU gfortran compiler v4.7 and ran on a single core of Intel Xeon Quad-core E5420 Harpertown processors, which have a 2.5 GHz CPU with 8 GB memory. To assess the solution quality of *k-Opt* heuristic, we solved two sets of instances from 20 jobs and 40 workers up to 1000 jobs and 1500 workers by Gurobi solver version 8.1 (Gurobi Optimization 2019) as well as *k-Opt* Heuristic. We also explored the solver on parallel computing environments by setting the number of threads larger than one. All computing jobs were submitted through the Open PBS Job Management System.

4.1 Parameter Tuning of Heuristic

For seasonal businesses, manager hires both seasonal and permanent workers to perform jobs during the seasons (Wishon et al. 2015). Each job has multiple tasks and one or more workers can work together on the same job but different tasks. On this basis, we first chose 200 jobs and 400 workers to evaluate our models and the algorithm. We chose a different probability (demand side) for a specific task needed on a job randomly. We chose a different probability (supply side) for a qualified worker with a specific skill. This probability can also be used if we consider the ratio of seasonal workers to permanent workers. A high value indicates that the majority of workers are permanent, whereas a low value indicates more workers are seasonal. The probability values were 0.5 and 0.8, respectively. If the time a worker required to perform the tasks in a job was less than the time it was available during the day, we calculated the labor cost as the product of time and cost per hour of deploying that worker for that specific job. As far as the criteria for

stopping the algorithm, our initial implementation tested different values of *p1* and *p2*. These values affected the amount of time required to run the algorithm and the quality of the solution. The values of Tabu_ten also affected the solution quality and we found Tabu_ten = 5 generated a significantly better solution than did values of 3, 10, and 15. Thus, we chose Tabu_ten = 5, *p1* = 1, *p2* = alpha * m + 1 where alpha = 0.05, 0.12, and 0.20 in our initial experiment. For each problem, we ran the algorithm 10, 20, and 30 times. We chose three scenarios to generate the test problems with mismatches of demand for and supply of skill: the skill of the qualified worker is less than, is equal to, or is greater than the skill required to complete the task in the job. These scenarios also reflect on the decision to hire different ratios of seasonal and permanent workers. Results for 200 jobs and 400 workers are presented in Tables 3 - 5.

We adopted the following notation:

NW	Number of times the whole algorithm was run
Z*	The value of the objective function for the best solution found
TB[s]	Time to reach the best solution in seconds
TT[s]	Total time to run the whole algorithm in seconds
prob_job	Probability that a task would require a specific skill (demand side) in a job
prob_skill	Probability that a worker has a specific skill (supply side)
alpha	Number of inner loops in the algorithm; use percentage of the number of workers

Table 3

Comparison of k-*Opt* strategies with different alpha values and n=200, m=400, N_j=6, NW=10

prob_job=0.8		**alpha=0.05**			**alpha=0.12**			**alpha=0.2**	
prob_skill=0.5	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	37439	3.72	16.32	37636	12.16	47.42	37595	38.09	84.16
3-Opt	37584	0.19	16.26	37574	0.25	48.23	37554	23.44	85.49
4-Opt	37619	0.13	16.17	37722	0.27	48.71	37563	1.75	82.30
ALL	37522	5.70	15.82	37413	0.55	49.97	37535	76.86	85.35
Average	37541	2.438	16.14	37586	3.306	48.58	37562	35.03	84.33
prob_job=0.8		**alpha=0.05**			**alpha=0.12**			**alpha=0.2**	
prob_skill=0.8	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	35738	0.26	15.50	35738	0.28	43.01	35624	67.63	76.17
3-Opt	35756	14.44	15.84	35694	23.98	43.34	35684	50.36	75.76
4-Opt	35640	1.00	15.07	35722	31.12	41.38	35721	26.36	76.56
ALL	35557	3.76	15.58	35586	0.19	42.49	35688	33.90	76.54
Average	35673	4.862	15.50	35685	13.89	42.56	35679	44.561	76.26
prob_job=0.5		**alpha=0.05**			**alpha=0.12**			**alpha=0.2**	

prob_skill=0.8	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	24418	4.42	9.87	24559	15.69	26.05	24471	25.88	46.25
3-Opt	24450	0.22	9.31	24505	0.09	26.44	24459	0.15	46.58
4-Opt	24577	0.20	9.17	24531	6.08	25.66	24500	0.43	43.72
ALL	24561	3.60	9.09	24508	0.20	25.04	24435	20.03	45.97
Average	24502	2.11	9.361	24526	5.52	25.80	24466	11.62	45.63

Table 4

Comparison of k-*Opt* strategies with different alpha values and n=200, m=400, N_j=6, NW=20

prob_job=0.8	**alpha=0.05**			**alpha=0.12**			**alpha=0.2**		
prob_skill=0.5	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	37439	3.92	36.84	37510	55.85	117.72	37595	38.19	182.08
3-Opt	37446	2.98	36.21	37539	96.45	116.24	37466	132.03	178.84
4-Opt	37521	4.08	35.79	37419	63.14	98.27	37628	2.30	170.70
ALL	37555	2.57	35.98	37531	81.74	113.60	37481	0.45	180.78
Average	37490	3.39	36.21	37500	74.29	111.46	37543	43.24	178.10
prob_job=0.8	**alpha=0.05**			**alpha=0.12**			**alpha=0.2**		
prob_skill=0.8	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	35738	0.25	32.76	35696	82.76	92.36	35562	83.64	159.86
3-Opt	35628	3.21	32.58	35740	1.11	92.05	35633	10.44	153.91
4-Opt	35562	0.44	32.03	35717	80.00	89.97	35634	8.18	155.80
ALL	35708	8.97	31.50	35771	11.02	93.01	35564	6.84	157.44
Average	35659	3.22	32.22	35731	43.72	91.85	35598	27.27	156.75
prob_job=0.5	**alpha=0.05**			**alpha=0.12**			**alpha=0.2**		
prob_skill=0.8	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	24418	4.31	20.49	24462	28.77	60.17	24471	26.18	98.12
3-Opt	24457	2.18	19.32	24423	30.27	56.41	24441	65.11	96.71
4-Opt	24418	16.67	18.71	24402	2.76	50.82	24493	4.13	92.95
ALL	24485	13.72	19.43	24474	22.46	52.84	24437	5.78	97.02
Average	24445	9.23	19.49	24440	21.07	55.06	24461	25.3	96.2

Table 5

Comparison of k-*Opt* strategies with different alpha values and n=200, m=400, N_j=6, NW=30

prob_job=0.8	**alpha=0.05**			**alpha=0.12**			**alpha=0.2**		
prob_skill=0.5	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	37439	3.64	53.96	37510	52.96	154.26	37595	38.98	339.68

3-Opt	37525	2.25	56.86	37498	42.89	155.07	37553	267.82	294.54
4-Opt	37480	5.66	52.47	37541	26.30	148.68	37586	100.22	265.29
ALL	37480	39.61	52.29	37498	77.48	153.61	37506	224.88	325.08
Average	37481	12.79	53.90	37512	49.91	152.90	37560	157.98	306.15
prob_job=0.8		**alpha=0.05**			**alpha=0.12**			**alpha=0.2**	
prob_skill=0.8	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	35710	33.77	50.14	35674	115.80	140.52	35562	83.41	248.14
3-Opt	35752	16.82	49.86	35496	76.06	135.46	35550	217.61	249.52
4-Opt	35552	48.42	48.81	35678	107.16	132.00	35661	130.55	238.59
ALL	35649	45.72	49.66	35611	76.09	134.20	35628	0.38	243.84
Average	35666	36.18	49.62	35615	93.78	135.54	35600	107.99	245.02
prob_job=0.5		**alpha=0.05**			**alpha=0.12**			**alpha=0.2**	
prob_skill=0.8	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**	**Z***	**TB[s]**	**TT[s]**
2-Opt	24418	4.32	30.23	24462	26.48	83.58	24471	25.92	148.33
3-Opt	24409	24.68	28.42	24439	57.51	82.77	24447	93.85	142.60
4-Opt	24417	30.24	30.50	24443	54.75	88.70	24469	9.77	140.79
ALL	24411	25.32	29.24	24453	80.67	88.73	24384	59.47	143.78
Average	24414	21.14	29.60	24449	54.85	85.94	24443	47.25	143.88

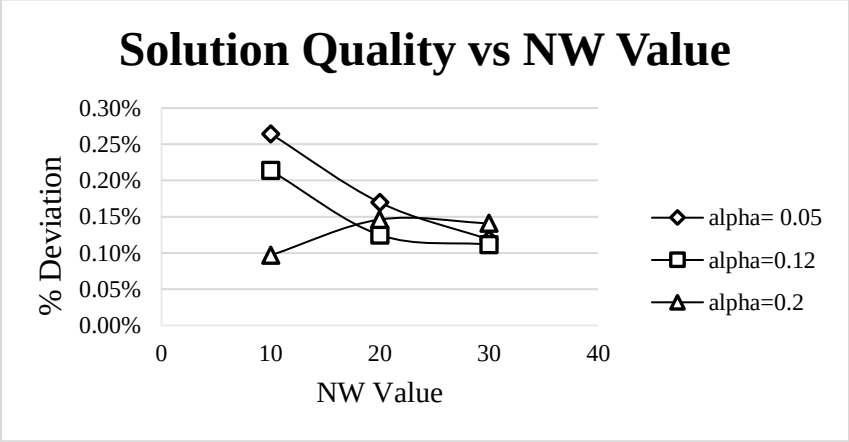


Fig. 2. Solution Quality with different alpha values (n=200, m=400)

In Tables 3–5, each pair of prob_job and prob_skill indicates a scenario of mismatched supply of and demand for skills in the test environment. There are three scenarios for each stopping criterion (NW = 10, 20, and 30) with a total of 9 sets of results. We reported the objective function value (OFV), CPU time to reach the best solution, and total CPU time within the number of times the whole algorithm ran for each scenario under different alpha values and different k-*Opt* strategies, as well as the average OFV of four

strategies. We highlighted the best OFV found and the best average OFV of four strategies. Among the 9 sets of results, our algorithm reported 4/9 of the best OFV and 5/9 of the best average OFV found when alpha was equal to 0.05; 4/9 of best OFV and 1/9 of the best average OFV found when alpha was equal to 0.12; and 1/9 of best OFV and 1/3 of the best average OFV found when alpha was equal to 0.2. Thus, the algorithm performed much better when alpha was equal to 0.05. When alpha is equal to 0.05, we found the best improvement of solution quality while increasing the number of times the whole algorithm ran (Fig. 3). %Deviation in Fig 2 is measured by:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{n}} \Big/ \bar{x}$$

We also analyzed the effect of the number of times required to run the whole algorithm on the solution quality (see Table AT.1 in Appendix). When we increased this number, we found a better OFV or better average OFV on all four strategies. When the skill of a qualified worker increases, the TT (total CPU time within the number of times the whole algorithm ran) value increased while the best OFV improved in correlation with increasing NW values (See Table AT.2 in Appendix). The possible correlation can be explained as follows: if more choices for assigning workers to tasks are available, then it takes more time to construct the initial solutions each time the whole algorithm runs. When the skill of a qualified worker increases, the TB (CPU time to reach the best solution) and the TT (total CPU time within the number of times the whole algorithm ran) values increase except that NW equals 30 ad the average OFV of four strategies improves along with increasing NW values (See Table AT.3 in Appendix).

We performed more tests on parameter settings with alpha = 0.05 and NW = 30 on 200 jobs and 400 workers to see how the problem of mismatches between supply of and demand for skills affects total cost. The probability value of the task demand and worker supply contains six combinations (0.3, 0.5), (0.5, 0.3), (0.5, 0.6), (0.6, 0.5), (0.5, 0.9), and (0.9, 0.5). Table AT.4 in Appendix shows that total cost increases when task demands exceed the supply of skills. Conversely, total cost decreases when the supply of skills exceeds the task demands (Figs. 3 and 4). We found the same effect on the average TB and TT values.

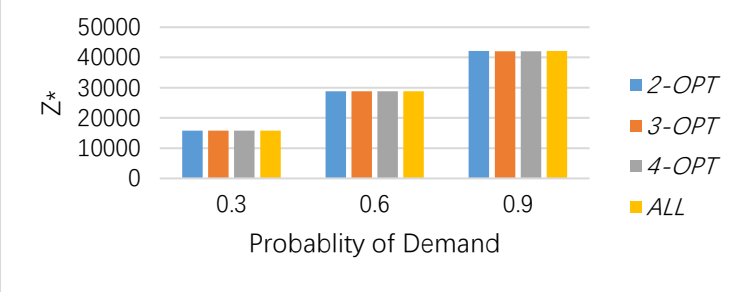


Fig. 3. Total cost under probability of 50%

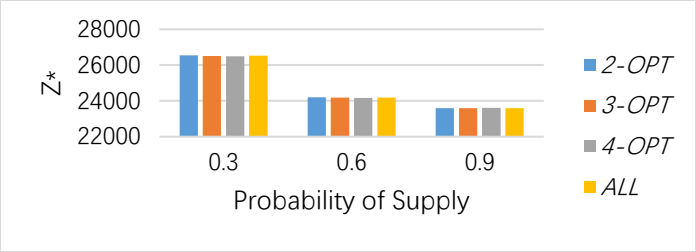


Fig. 4. Total cost under the same level of work needed on the job

To investigate the relationships between job size, worker availability, and match between demand for and supply of skills, we performed tests on 144 cases with 10 runs per case. The results of 1,440 randomly generated problems are reported at https://doi.org/10.18738/T8/RSXR1T. The parameter settings are alpha = 0.05 and NW = 30 on problems ranging from 200 to 400 jobs and 50 to 2,000 workers. The probability values for supply and demand are 0.01, 0.05, 0.15, 0.5, and 0.8, respectively. The results showed that mismatches between supply and demand affect the total cost. We report the average OFV for each case with different k-*Opt* strategies in Tables ST-1 to ST-9 in the Supplementary Material.

Tables AT.5 and AT.6 in the Appendix report the number of best solutions found, or the best CPU times found, on both types of problems (the ratio of jobs/workers is greater than 1 or less than 1) with the same average OFV of 10 randomly generated problems per 144 cases for each strategy. *ALL* strategy is better than the other three strategies on these test problems, especially on the large-scale problems. 2-*Opt* strategy is the least favorable among the four strategies. The complete results for different numbers of jobs and numbers of workers are given in Tables ST-10 and ST-11 in the Supplementary Material.

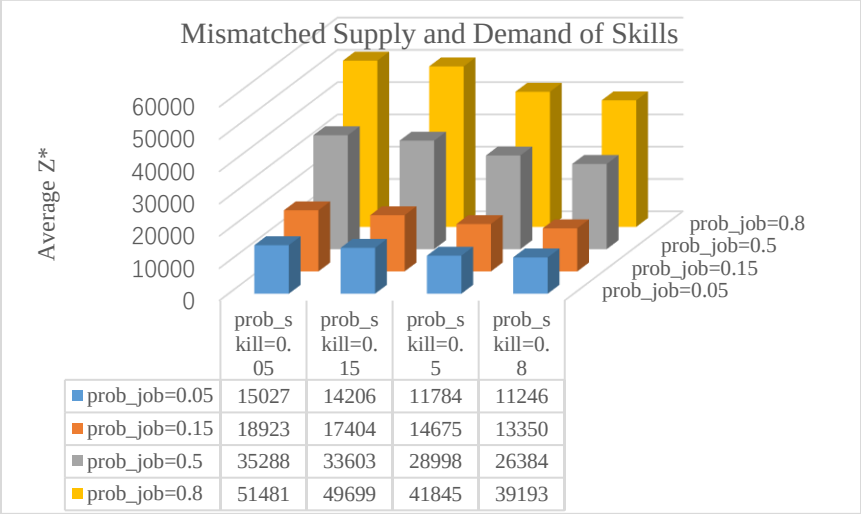
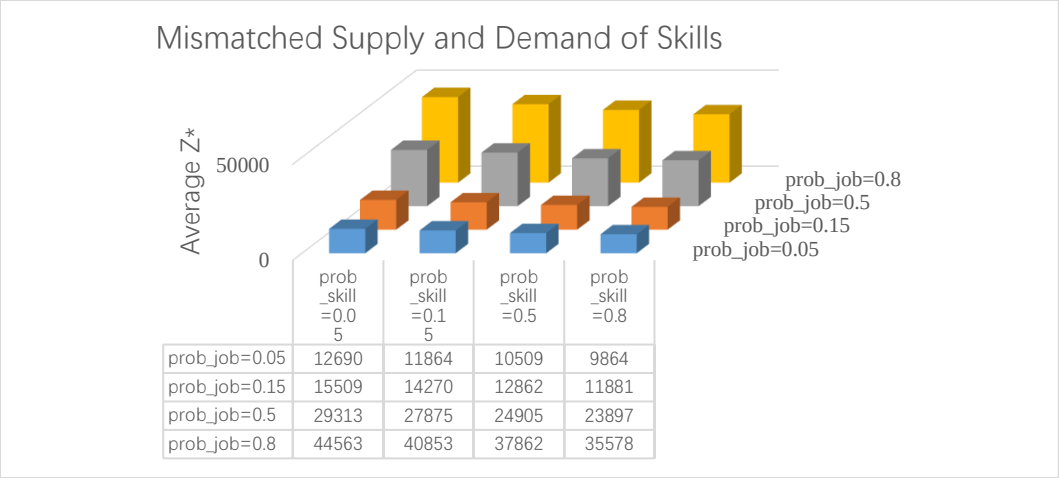
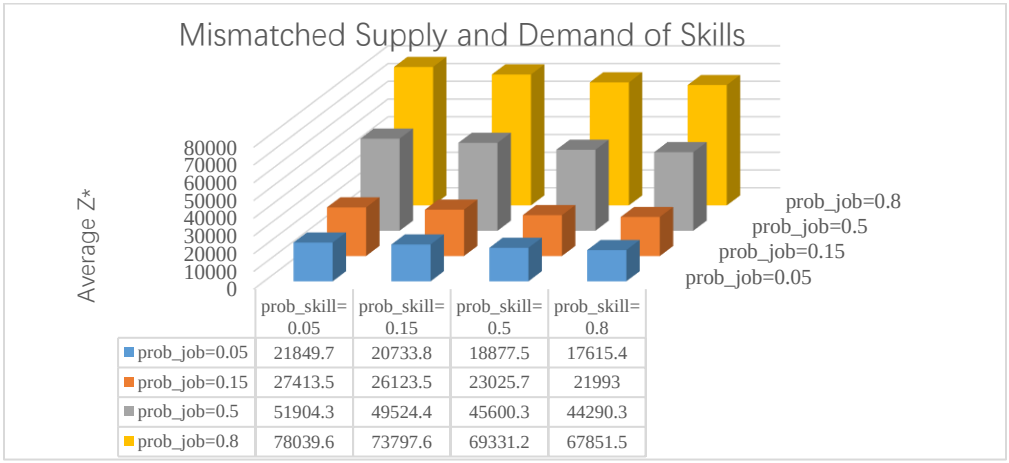


Fig. 5. Effect of mismatched supply and demand of skills on total cost (n=200, m=50)



	prob_skill=0.05	prob_skill=0.15	prob_skill=0.5	prob_skill=0.8
prob_job=0.05	12690	11864	10509	9864
prob_job=0.15	15509	14270	12862	11881
prob_job=0.5	29313	27875	24905	23897
prob_job=0.8	44563	40853	37862	35578

Fig. 6. Effect of mismatched supply and demand of skills on total cost (n=200, m=500)



	prob_skill=0.05	prob_skill=0.15	prob_skill=0.5	prob_skill=0.8
prob_job=0.05	21849.7	20733.8	18877.5	17615.4
prob_job=0.15	27413.5	26123.5	23025.7	21993
prob_job=0.5	51904.3	49524.4	45600.3	44290.3
prob_job=0.8	78039.6	73797.6	69331.2	67851.5

Fig. 7. Effect of mismatched supply and demand of skills on total cost (n=400, m=1000)

For all test problems, total cost increases when the demands for skills increase and reduces when supplies of skills increase (Fig. 5). For a given size of operation (job number of 200 or 400), the availability of workers reduces the total cost (Figs. 5 and 6). When the job size increases from 200 to 400 fields, the availability of workers also reduces total cost while the ratio between job number and available workers remains stable at 1:2.5. In Figs. 6 and 7, we reported the effect of mismatched skills on cost reduction when there are more workers available to managers. If qualified workers have more skills, then the total cost will be much lower (Fig. 8). For the same amount of work needed on a job, if qualified workers have more skills, it is less costly to finish all jobs, which is consistent with the results reported by Henao Ferrer, Muñoz, and Vera (2016) and Mac-Vicar, Ferrer, Muñoz, and Henao (2017) for the service sector.

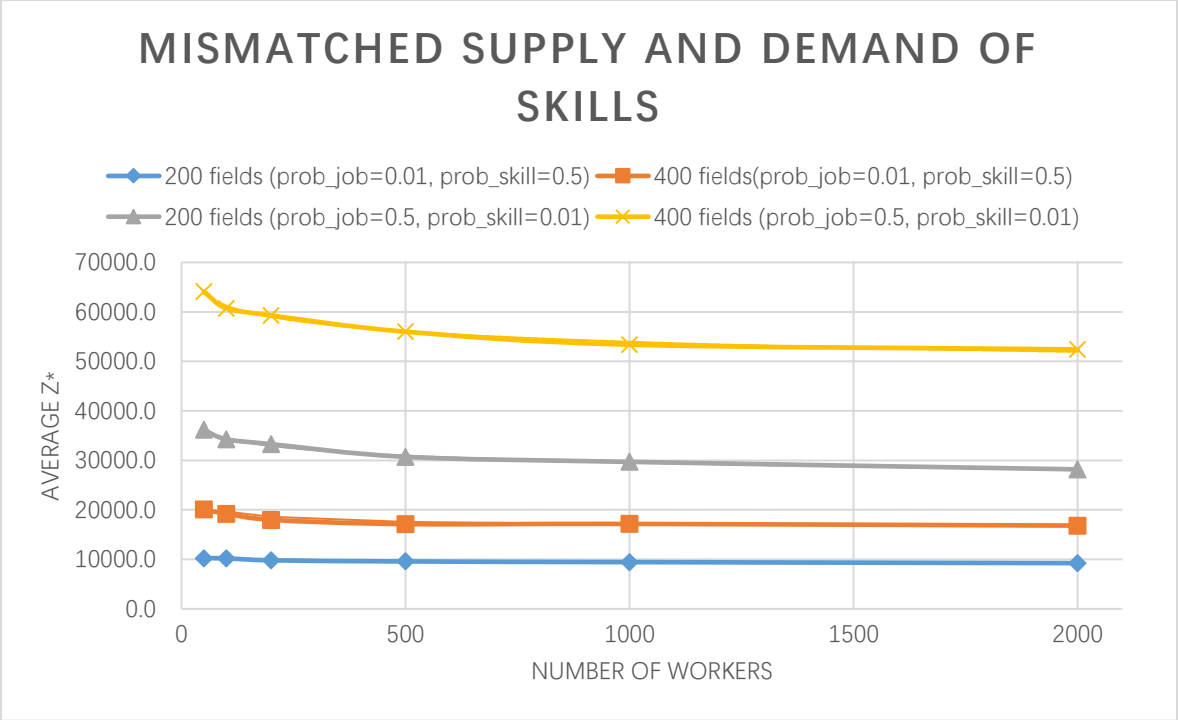


Fig. 8. Total cost of mismatched skills for different sizes of operation and available workers.

4.2 `Solution Quality and Parallelism

After the initial analysis of *k-Opt* heuristic, we evaluated its solution quality by comparing the results of Gurobi. First, we reported the optimal solutions for some instances found by Gurobi solver within the limit of computing resource and the best solutions found by the *k-Opt* heuristic in Tables 6 and 7. The computing time limit for the Gurobi solver is 288000 seconds and for *k-Opt* is 3600 seconds on single processor. We also reported the results for both Gurobi solver and *k-Opt* heuristic within 3600 seconds in Tables C.1 and C.2. Our initial analysis showed the smaller alpha values, which is associated with smaller p2 values, producing better solution quality on different instances. Thus, in the final analysis, we used alpha=0.01, 0.05, and 0.12 for all instances in the final analysis. We also used time limit as stopping criteria instead of NW values to allow the algorithms running until reaching 3600 seconds in the final analysis. In Tables 6 and 7, The data files of these instances are available at https://doi.org/10.18738/T8/RSXR1T. We reported the results for each instance with different k-*Opt* strategies in Tables ST-12 and ST-13 in the Supplementary Material. For small sized instances, *k-Opt* heuristic found the optimal solutions on all instances as Gurobi did and took less than 1 minute. For medium and large sized instances, the solutions found by the *k-Opt* heuristic have less than 1% GAP from the best solution found by

Gurobi while the *k-Opt* heuristic uses less than 1% of time on average. For very large sized instances with high demand (prob_job=0.8), the *k-Opt* heuristic found a better solution than Gurobi under the time limit (Tables 6 and 7).

We also explored the effect of parallelism of Gurobi for solving medium and large sized instances by setting "threads" parameters to allow it to run under a parallel computing environment. When we use "threads=2", the total computing time limit on two processors allowed by the solver is 200% of the sequential run. We found a super-linear speedup on some large instances, in which the rate of speedup is larger than the number of threads used. For two large sized instances with high job demand, Gurobi was unable to find initial integer solutions within a time limit in the sequential run while it found good integer solutions under parallel computing environment. Because Gurobi solver uses branch-and-bound, branch-and cut, and backtracking dynamically for solving a tree search method in the optimization procedure, the thread, which performed the process of a branch-and-bound node or the process of backtracking, might affect the behavior of other threads by updating the new global bound of the objective function, and short-circuit the evaluation of nodes in the solution search procedure. In addition, when we compared the process log files from sequential run and parallel run, we found some feasible solution points and cuts in the sequential run missing in the parallel run. This gave us some evidence on short-circuiting the evaluation of nodes. For some large sized instances in Tables 6 and 7, the computational time on "threads=2" is better than the one on "threads=4". This might indicate the search strategies and effects different threads might have on updating the global bounds and the evaluation of nodes. There are several other factors on linear speedup including a balanced search tree with lot of search required and build-in heuristics within the solver.

Table 6. Solution quality of k-*Opt* strategies and parallelism of Gurobi solver (prob_job=0.5 prob_skill=0.3, N_j=8)

	alpha=0.01 for K-Opt/ threads=1 for Gurobi		**alpha=0.05 for K-Opt/ threads=2 for Gurobi**		**alpha=0.12 for K-Opt/ threads=4 for Gurobi**	
100 jobs 200 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	19780[a]	0.53	19781[a]	1.25	19780[d]	0.02
Gurobi	19780 (19780)	53.14	19780 (19780)	56.01	19780 (19780)	32.09
150 jobs 300 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	28493[c]	2.492	28557[c]	0.019	28541[b]	0.076

Gurobi	28493 (28493)	873.9	28493 (28493)	523.116	28493 (28493)	319.21
200 jobs 400 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	34467[c]	5.789	34487[b]	15.013	34473[c]	5.243
Gurobi	34467 (34467)	190.197	34467 (34467)	166.845	34467 (34467)	106.373
250 jobs 500 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	41125[a]	24.384	41125[a]	38.344	41150[d]	85.19
Gurobi	41075 (41075)	4544.81	41075 (41075)	1401.05	41075 (41075)	1337.79
300 jobs 600 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	46784[c]	121.065	46864[b]	43.551	46887[a]	181.34
Gurobi	46784 (46784)	5956.57	46784 (46784)	1510.92	46784 (46784)	2863.5
400 jobs 800 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	59540[b]	121.22	59564[b]	184.325	59567[c]	630.327
Gurobi	59506 (59506)	312246	59506 (59506)	121303	59506 (59506)	185467
500 jobs 1000 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	60735[b]	127.406	60798[c]	120.264	60803[b]	623.172
Gurobi	60732* (59924)	1726470	60696* (60308)	563573	60682* (60499)	1150490

[a]2-Opt, [b]3-Opt,[c]4-Opt,[d]ALL algorithm, *Best found within the time limit

Table 7. Solution quality of k-*Opt* strategies and parallelism of Gurobi solver (prob_job=0.8 prob_skill=0.5, N_j=8)

	alpha=0.01 for K-Opt/ threads=1 for Gurobi		**alpha=0.05 for K-Opt/ threads=2 for Gurobi**		**alpha=0.12 for K-Opt/ threads=4 for Gurobi**	
100 jobs 200 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	25596[c]	0.566	25607[b]	1.338	25609[b]	5.11
Gurobi	25596 (25596)	215.713	25596 (25596)	181.518	25596 (25596)	109.779
150 jobs 300 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**

k-Opts	36799[b]	2.076	36813[c]	11.813	36815[b]	16.729
Gurobi	36799	16149.6	36799	17041.5	36799	4979.2
	(36799)		(36799)		(36799)	
200 jobs 400 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	47466[a]	29.293	47528[d]	46.492	47587[d]	15.892
Gurobi	47466	23144.6	47466	5299.29	47466	7420.67
	(47466)		(47466)		(47466)	
250 jobs 500 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	57519[a]	32.648	57554[d]	134.75	57562[c]	221.79
Gurobi	57519	25023.3	57519	6458.2	57519	11527.4
	(57519)		(57519)		(57519)	
300 jobs 600 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	59876[d]	151.283	59944[b]	66.065	59897[b]	323.77
Gurobi	59932*	288029	59878*	575682	59856*	1151460
	(58976)		(59477)		(59480)	
400 jobs 800 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	58799[d]	37.527	58773[c]	2.617	58873[d]	104.88
Gurobi	59562*	288306	58849*	574757	58968*	1151250
	(57854)		(57966)		(58096)	
500 jobs 1000 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	81289[c]	2075.01	81414[c]	6.609	81458[c]	145.41
Gurobi	83229*	288065	82272*	575530	81983	1151130
	(79882)		(79979)		(80002)	

[a]2-Opt, [b]3-Opt,[c]4-Opt,[d]ALL algorithm, *Best found within the time limit

5. Discussion and conclusion

5.1 Managerial implications

Seasonal businesses use different types of workers for the job to help keep labor costs in order to remain competitive. For example, recent economic and labor statistics from the U.S. Department of Agriculture suggests that a multi-skilled workforce could be an effective response to a declining labor force and high labor costs. In other business operations, such as event and tourist operations, hiring and training large number of seasonal workers in a short time windows, it is highly challenged to balance the labor utilization and cost of training. The results of this study showed how to seek the optimal balance with an effective

model and solution approach to such large scale assignment problem.

The MIP model is popular for machine scheduling and workforce management because it is easy to integrate multiple objectives and accommodate different types of resource constraints. The model proposed in this study focused on reducing the cost of production while increasing resource utilization, which is the main finding of our analysis. The cross-training of workers could help to develop a multi-skilled workforce and maximize use of the skills offered by qualified workers. Strategic workforce planning should be a high priority as technology advances.

This study examines an integrated model to strategically balance scheduling of multi-skilled workers and workforce management of cross-training. A multi-skilled workforce is able to meet the demands of a task and fully utilize the skill of worker, leading to lower total costs. In fact, cross-training workers in multiple skills instead of hiring seasonal workers could help seasonal businesses to achieve their goal of improving productivity. Thus, the integrated model proposed here can help seasonal businesses to balance labor costs and human capital investments while developing a multi-skilled workforce that will exceed the demand for jobs.

As pointed out earlier, both MIP models are NP hard in general and it might become computationally challenging to the planner if the number of jobs/tasks and workers increase. The computational time reported by our algorithm offered the technical feasibility to consider these conditions as additional constraints in the model. The MIP models can be calibrated to real data from seasonal businesses as a flexible planning tool. The results of the numerical study showed that the algorithm performed well after selecting the appropriate values for alpha and the number of runs. It took less than one hour to compute most large sized realistic instances and decision makers could adjust the plan in real time. Our finding is also applicable to many industries that depend heavily on automation and multi-skilled workers.

Nowadays, many scheduling software use cloud computing platforms that run on multiple servers and thousands of processors. Our exploration on solver parallelism provided an alternative general solution approach on solving large-sized instances using off-self commercial solvers in addition to developing heuristics. However, if we consider the use of computing resources, heuristic algorithm is more efficient comparing to solvers.

5.2 Limitations and future research

Our integrated MIP models could benefit from a qualitative study to obtain more information on pay scales based on skill sets and labor market demand. Such a qualitative study could also provide information on important soft skills such as knowledge management, learning ability, collaboration, and awareness of the environment to improve modeling of complex environments. In addition, our proposed MIP models did not

account for workers' ideal shifts, preferred jobs, and desired skills for cross-training. Many conditions affect the usage of an integrated model for a planning horizon including uncertainty of demand, a complex labor environment, volatile salaries and wages, high training costs, and other uncontrollable supply chain risks. We also did not consider workers' absences due to sickness or personal issues for the planning horizon. However, for a planning horizon longer than one week, the manager would have to take absences into account for purposes of rescheduling tasks. In addition, we did not consider the natural interdependency of tasks in the jobs when we created the initial solution for worker and job sequence due to the seasonal planning reality. For example in agriculture business, weed removal should be performed before tilling, and tilling should be performed before fertilizing. Future lines of research could also analyze these conditions and interdependent structures. The results of this study showed that some combined settings produced more favorable solutions. In fact, the proposed models and the algorithm could be replicated in other multitasking, multi-skilled applications by using a similar data structure and including constraints such as available training time or budget, machine-oriented training options, productivity losses of mentors, and other cross-training-related resource limitations. We will report these findings in future papers.

Acknowledgements:

The authors would like to thank the editor and two anonymous referees for their constructive and helpful comments on initial version of this paper. Their careful readings have helped us produce an improved paper**.**

Appendix A

Recent Development and Applications of *K-Opt* Method

Reference	Applications	*k-Opt* as standalone	*k-Opt* as part of integrated method	Comparative method or solver for comparison
Hansen and Mladenović (2001)	Facility Location , P-Median problem, Cluster analysis, Bilinear QP with bilinear constraints	Yes	No	Tabu Search
Merz and	Unconstrained	Yes	Yes, with greedy heuristics	No

Freisleben (2002)	Binary Quadratic Programming			
Fournier and Pierre (2005)	Mobile Network Design	No	Yes, with Ant Colony	Tabu Search
Katayama, Hamamoto, and Narihisa (2005)	Maximum Clique	Yes	No	Genetic, iterated and multistart local search from DIMACS
Sadfi et al. (2005)	Machine Scheduling	Yes	No	No
Shylo, Prokopyev, and Shylo (2008)	MAX-SAT	Yes	Yes, as part of global equilibrium search framework	Tabu Search, Simulated Annealing
Rodríguez-Martín and Salazar-González (2010)	Fixed Charge Transportation Problem	No	Yes, as local branch algorithm for CPLEX solver	Tabu Search
Blazinskas and Misevicius (2011)	TSP	Yes	No	No
Mladenović et al. (2012)	TSP	Yes	No	GRASP and GA
Mahi, Baykan, and Kodaz (2015)	TSP	No	Yes, with Particle Swarm and Ant Colony	Neural networks, Simulated Annealing and other hybrid ACO
Alidaee and Wang (2017)	Maximum Diversity	No	Yes. Tabu Search	No, but reported new best solutions for benchmark instances
Alidaee, Sloan, and Wang (2017)	MAX-Cut	No	Yes, Tabu Search	No, but reported new best solutions for benchmark

				instances
Gokbayrak and Yıldırım (2017)	wireless mesh networks	Yes	No	CPLEX solver
Wang and Alidaee (2018)	Machine Scheduling	No	Yes, Tabu Search	CPLEX solver
Papageorgiou et al. (2018)	Maritime Inventory Routing	Yes	No	CPLEX solver
Wang and Alidaee (2019)	Machine Scheduling	No	Yes, Tabu Search and Genetic Algorithm	IG algorithm
Taillard and Helsgaun (2019)	TSP	Yes	Yes, part of *Partial OPtimization Metaheuristic Under Special Intensification Conditions* framework	No, comparing to TSPLIB instances

Appendix B

Comparison between MSWM and UFL problems

In Section 3.2, we stated MSWM problem is the special case of an uncapacitated facility location (UFL) problem. To compare both problems in term of similarity, the description of UFL is given as follows: a set of clients $I = \{1, \cdots, m\}$ is served by a set of facilities $J = \{1, \cdots, n\}$, if client *i* is served by facility *j*, the cost is c_{ij}, f_j is the fixed cost to operate facility *j* and *p* is the maximum number of facilities to operate. The values of two sets of binary decision variables: x_{ij} and y_j, are determined to be 1 if facility *j* serves client *i* and facility j is in operation, 0 otherwise.

The integer programming formulation for UFL is given (Galvão and Raggi 1989):

$$Min\ Z = \sum_{j=1}^{n} a_i\, y_i + \sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij} x_{ij} \quad \text{(B.1)}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i \quad \text{(B.2)}$$

$$\sum_{j=1}^{n} y_j \leq p, \quad \text{(B.3)}$$

$$x_{ij} \leq y_j, \qquad \forall j, i \quad \text{(B.4)}$$

$$y_i, x_{ij} \in \{0,1\} \qquad \forall j, i \quad \text{(B.5)}$$

Constraints (B.2) ensure each client is served by a facility, constraint (B.3) ensures at most p facilities can

be operated. Constraints (B.4) ensure a client can be served by a facility only if the facility is in operation. If we consider a task in MSWM problem as the client in UFL problem, the multi-skilled worker as the facility in UFL problem, the task does not require specific skill from the worker, then two problems are equivalent. The objective function of MSWM can be written as B.1 in UFL, constraints (1) in MSWM are equivalent to B.2 in UFL, constraints (2.2) in MSWM are equivalent to B.4 in UFL and constraints (2.1) in MSWM can be rewritten as B.3 in UFL.

However, the task in MSWM problem requires specific skill from the worker, thus it is special case of UFL problem, where the client with specific demand is served by the facility with specific function to meet the demand.

The UFL problem and its variants have many applications in the literature {Galvão, 1989 #143;Correia, 2016 #142;Melo, 2006 #141}. These applications are formulated as MILP model and solved by CPLEX.

Appendix C

Computational Results of k-*Opt* Heuristic and Gurobi solver under 3600 seconds time limit

Table C1. Solution quality of k-*Opt* strategies and parallelism of Gurobi solver (prob_job=0.5 prob_skill=0.3, N_j=8)

	alpha=0.01 for K-Opt/ threads=1 for Gurobi		**alpha=0.05 for K-Opt/ threads=2 for Gurobi**		**alpha=0.12 for K-Opt/ threads=4 for Gurobi**	
100 jobs 200 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	19780[a]	0.53	19781[a]	1.25	19780[d]	0.02
Gurobi	19780 (19780)	53.14	19780 (19780)	56.01	19780 (19780)	32.09
150 jobs 300 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	28493[c]	2.492	28557[c]	0.019	28541[b]	0.076
Gurobi	28493 (28493)	873.9	28493 (28493)	523.116	28493 (28493)	319.21
200 jobs 400 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	34467[c]	5.789	34487[b]	15.013	34473[c]	5.243
Gurobi	34467 (34467)	190.197	34467 (34467)	166.845	34467 (34467)	106.373
250 jobs 500 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	41125[a]	24.384	41125[a]	38.344	41150[d]	85.19
Gurobi	41075	694.05	41075	758.93	41075	514.17

	(41075)		(41075)		(41075)	
300 jobs 600 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	46784[c]	121.065	46864[b]	43.551	46887[a]	181.34
Gurobi	46784	1819.16	46784	883.29	46784	1251.40
	(46784)		(46784)		(46784)	
400 jobs 800 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	59540[b]	121.22	59564[b]	184.325	59567[c]	630.327
Gurobi	60195	3600	60166	3600	59551	3600
	(59006)		(58878)		(58878)	
500 jobs 1000 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	60735[b]	127.406	60798[c]	120.264	60803[b]	623.172
Gurobi	60821	3600	60852	3600	60997	3600
	(59926)		(59924)		(59925)	

[a]2-Opt, [b]3-Opt,[c]4-Opt,[d]ALL algorithm

Table C2. Solution quality of k-*Opt* strategies and parallelism of Gurobi solver (prob_job=0.8 prob_skill=0.5, N_j=8)

	alpha=0.01 for K-Opt/ threads=1 for Gurobi		**alpha=0.05 for K-Opt/ threads=2 for Gurobi**		**alpha=0.12 for K-Opt/ threads=4 for Gurobi**	
100 jobs 200 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	25596[c]	0.566	25607[b]	1.338	25609[b]	5.11
Gurobi	25596	215.713	25596	181.518	25596	109.779
	(25596)		(25596)		(25596)	
150 jobs 300 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	36799[b]	2.076	36813 [c]	11.813	36815[b]	16.729
Gurobi	36892	3600	36799	2370	36799	2013
	(36489)		(36799)		(36799)	
200 jobs 400 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	47466[a]	29.293	47528[d]	46.492	47587[d]	15.892
Gurobi	47518	3600	47506	3600	47466	3486.14
	(47302)		(47341)		(47466)	
250 jobs 500 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	57519[a]	32.648	57554[d]	134.75	57562[c]	221.79

Gurobi	57519 (57519)	2613.39	57519 (57519)	2901.60	57519 (57519)	3326.79
300 jobs 600 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	59876[d]	151.283	59957[d]	301.52	59897[b]	323.77
Gurobi	60636 (59029)	3600	60069 (58977)	3600	60123 (59029)	3600
400 jobs 800 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	58799[d]	37.527	58773[c]	2.617	58873[d]	104.88
Gurobi	59906 (57965)	3600	59624 (57935)	3600	59403 (57995)	3600
500 jobs 1000 workers	**Z***	**TB[s]**	**Z***	**TB[s]**	**Z***	**TB[s]**
k-Opts	81289[c]	2075.01	81414[c]	6.609	81458[c]	145.41
Gurobi	1888989 (0)	3600	83229 (79886)	3600	83229 (79886)	3600

[a]2-Opt, [b]3-Opt,[c]4-Opt,[d]ALL algorithm

References

Alidaee, Bahram, Hugh Sloan, and Haibo Wang. 2017. "Simple and fast novel diversification approach for the UBQP based on sequential improvement local search." *Computers & Industrial Engineering* 111:164-75. doi: https://doi.org/10.1016/j.cie.2017.07.012.

Alidaee, Bahram, and Haibo Wang. 2017. "A note on heuristic approach based on UBQP formulation of the maximum diversity problem." *Journal of the Operational Research Society* 68 (1):102-10. doi: 10.1057/s41274-016-0031-4.

Allen, Stuart J., and Edmund W. Schuster. 2004. "Controlling the Risk for an Agricultural Harvest." *Manufacturing & Service Operations Management* 6 (3):225-36. doi: 10.1287/msom.1040.0035.

Basnet, C. B., L. R. Foulds, and J. M. Wilson. 2006. "Scheduling contractors' farm-to-farm crop harvesting operations." *International Transactions in Operational Research* 13 (1):1-15. doi: 10.1111/j.1475-3995.2006.00530.x.

Blazinskas, Andrius, and Alfonsas Misevicius. 2011. combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem. Paper presented at the 17th International Conference on Information and Software Technologies.

Campbell, Gerard M. 1999. "Cross-Utilization of Workers Whose Capabilities Differ." *Management Science* 45 (5):722-32. doi: 10.1287/mnsc.45.5.722.

Cheng, Mei-Ying, Abhishek Gupta, Yew-Soon Ong, and Zhi-Wei Ni. 2017. "Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design." *Engineering Applications of Artificial Intelligence* 64:13-24. doi: https://doi.org/10.1016/j.engappai.2017.05.008.

Croes, Georges A. 1958. "A method for solving traveling-salesman problems." *Operations Research* 6 (6):791-812.

Ebeling, A. C., and C. Y. Lee. 1994. "Cross-training effectiveness and profitability." *International Journal of Production Research* 32 (12):2843-59. doi: 10.1080/00207549408957104.

Eden, Maya, and Paul Gaggl. 2018. "On the welfare implications of automation." *Review of Economic Dynamics* 29:15-43. doi: https://doi.org/10.1016/j.red.2017.12.003.

Edwards, Gareth, Claus G. Sørensen, Dionysis D. Bochtis, and Lars J. Munkholm. 2015. "Optimised schedules for sequential agricultural operations using a Tabu Search method." *Computers and Electronics in Agriculture* 117:102-13. doi: https://doi.org/10.1016/j.compag.2015.07.007.

Edwards, William, and Ann M. Johanns. "Wages and Benefits for Farm Employees." https://www.extension.iastate.edu/agdm/wholefarm/html/c1-60.html.

Fournier, Joseph R. L., and Samuel Pierre. 2005. "Assigning cells to switches in mobile networks using an ant colony optimization heuristic." *Computer Communications* 28 (1):65-73. doi: https://doi.org/10.1016/j.comcom.2004.07.006.

Galvão, Roberto Diéguez, and Luiz Aurélio Raggi. 1989. "A method for solving to optimality uncapacitated location problems." *Annals of Operations Research* 18 (1):225-44. doi: 10.1007/BF02097805.

Garey, Michael R., and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*: W. H. Freeman Co.

Gerhard, Ulrike, and Barbara Hahn. 2005. "Wal-Mart and Aldi: Two Retail Giants in Germany." *GeoJournal* 62 (1):15-26. doi: 10.1007/s10708-004-1930-5.

Glover, Fred, and Gary A. Kochenberger. 1996. "Critical Event Tabu Search for Multidimensional Knapsack Problems." In *Meta-Heuristics: Theory and Applications*, edited by Ibrahim H. Osman and James P. Kelly, 407-27. Boston, MA: Springer US.

Glover, Fred, Gary A. Kochenberger, and Bahram Alidaee. 1998. "Adaptive Memory Tabu Search for Binary Quadratic Programs." *Manage. Sci.* 44 (3):336-45. doi: 10.1287/mnsc.44.3.336.

Gnanlet, Adelina, and Wendell G. Gilland. 2014. "Impact of productivity on cross-training configurations and optimal staffing decisions in hospitals." *European Journal of Operational Research* 238 (1):254-69. doi: https://doi.org/10.1016/j.ejor.2014.03.033.

Gokbayrak, Kagan, and E. Alper Yıldırım. 2017. "Exact and heuristic approaches based on noninterfering transmissions for joint gateway selection, time slot allocation, routing and power control for wireless mesh networks." *Computers & Operations Research* 81:102-18. doi: 10.1016/j.cor.2016.09.021.

Gurobi Optimization. "Gurobi Optimizer 8.1." http://www.gurobi.com/.

Hall, Nicholas G., Joseph Y. ‐ T Leung, and Chung ‐ Lun Li. 2015. "The Effects of Multitasking on Operations Scheduling." *Production & Operations Management* 24 (8):1248-65. doi: 10.1111/poms.12331.

Hall, Nicholas G., Joseph Y. T. Leung, and Chung-Lun Li. 2016. "Multitasking via alternate and shared processing: Algorithms and complexity." *Discrete Applied Mathematics* 208:41-58. doi: https://doi.org/10.1016/j.dam.2016.03.018.

Hansen, Pierre, and Nenad Mladenović. 2001. "Variable neighborhood search: Principles and applications." *European Journal of Operational Research* 130 (3):449-67.

He, Pengfei, and Jing Li. 2019. "The two-echelon multi-trip vehicle routing problem with dynamic satellites for crop harvesting and transportation." *Applied Soft Computing* 77:387-98. doi: https://doi.org/10.1016/j.asoc.2019.01.040.

He, Pengfei, Jing Li, and Xin Wang. 2018. "Wheat harvest schedule model for agricultural machinery cooperatives considering fragmental farmlands." *Computers and Electronics in Agriculture* 145:226-34. doi: https://doi.org/10.1016/j.compag.2017.12.042.

He, Yong, Weiya Zhong, and Huikun Gu. 2006. "Improved algorithms for two single machine scheduling problems." *Theoretical Computer Science* 363 (3):257-65. doi: https://doi.org/10.1016/j.tcs.2006.04.014.

Helsgaun, Keld. 2009. "General k-opt submoves for the Lin–Kernighan TSP heuristic." *Mathematical Programming Computation* 1 (2-3):119-63.

Ji, Min, Wenya Zhang, Lijuan Liao, T. C. E. Cheng, and Yuanyuan Tan. 2019. "Multitasking parallel-machine scheduling with machine-dependent slack due-window assignment." *International Journal of Production Research* 57 (6):1667-84. doi: 10.1080/00207543.2018.1497312.

Katayama, Kengo, Akihiro Hamamoto, and Hiroyuki Narihisa. 2005. "An effective local search for the maximum clique problem." *Information Processing Letters* 95 (5):503-11. doi: https://doi.org/10.1016/j.ipl.2005.05.010.

Kim, Sungsu, and DavidA Nembhard. 2013. "Rule mining for scheduling cross training with a heterogeneous workforce." *International Journal of Production Research* 51 (8):2281-300. doi: 10.1080/00207543.2012.716169.

Knox, John. 1994. "Tabu search performance on the symmetric traveling salesman problem." *Computers & Operations Research* 21 (8):867-76. doi: https://doi.org/10.1016/0305-0548(94)90016-7.

Laporte, Gilbert, Stefan Nickel, and Francisco Saldanha da Gama. 2015. *Location science*. Vol. 528: Springer.

Lin, Shen. 1965. "Computer solutions of the traveling salesman problem." *Bell System Technical Journal* 44 (10):2245-69.

Lin, Shen, and Brian W Kernighan. 1973. "An effective heuristic algorithm for the traveling-salesman problem." *Operations Research* 21 (2):498-516.

Liu, Ming, Shijin Wang, Feifeng Zheng, and Chengbin Chu. 2017. "Algorithms for the joint multitasking scheduling and common due date assignment problem." *International Journal of Production Research* 55 (20):6052-66. doi: 10.1080/00207543.2017.1321804.

Lowe, Timothy J., and Paul V. Preckel. 2004. "Decision Technologies for Agribusiness Problems: A Brief Review of Selected Literature and a Call for Research." *Manufacturing & Service Operations Management* 6 (3):201-8. doi: 10.1287/msom.1040.0051.

Luce, Stephanie. 2013. "Global retail report." *Nyon, Switzerland: UNI Global Union*.

Lusa, Amaia, Albert Corominas, and Rafael Pastor. 2008. "An exact procedure for planning holidays and working time under annualized hours considering cross-trained workers with different efficiencies." *International Journal of Production Research* 46 (8):2123-42. doi: 10.1080/00207540601080480.

Mabert, Vincent A, and Michael J Showalter. 1990. "Measuring the impact of part-time workers in service organizations." *Journal of Operations Management* 9 (2):209-29.

Mahi, Mostafa, Ömer Kaan Baykan, and Halife Kodaz. 2015. "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem." *Applied Soft Computing* 30:484-90.

Malhotra, Manoj K., and Larry P. Ritzman. 1994. "Scheduling Flexibility in the Service Sector: A Postal Case Study." *Production and Operations Management* 3 (2):100-17. doi: 10.1111/j.1937-5956.1994.tb00113.x.

Merz, Peter, and Bernd Freisleben. 2002. "Greedy and local search heuristics for unconstrained binary quadratic programming." *Journal of heuristics* 8 (2):197-213.

Mladenović, Nenad, Dragan Urošević, Saı¨d Hanafi, and Aleksandar Ilić. 2012. "A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem." *European Journal of Operational Research* 220 (1):270-85. doi: https://doi.org/10.1016/j.ejor.2012.01.036.

Olivella, Jordi, Albert Corominas, and Rafael Pastor. 2013. "Task assignment considering cross-training goals and due dates." *International Journal of Production Research* 51 (3):952-62. doi: 10.1080/00207543.2012.693645.

Olivella, Jordi, and David Nembhard. 2016. "Calibrating cross-training to meet demand mix variation and employee absence." *European Journal of Operational Research* 248 (2):462-72. doi: https://doi.org/10.1016/j.ejor.2015.07.036.

Papageorgiou, Dimitri J, Myun-Seok Cheon, Stuart Harwood, Francisco Trespalacios, and George L Nemhauser. 2018. "Recent progress using matheuristics for strategic maritime inventory routing." In *Modeling, Computing and Data Handling Methodologies for Maritime Transportation*, 59-94. Springer.

Potvin, Jean-Yves. 1996. "Genetic algorithms for the traveling salesman problem." *Annals of Operations Research* 63 (3):337-70.

Rodríguez-Martín, Inmaculada, and Juan José Salazar-González. 2010. "A local branching heuristic for the capacitated fixed-charge network design problem." *Computers & Operations Research* 37 (3):575-81.

Sadfi, Cherif, Bernard Penz, Christophe Rapine, Jacek Błażewicz, and Piotr Formanowicz. 2005. "An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints." *European Journal of Operational Research* 161 (1):3-10. doi: https://doi.org/10.1016/j.ejor.2003.08.026.

Salem, Islam El-Bayoumi, and Mohamed Kamal Abdien. 2017. "Implementation of employee cross-training during perilous conditions in hotels." *Tourism Management Perspectives* 23:68-74. doi: https://doi.org/10.1016/j.tmp.2017.05.005.

Shakeri, Shakib, and Rasaratnam Logendran. 2007. "A mathematical programming-based scheduling framework for multitasking environments." *European Journal of Operational Research* 176 (1):193-209. doi: https://doi.org/10.1016/j.ejor.2005.06.062.

Shylo, Oleg V., Oleg A. Prokopyev, and Vladimir P. Shylo. 2008. "Solving weighted MAX-SAT via global equilibrium search." *Operations Research Letters* 36 (4):434-8. doi: https://doi.org/10.1016/j.orl.2007.11.007.

Sun, Minghe, Jay E. Aronson, Patrick G. McKeown, and Dennis Drinka. 1998. "A tabu search heuristic procedure for the fixed charge transportation problem." *European Journal of Operational Research* 106 (2):441-56. doi: https://doi.org/10.1016/S0377-2217(97)00284-1.

Taillard, Éric D., and Keld Helsgaun. 2019. "POPMUSIC for the travelling salesman problem." *European Journal of Operational Research* 272 (2):420-9. doi: https://doi.org/10.1016/j.ejor.2018.06.039.

Telhada, João. 2014. "Alternative MIP formulations for an integrated shift scheduling and task assignment problem." *Discrete Applied Mathematics* 164:328-43. doi: https://doi.org/10.1016/j.dam.2013.04.021.

United States Department of Labor. 2018. "Occupational Employment Statistics." Accessed May 10. https://www.bls.gov/oes/current/oes131074.htm.

Van Elderen, E. 1980. "Scheduling farm operations." *European Journal of Operational Research* 4 (1):19-23. doi: https://doi.org/10.1016/0377-2217(80)90035-1.

Wang, Haibo, and Bahram Alidaee. 2018. "Unrelated Parallel Machine Selection and Job Scheduling With the Objective of Minimizing Total Workload and Machine Fixed Costs." *IEEE Transactions on Automation Science and Engineering*:1-9. doi: 10.1109/TASE.2018.2832440.

———. 2019. "Effective Heuristic for Large-Scale Unrelated Parallel Machines Scheduling Problems." *Omega* 83:261-74. doi: https://doi.org/10.1016/j.omega.2018.07.005.

Wijngaard, Peter J. M. 1988. "A heuristic for scheduling problems, especially for scheduling farm operations." *European Journal of Operational Research* 37 (1):127-35. doi: https://doi.org/10.1016/0377-2217(88)90287-1.

Wishon, C., J. R. Villalobos, N. Mason, H. Flores, and G. Lujan. 2015. "Use of MIP for planning temporary immigrant farm labor force." *International Journal of Production Economics* 170:25-33. doi: https://doi.org/10.1016/j.ijpe.2015.09.004.

Zhu, Zhanguo, Feifeng Zheng, and Chengbin Chu. 2017. "Multitasking scheduling problems with a rate-modifying activity." *International Journal of Production Research* 55 (1):296-312. doi: 10.1080/00207543.2016.1208852.